Amended version of a paper appeared in N. Guarino (ed.), *Formal Ontology in Information Systems. Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998.* Amsterdam, IOS Press, pp. 3-15.

Formal Ontology and Information Systems

Nicola Guarino
National Research Council, LADSEB-CNR, Corso Stati Uniti 4, I-35127 Padova, Italy
guarino@ladseb.pd.cnr.it

Abstract. Research on ontology is becoming increasingly widespread in the computer science community, and its importance is being recognized in a multiplicity of research fields and application areas, including knowledge engineering, database design and integration, information retrieval and extraction. We shall use the generic term "information systems", in its broadest sense, to collectively refer to these application perspectives. We argue in this paper that so-called ontologies present their own methodological and architectural peculiarities: on the methodological side, their main peculiarity is the adoption of a highly interdisciplinary approach, while on the architectural side the most interesting aspect is the centrality of the role they can play in an information system, leading to the perspective of *ontology-driven information systems.*

1 Introduction

Research on ontology is becoming increasingly widespread in the computer science community. While this term has been rather confined to the philosophical sphere in the past, it is now gaining a specific role in Artificial Intelligence, Computational Linguistics, and Database Theory. In particular, its importance is being recognized in research fields as diverse as knowledge engineering [20,45,15,18], knowledge representation [23,2,42], qualitative modelling [19,9,13], language engineering [31,5], database design [11,47], information modelling [3,53], information integration [55,7,35], object-oriented analysis [51,39], information retrieval and extraction [24,6,34,54], knowledge management and organization [40], agent-based systems design[1]. Current applications areas are disparate, including enterprise integration [22,46], natural language translation [30,33], medicine [16], mechanical engineering [10], standardization of product knowledge [8,4,26], electronic commerce [32], geographic information systems [12], legal information systems[2], biological information systems[3]. I shall use the generic term *information systems*, in its broadest sense, to collectively refer to these fields and application areas.

In some cases, the term "ontology" is just a fancy name denoting the result of familiar activities like conceptual analysis and domain modelling, carried out by means of standard methodologies. In many cases, however, so-called ontologies present their own *methodological* and *architectural* peculiarities. On the methodological side, the main peculiarity is the adoption of a *highly interdisciplinary approach*, where philosophy and linguistics play a fundamental role in analyzing the structure of a given reality at a high level of generality and in formulating a clear and rigorous vocabulary. On the architectural side, the most interest-

[1] See the recent FIPA (Federation for Intelligent Physical Agents) call for proposals in the area of ontology management at http://drogo.cselt.it/fipa/palo_alto/cfp4.htm
[2] See the proceedings of the LEGONT'97 workshop at http://www.csc.liv.ac.uk/~pepijn/legont.html
[3] See the forthcoming workshop on "Semantic Foundations for Molecular Biology: Schemata, Controlled Vocabularies and Ontologies" at http://www-lbit.iro.umontreal.ca/ISMB98/anglais/ontology.html

ing aspect is the centrality of the role that an ontology can play in an information system, leading to the perspective of *ontology-driven information systems.*

I have stressed elsewhere [23,24] the importance of an interdisciplinary approach in the practice of ontological engineering, underlying in particular the role played by *formal ontology*. Since this topic is going to be addressed in more detail in the present volume [41,50], I will avoid here any further comment.

Rather, I will first propose the most recent elaboration of the clarification work on the way the term "ontology" is being used in computer science originally presented in [27]. Although some progress has been made, I believe there still a good deal of terminological and conceptual confusion, and I will try therefore to further clarify – with respect to the past work – the notions of *ontology*, *ontological commitment*, and *conceptualization.* I shall then introduce the perspective of ontology-driven information systems, showing how ontologies can play a central role by impacting the main components of an information system: information resources, user interfaces, and application programs.

2. Ontology and ontologies

Since this paper is deliberately addressed to an interdisciplinary audience, it is advisable to pay attention to some preliminary terminological clarifications, especially because some crucial terms appear to be used with different senses in different communities[4]. Let us first consider the distinction between "Ontology" (with the capital "o"), as in the statement "Ontology is a fascinating discipline" and "ontology" (with the lowercase "o"), as in the expressions "Aristotle's ontology" or "CYC's ontology". The same term has an uncountable reading in the former case, and a countable reading in the latter. While the former reading seems to be reasonably clear (as referring to a particular philosophical discipline), two different senses are assumed by the philosophical community and the Artificial Intelligence community (and, in general, the whole computer science community) for the latter term.

In the philosophical sense, we may refer to an ontology as a particular system of categories accounting for a certain vision of the world. As such, this system does not depend on a particular *language*: Aristotle's ontology is always the same, independently of the language used to describe it. On the other hand, in its most prevalent use in AI, an ontology refers to an *engineering artifact*, constituted by a specific *vocabulary* used to describe a certain reality, plus a set of explicit assumptions regarding the *intended meaning* of the vocabulary words. This set of assumptions has usually the form of a first-order logical theory[5], where vocabulary words appear as unary or binary predicate names, respectively called concepts and relations. In the simplest case, an ontology describes a hierarchy of concepts related by subsumption relationships; in more sophisticated cases, suitable axioms are added in order to express other relationships between concepts and to constrain their intended interpretation.

The two readings of "ontology" described above are indeed related each other, but in order to solve the terminological impasse we need to choose one of them, inventing a new name for the other: we shall adopt the AI reading, using the word *conceptualization* to refer to the philosophical reading. So two ontologies can be different in the vocabulary used (using English or Italian words, for instance) while sharing the same conceptualization.

[4] I elaborate here on some material already published in [27]

[5] In this case, an ontology is sometimes called a *formal* ontology, although we shall use the expression "formal ontology" only to refer to a philosophical research field.

2.1 What is a conceptualization.

The notion of conceptualization introduced above requires however a suitable formalization, since it may generate some confusions. Indeed, a conceptualization has been defined in a well-known AI textbook [17] as a structure <D,**R**>, where D is a domain and **R** is a set or relevant relations on D[6]. This definition has been then used by Tom Gruber, who defined an ontology as "a specification of a conceptualization" [21]. Together with Pierdaniele Giaretta, I have discussed such a definition in [27], arguing that, in order for it to have some sense, a different, *intensional* account of the notion of conceptualization has to be introduced. I try here to further clarify these notions, making clear the relationship between an ontology, its intended models, and a conceptualization.

The problem with Genesereth and Nilsson's notion of conceptualization is that it refers to ordinary mathematical relations on D, i.e. *extensional* relations. These relations reflect a *particular* state of affairs: for instance, in the blocks world, they may reflect a particular arrangement of blocks on the table. We need instead to focus on the *meaning* of these relations, independently of a state of affairs: for instance, the meaning of the "above" relation lies in the *way* it refers to certain couples of blocks according to their spatial arrangement. We need therefore to speak of *intensional* relations: we shall call them *conceptual relations*, reserving the simple term "relation" to ordinary mathematical relations.

A standard way to represent intensions (and therefore conceptual relations) is to see them as functions from possible worlds into sets. This has some disadvantages[7], but works fairly well for our purposes. While ordinary relations are defined on a certain domain, conceptual relations are defined on a *domain space*. We shall define a domain space as a structure <D, W>, where D is a domain and W is a set of maximal states of affairs of such domain (also called *possible worlds*). For instance, D may be a set of blocks on a table and W can be the set of all possible spatial arrangements of these blocks. Given a domain space <D, W>, we shall define a *conceptual relation* ρ^n of arity *n* on <D, W> as a total function $\rho^n: W \rightarrow 2^{D^n}$ from W into the set of all n-ary (ordinary) relations on D. For a generic conceptual relation ρ, the set $\mathbf{E}_\rho = \{\rho(w) \mid w \in W\}$ will contain the *admittable extensions* of ρ. A *conceptualization* for D can be now defined as an ordered triple $\mathbf{C} = \langle D, W, \mathfrak{R} \rangle$, where $\mathfrak{R}$ is a set of conceptual relations on the domain space <D, W>[8]. We can say therefore that a conceptualization is a set of conceptual relations defined on a domain space.

Consider now the structure <D, **R**> introduced in [17]. Since it refers to a particular world (or state of affairs), we shall call it a *world structure*. It is easy to see that a conceptualization contains many of such world structures, one for each world: they shall be called the *intended world structures* according to such conceptualization.

Let $\mathbf{C} = \langle D, W, \mathfrak{R} \rangle$ be a conceptualization. For each possible world $w \in W$, the intended structure of w according to **C** is the structure $\mathbf{S}_{w\mathbf{C}} = \langle D, \mathbf{R}_{w\mathbf{C}} \rangle$, where $\mathbf{R}_{w\mathbf{C}} = \{\rho(w) \mid \rho \in \mathfrak{R}\}$ is the set of extensions (relative to w) of the elements of $\mathfrak{R}$. We shall denote with $\mathbf{S}_\mathbf{C}$ the set $\{\mathbf{S}_{w\mathbf{C}} \mid w \in W\}$ all the intended world structures of **C**.

---

[6] In a subsequent paper [37], Nils Nilsson stresses the importance of the conceptualization for a modeling task.

[7] For instance, the two relations "trilateral" and "triangle" turn out to be the same, as they have the same extension in all possible worlds.

[8] In the following, symbols denoting structures and sets of sets appear in boldface.

Let us consider now a logical language **L**, with a vocabulary V. Rearranging the standard definition, we can define a *model* for **L** as a structure <**S**, I>, where **S** = <D, **R**> is a world structure and I: V → D **R** is an interpretation function assigning elements of D to constant symbols of V, and elements of **R** to predicate symbols of V. As well known, a model fixes therefore a particular extensional interpretation of the language. Analogously, we can fix an *intensional* interpretation by means of a structure <**C**, >, where **C** = <D, W, > is a conceptualization and : V → D is a function assigning elements of D to constant symbols of V, and elements of to predicate symbols of V. We shall call this intensional interpretation an *ontological commitment* for L. If **K** = <**C**, > is a an ontological commitment for **L**, we say that **L** *commits* to **C** by means of **K**, while **C** is the *underlying conceptualization* of **K**[9].

Given a language **L** with vocabulary V, and an ontological commitment **K** = <**C**, > for **L**, a model <**S**, I> will be *compatible* with **K** if: i) **S** $\mathbf{S}_{\mathbf{C}}$; ii) for each constant *c*, I(*c*) = (*c*); iii) there exists a world *w* such that[10], for each predicate symbol *p*, I maps such predicate into an admittable extension of (*p*), i.e. there exists a conceptual relation ρ such that (*p*) = ρ ρ(*w*) = I(*p*). The set $\mathbf{I}_{\mathbf{K}}(\mathbf{L})$ of all models of **L** that are compatible with **K** will be called the set of *intended models* of **L** according to **K**.

In general, there will be no way to reconstruct the ontological commitment of a language from a set of its intended models, since a model does not necessarily reflect a particular world: in fact, since the relevant relations considered may not be enough to completely characterize a state of affairs, a model may actually describe a situation common to *many* states of affairs. This means that it is impossible to reconstruct the correspondence between worlds and extensional relations established by the underlying conceptualization. A set of intended models is therefore only a *weak* characterization of a conceptualization: it just excludes some absurd interpretations, without really describing the "meaning" of the vocabulary.

2.2 What is an ontology

We can now clarify the role of an ontology, considered as a set of logical axioms designed to account for the intended meaning of a vocabulary. Given a language **L** with ontological commitment **K**, an ontology for **L** is a set of axioms designed in a way such that the set of its models approximates as best as possible the set of intended models of **L** according to **K** (Fig. 1). In general, it is not easy (nor always convenient) to find the right set of axioms, so that an ontology will admit other models besides the intended ones. Therefore, an ontology can "specify" a conceptualization only in a very indirect way, since i) it can only approximate a set of intended models; ii) such a set of intended models is only a weak characterization of a conceptualization[11]. We shall say that an ontology **O** for a language **L** *approximates* a conceptualization **C** if there exists an ontological commitment **K** = <**C**, > such that the intended models of **L** according to **K** are included in the models of **O**. An

---

[9] The expression "ontological commitment" has been sometimes used to denote the *result* of the commitment itself, i.e., in our terminology, the underlying conceptualization.

[10] There is a mistake in the version of this paper published on the FOIS'98 proceedings, due to an erroneous quantification on possible worlds. The present version is the correct one. The point is that a compatible model must pick up a single possible world.

[11] This statement clarifies a possible confusion arising from [25], where I stated that the definition "an ontology is an explicit, partial account of a conceptualization" can be rephrased as "an ontology is an explicit, partial account of the intended models of a logical language". We have seen that a conceptualization is not equivalent to a set of intended models, so the two statements are not equivalent (although the former implies the latter).

ontology *commits* to **C** if i) it has been designed with the purpose of characterizing **C**, and ii) it approximates **C**. A language **L** *commits* to an ontology **O** if it commits to some conceptualization **C** such that **O** agrees on **C**. With these clarifications, we come up to the following definition, which refines Gruber's definition by making clear the difference between an ontology and a conceptualization:

> An ontology is a logical theory accounting for the *intended meaning* of a formal vocabulary[12], i.e. its *ontological commitment* to a particular *conceptualization* of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models.

The relationships between vocabulary, conceptualization, ontological commitment and ontology are illustrated in Fig. 1. It is important to stress that an ontology is *language-dependent*, while a conceptualization is *language-independent*. In its *de facto* use in AI, the term "ontology" collapses the two aspects, but a clear separation between them becomes essential to address the issues related to *ontology sharing, fusion,* and *translation,* which in general imply multiple vocabularies and multiple conceptualizations.

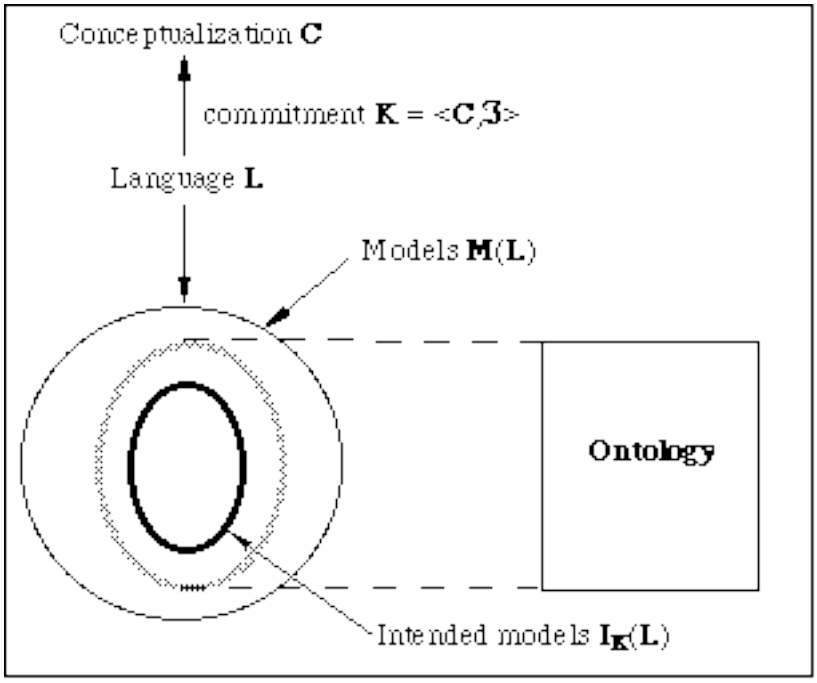

Fig. 1. The intended models of a logical language reflect its commitment to a conceptualization. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating this set of intended models.

2.3 Coarse vs. fine-grained ontologies

As an ontology only indirectly accounts for a conceptualization, we may wonder *how much* can it get close to that. In fact, we can classify ontologies according to their *accuracy* in characterizing the conceptualization they commit to. There are two possible ways an ontology can get closer to a conceptualization: by developing a richer axiomatization, and by adopting a richer domain and/or a richer set of relevant conceptual relations. In the first

[12] Not necessarily this formal vocabulary will be part of a logical language: for example, it may be a protocol of communication between agents.

case, the distance between the set of ontology models and the set of intended models is reduced. In the second case, it is possible – at least in principle – to include in the set of relevant conceptual relations (some of) those relations that characterize a world state, extending at the same time the domain in order to include the entities involved by such relations: for instance, in the case of the blocks world, we may consider the spatial location of a block as a relevant conceptual relation, including therefore locations in the domain, and considering a relation like *on(x,y)* as completely derivable from the locations of *x* and *y*. Since every model now carries the information concerning the state of the world it refers to, the underlying conceptualization can be reconstructed from the set of its intended models. In this case, if an ontology is axiomatized in such a way to have exactly the same models, then it would be a "perfect" ontology.

Another possibility to increase the accuracy of an ontology consists of either adopting a modal logic, which allows one to express constraints across worlds, or just reifying worlds as ordinary objects of the domain. Of course, there is a tradeoff between a coarse and a fine-grained ontology committing to the same conceptualization: the latter gets closer to specifying the intended meaning of a vocabulary (and therefore may be used to *establish consensus* about sharing that vocabulary, or a knowledge base which uses that vocabulary), but it may be hard to develop and to reason on, both because the number of axioms and the expressiveness of the language adopted. A coarse ontology, on the other hand, may consist of a minimal set of axioms written in a language of minimal expressivity, to support only a limited set of specific services, intended to be shared among users which *already agree* on the underlying conceptualization. We can distinguish therefore between detailed *reference ontologies* and coarse *shareable ontologies*, or maybe between *off-line* and *on-line ontologies:* the former are only accessed from time to time for reference purposes, while the latter support core system's functionalities.

2.4 The Ontology Integration Problem

As we have seen in the introduction, information integration is a major application area for ontologies. As well known, even if two systems adopt the same vocabulary, there is no guarantee that they can agree on a certain information unless they commit to the same conceptualization. Assuming that each system has its own conceptualization, a necessary condition in order to make an agreement possible is that the intended models of the original conceptualizations overlap (Fig. 2).

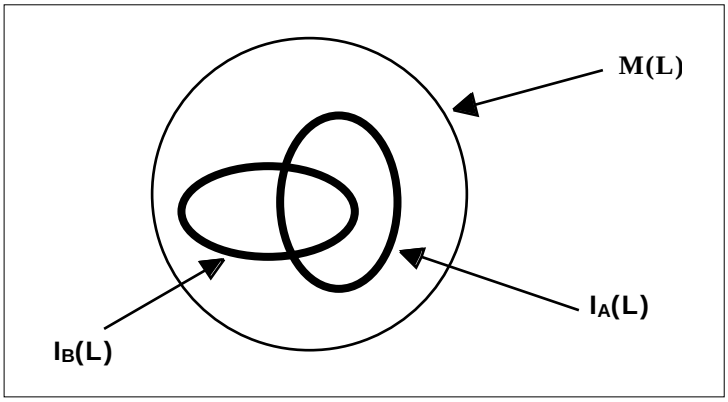

Fig. 2. Two systems **A** and **B** using the same language **L** can communicate only if the set of intended models $\mathbf{I}_A(\mathbf{L})$ and $\mathbf{I}_B(\mathbf{L})$ associated to their conceptualizations overlap.

Supposing now that these two sets of intended models are approximated by two different ontologies, it may be the case that the two ontologies overlap while the intended models do not (Fig. 3). This means that a bottom-up approach to systems integration based on the integration of multiple local ontologies may not work, especially if the local ontologies are only focused on the conceptual relations relevant to a specific *context*, and therefore they are only weak and *ad hoc* approximations of the intended models. Hence, it seems more convenient to agree on a single *top-level* ontology rather than relying on agreements based on the intersection of different ontologies.

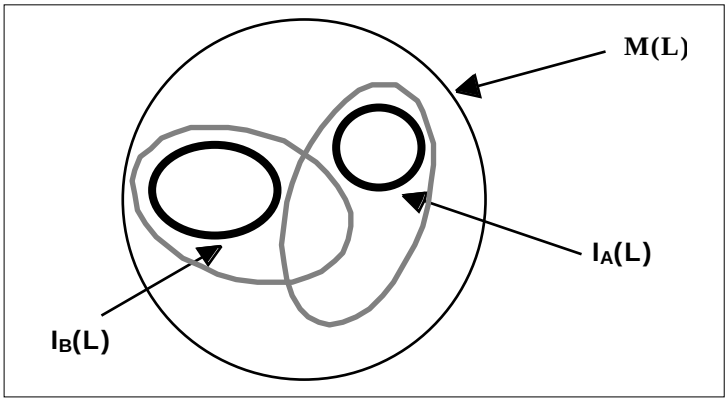

Fig. 3. The sets of models of two different axiomatizations, corresponding to different ontologies, may intersect while the sets of intended models do not.

The considerations above suggest the opportunity to develop different kinds of ontology according to their level of generality, as shown in Fig. 4 below (see [25] for a more detailed discussion).

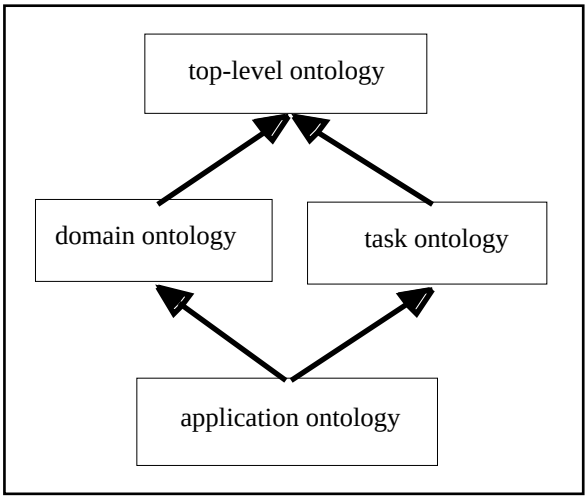

Fig. 4. Kinds of ontologies, according to their level of dependence on a particular task or point of view. Thick arrows represent specialization relationships. From [24].

- *Top-level ontologies* describe very general concepts like space, time, matter, object, event, action, etc., which are independent of a particular problem or domain: it seems therefore reasonable, at least in theory, to have unified top-level ontologies for large communities of users.

- *Domain ontologies* and *task ontologies* describe, respectively, the vocabulary related to a generic domain (like medicine, or automobiles) or a generic task or activity (like diagnosing or selling), by specializing the terms introduced in the top-level ontology.
- *Application ontologies* describe concepts depending both on a particular domain and task, which are often specializations of *both* the related ontologies. These concepts often correspond to *roles* played by domain entities while performing a certain activity, like *replaceable unit* or *spare component*.

As a final consideration, it may be important to make clear the difference between an application ontology and a knowledge base. The answer is related to the purpose of an ontology, which is a *particular* knowledge base, describing facts assumed to be always true by a community of users, in virtue of the agreed-upon meaning of the vocabulary used. A generic knowledge base, instead, may also describe facts and assertions related to a particular state of affairs or a particular epistemic state. Within a generic knowledge base, we can distinguish therefore two components: the ontology (containing state-independent information) and the "core" knowledge base (containing state-dependent information).

3 Towards Ontology-Driven Information Systems

Every (symbolic) information system (IS) has its own ontology, since it ascribes meaning to the symbols used according to a particular view of the world. I shall discuss here the specific, peculiar role an *explicit* ontology can play within an information system, arguing in favor of an architectural perspective where this role is a *central* one, and the ontology profitably "drives" all aspects and all components of an IS, so that we can speak of *ontology-driven information systems.*

An IS consists of components of three different types: application programs, information resources like databases and/or knowledge bases, and user interfaces. These components are integrated in such a way as to accomplish a concrete (business) purpose. When discussing the impact an ontology can have on an IS, we can distinguish two orthogonal dimensions: a temporal dimension, concerning whether an ontology is used at *development time* or at *run time* (i.e., *for* an IS or *within* an IS), and a more structural dimension, concerning the particular way an ontology can affect the main IS components. I shall first discuss some general aspects concerning the first dimension, while analyzing then in more detail the specific impact of ontologies on the IS components, both at development time and at run time.

3.1 The temporal dimension: using ontologies at development time vs. run time

Before focusing on the specific IS components, let us first discuss in general the role that ontologies can have at development time and run time. Note that these two notions are relative to the IS and not to the ontology, which we assume to be a finished product: yet, at the development time, we have to adapt the "finished" ontology at our disposal to the specific requirements of our IS. When the ontology is used by an IS at run time, we speak of an "ontology-driven IS" proper; when it is used at development time, we speak of "ontology-driven IS development".

3.1.1 Using an ontology at development time. In this context, we can distinguish two different scenarios. In the first scenario, we have a set of reusable ontologies at our disposal, organized in an *ontology library* containing domain and task ontologies [49]. In the second

scenario, the degree of reusability is very limited, as we only have a very generic ontology, consisting of coarse domain-level distinctions among the basic entities of the world and meta-level distinctions about kinds of class and kinds of relation.

In the first scenario, the semantic content expressed by the ontology (or ontologies) selected from the ontology library gets transformed and translated into an IS component, reducing the costs of conceptual analysis and assuring – on the assumption of a correct ontology – the *ontological adequacy* of the IS. If the IS to be built is a traditional one, such a content will be just embedded in the standard components; if it is going to be an *ontology-driven* IS, then the result of this development phase will be a separate component, namely an *application ontology* like the one depicted in Fig. 4, which can be seen as a specialization of both a domain ontology and a task ontology. See [25,48] for a discussion of the relationships between an application ontology and an ontology library. An important benefit of using an ontology at development time is that it enables the developer to practice a "higher" level of reuse than is usually the case in software engineering (i.e. knowledge reuse instead of software reuse). Moreover, it enables the developer to reuse and share application domain knowledge using a common vocabulary across heterogeneous software platforms. It also enables the developer to concentrate on the structure on the domain and the task at hand and protects him from being bothered too much by implementation details. Unfortunately, the availability of *off the shelf* ontologies to be used in this way is today extremely limited (see for instance [44] for an example of reuse of an ontology of physical measures). In my opinion, the reason is that these ontologies are not general enough to be effectively specialized for various applications. Moreover, in those cases where an integration of several off-the-shelf ontologies would be necessary, we would encounter the problems discussed in section 2.4.

The second scenario depicted above appears therefore much more realistic. The *quantity* of ontological knowledge available may be modest, but it is its *quality*, i.e. the nature of some fundamental ontological distinctions, which can help the designer in his task of conceptual analysis. In practice, the role of the ontology in this case is not that of a building block going to be adapted and reused, but rather a powerful *tool* – analogous to any other CASE (Computer-Aided Software Engineering) tool – that can increase the quality of the analysis process. In other words, we can think in this case to develop an application ontology with the help of a (relatively small) top-level ontology, without necessarily having an already-developed domain ontology at hand.

An important point to keep in mind in ontology-driven IS development is that an ontology not only can be used to build a new IS (IS engineering), but it can also be used equally profitably for IS *re-engineering*, in order to increase reuse and maintainability. In this way large software investments done in the past can be protected and leveraged. See for instance [38] for a good example of using ontological principles for systems re-engineering.

3.1.2 Using an ontology at run time. We must distinguish here an *ontology-aware* IS from an *ontology-driven* IS: in the first case, an IS component is just aware of the existence of a (possibly remote) ontology and can use it (i.e., query it) for whatever specific application purpose is needed. In the second case, the ontology is just another component (typically local to the IS), cooperating at run time towards the "higher" overall IS goal.

An important reason for using an ontology at run time is enabling the communication between software agents. Software agents are communicating with each other via messages that contain expressions formulated in terms of an ontology (ontology-driven communica-

tion). In order for a software agent to understand the meaning of these expressions, the agent needs access to the ontology they commit to.

3.2 The structural dimension: impact of ontologies on IS components

Each of the components of an IS – application programs, databases, user interfaces – can use an ontology in its own specific way. In the following, we deal with each of these components in turn, investigating the specific role an ontology can play, and distinguishing between development time and runtime aspects.

3.2.1 Using an ontology for the database component. The most obvious use of an ontology is in connection with the database component. In fact, an ontology can be compared with the schema component of a database.

At the development time, an ontology can play an important role in the requirement analysis and conceptual modelling phase, especially if integrated with lexical resources like WordNet [36] in order to support the analysis of natural language informal specifications [29,1,11,47]. The resulting conceptual model can be represented as a computer processable ontology and from there mapped to concrete target platforms. These aspects have been extensively studied in the ESPRIT IDEA project [14], especially for what concerns the mapping of the "knowledge specification" (the ontology) to schemes for many different types of databases (relational, object-oriented, deductive, active). Another example of use of ontologies at development time is information integration: a common conceptual schema to be used for instance in a data warehousing application can be built by (semi-automatically) mapping heterogeneous conceptual schemes on a common top-level ontology (see [7] for a preliminary step in this direction).

At run time, there are many ways in which ontologies and databases can cooperate. The availability of explicit ontologies for information resources (i.e., run-time accessible database schemas) is at the core of the mediation-based approach to information integration [55]. Ontologies can support "intensional queries" regarding the content of a particular database, or dynamic management of queries involving multiple databases.

3.2.2 Using an ontology for the user interface component. Maybe not so obvious, but nevertheless very important, is the use of an ontology in connection with the user interface component. Since ontologies embody semantic information on the constraints imposed on the classes and relationships used to model a given domain and task, they have been successfully used in the Protégé Project[13] to generate form-based interfaces that check for constraints violation.

At run time, the first role an ontology can play within the user interface is to allow itself to be queried and browsed by the user. In this case, the user is aware of the ontology, and uses (i.e. queries) it as part of his normal use of the IS. In this way, the user can browse the ontology in order to better understand the vocabulary used by the IS, being able therefore to formulate queries at the desired level of specificity. If the ontology in question is a sufficiently large one, i.e. a "top-level" linguistic ontology like WordNet, Mikrokosmos [33], or Pangloss [43], then another useful task it can play in the context of a user interface is *vocabulary detaching*: the user is free to adopt his own natural language terms, which are mapped (after a possible disambiguation step) to the IS vocabulary with the help of the ontology. See [28] for an example of this use.

[13] URL: http://smi-web.stanford.edu/projects/prot-nt/documentation/

3.2.3 Using an ontology for the application program component. Application programs are still an important part of many ISs. They usually contain a lot of domain knowledge, which, for various reasons, is not explicitly stored in the database. Some parts of this knowledge are encoded in the static part of the program in the form of type or class declarations, other parts (like for example business rules) are implicitly stored in the (sometimes obscure) procedural part of the program.

At the development time, an IS developer can – in principle – generate the static part of a program with help of an ontology. Moreover, ontologies integrated with linguistic resources can be used to support the development of object-oriented software like we have seen in the case of databases, but I am not aware of any contribution in this field (although some proposals to use ontological principles for object-oriented design have been made [52,39]).

At run time, we may decide to represent explicitly all the domain knowledge implicitly encoded in the application program, turning the program in a knowledge-based system. As well known, this has large benefits from the point of view of ease-of-maintenance, extensibility and flexibility. In this case, the knowledge base could be constituted by a core knowledge base plus an ontology. One may object that there could be good reasons to keep the core, "strategic" knowledge in a non-explicit form, either for security reasons, or because of legacy problems: in this case, however, at least the *ontological commitment* of the application program should be made explicit, in order to facilitate its accessibility, maintainability, and integrability. Ontologies can help therefore to increase the *transparency* of application software.

Conclusions

After many papers mainly devoted to the philosophical foundations of ontology development, I have focused here on the application side, trying to offer a systematic account of the central role ontologies may play in future information systems. The revisitation of the recent definitions concerning the notion of ontology was not planned in advance, but I realized that some further work on this issue was still necessary only while writing this paper with an interdisciplinary audience in mind. Once again, the interdisciplinary perspective has paid off.

The material presented in section 3 has been developed in the framework of a recently started Research Program on Ontology-Driven Information Systems (ODIS), in cooperation with Bert Fitié (AnalyTech Consulting, Den Haag). I am indebted to Claudio Masolo for his helpful comments on an earlier draft of this work.

Bibliography

[1] Ambrosio, A. P., Métais, E., and Meunier, J.-N. 1997. The Linguistic Level: Contribution for Conceptual Design, View Integration, Reuse and Documentation. *Data and Knowledge Engineering,* **21**: 111-129.

[2] Artale, A., Franconi, E., Guarino, N., and Pazzi, L. 1996. Part-Whole Relations in Object-Centered Systems: an Overview. *Data and Knowledge Engineering,* **20**(3): 347-383.

[3] Ashenhurst, R. L. 1996. Ontological Aspects of Information Modelling. *Minds and Machines,* **6**: 287-394.

[4] Barley, M., Clark, P., Williamson, K., and Woods, S. 1997. The Neutral Representation Project. In *Proceedings of AAAI Spring Symposium on Ontological Engineering.* Stanford University, CA, AAAI Press: 1-8.

[5] Bateman, J. A. 1995. On the Relationship Between Ontology Construction and Natural Language: a

Socio-Semiotic View. *International Journal of Human-Computer Studies*, **43**: 929-944.
[6] Benjamins, V. R. and Fensel, D. 1998. The Ontological Engineering Initiative (KA)^2. In N. Guarino (ed.) *Formal Ontology in Information Systems*. IOS Press (this volume).
[7] Bergamaschi, S., Castano, S., De Capitani di Vimercati, S., Montanari, S., and Vincini, M. 1998. An Intelligent Approach to Information Integration. In N. Guarino (ed.) *Formal Ontology in Information Systems*. IOS Press (this volume).
[8] Boley, H. and Guarino, N. 1996. Proceedings of the Workshop on Product Knowledge Sharing for Integrated Enterprises. In M. Wolf and U. Reimer (eds.), *Proceedings of the First International Conference on Practical Aspects of Knowledge Management*. Schweizer Informatiker Gesellschaft, Basel, Switzerland.
[9] Borgo, S., Guarino, N., and Masolo, C. 1997. An Ontological Theory of Physical Objects. In *Proceedings of Qualitative Reasoning 11th International Workshop*. Cortona, Italy, IAN-CNR, Pavia: 223-231.
[10] Borst, W. N., Akkermans, J. M., and Top, J. L. 1997. Engineering Ontologies. *International Journal of Human-Computer Studies*, **46**: 365-406.
[11] Burg, J. F. M. 1997. *Linguistic Instruments in Requirements Engineering*. IOS Press.
[12] Casati, R., Smith, B., and Varzi, A. 1998. Ontological Tools for Geographic Representation. In N. Guarino (ed.) *Formal Ontology in Information Systems*. IOS Press (this volume).
[13] Casati, R. and Varzi, A. 1997. Spatial Entities. In O. Stock (ed.) *Spatial and Temporal Reasoning*. Kluwer, Dordrecht.
[14] Ceri, S. and Fraternali, P. 1997. *Designing Database Applications with Objects and Rules: The IDEA Methodology*. Addison Wesley.
[15] Gaines, B. 1997. Editorial: Using Explicit Ontologies in Knowledge-based System Development. *International Journal of Human-Computer Systems*, **46**: 181.
[16] Gangemi, A., Pisanelli, D., and Steve, G. 1998. Ontology Alignment: Experiences With Medical Terminologies. In N. Guarino (ed.) *Formal Ontology in Information Systems*. IOS Press (this volume).
[17] Genesereth, M. R. and Nilsson, N. J. 1987. *Logical Foundation of Artificial Intelligence*. Morgan Kaufmann, Los Altos, California.
[18] Gómez-Pérez, A. K. 1997. Knowledge Sharing and Reuse. In Liebowitz (ed.) *The Handbook on Expert Systems*. CRC Press.
[19] Gotts, N. M., Gooday, J. M., and Cohn, A. G. 1996. A Connection Based Approach to Commonsense Topological Description and Reasoning. *The Monist: An International Journal of General Philosophical Inquiry*, **79**(1).
[20] Gruber, T. R. 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition*, **5**: 199-220.
[21] Gruber, T. R. 1995. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human and Computer Studies*, **43**(5/6): 907-928.
[22] Gruninger, M. and Fox, M. S. 1995. The Logic of Enterprise Modelling. In J. Brown and D. O' Sullivan (eds.), *Reengineering the Enterprise*. Chapman and Hall.
[23] Guarino, N. 1995. Formal Ontology, Conceptual Analysis and Knowledge Representation. *International Journal of Human and Computer Studies*, **43**(5/6): 625-640.
[24] Guarino, N. 1997. Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. In M. T. Pazienza (ed.) *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*. Springer Verlag: 139-170.
[25] Guarino, N. 1997. Understanding, Building, and Using Ontologies: A Commentary to "Using Explicit Ontologies in KBS Development", by van Heijst, Schreiber, and Wielinga. *International Journal of Human and Computer Studies*(46): 293-310.
[26] Guarino, N., Borgo, S., and Masolo, C. 1997. Logical Modelling of Product Knowledge: Towards a Logical Semantics for STEP. In *Proceedings of European Conference on Product Data Technology (PDT Days 97)*. Sophia Antipolis, France, QMS, Berkshire, UK: 183-190.
[27] Guarino, N. and Giaretta, P. 1995. Ontologies and Knowledge Bases: Towards a Terminological Clarification. In N. Mars (ed.) *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing 1995*. IOS Press, Amsterdam: 25-32.
[28] Guarino, N., Masolo, C., and Vetere, G. 1998. OntoSeek: Using Large Linguistic Ontologies for Gathering Information Resources from the Web. LADSEB-CNR Int. Rep. 02/98, March 1998.
[29] Hoppenbrouwers, J., van der Vos, B., and Hoppenbrouwers, S. 1996. NL Structures and Conceptual Modelling: the KISS case. In R. P. van de Riet, J. F. M. Burg and A. J. van der Vos (eds.), *Applications of Natural Language to Information Systems*. IOS Press.
[30] Knight, K. and Luk, S. 1994. Building a Large Knowledge Base for Machine Translation. In *Proceed-*

ings of American Association of Artificial Intelligence Conference (AAAI-94). Seattle, WA.
[31] Lang, E. 1991. The LILOG Ontology from a Linguistic Point of View. In O. Herzog and C. R. Rollinger (eds.), *Text Understanding in LILOG*. Springer-Verlag, Berlin.
[32] Lehmann, F. 1995. Machine-Negotiated, Ontology-Based EDI (Electronic Data Interchange). In *Proceedings of CIKM-94 Workshop on Electronic Commerce*, Springer Verlag.
[33] Mahesh, K. 1996. Ontology Development for Machine Translation: Ideology and Methodology. New Mexico State University, Computing Research Laboratory MCCS-96-292.
[34] McGuinness, D. 1998. Ontological Issues for Knowledge-Enhanced Search. In N. Guarino (ed.) *Formal Ontology in Information Systems*. IOS Press (this volume).
[35] Mena, E., Kashyap, V., Illarramendi, A., and Sheth, A. 1998. Domain Specific Ontologies for Semantic Information Brokering on the Global Information Infrastructure. In N. Guarino (ed.) *Formal Ontology in Information Systems*. IOS Press (this volume).
[36] Miller, G. A. 1995. WORDNET: A Lexical Database for English. *Communications of ACM*(11): 39-41.
[37] Nilsson, N. 1991. Logic and Artificial Intelligence. *Journal of Artificial Intelligence*: 31-55.
[38] Partridge, C. 1996. *Business Objects: Re-Engineering for Reuse.* Butterworth-Heinemann, Oxford.
[39] Pazzi, L. 1998. Three Points of View in the Characterization of Complex Entities. In N. Guarino (ed.) *Formal Ontology in Information Systems*. IOS Press (this volume).
[40] Poli, R. 1996. Ontology and Knowledge Organization. In *Proceedings of 4th Conference of the International Society of Knowledge Organization (ISKO 96).* Washington.
[41] Smith, B. 1998. Basic Tools of Formal Ontology. In N. Guarino (ed.) *Formal Ontology in Information Systems*. IOS Press (this volume).
[42] Sowa, J. F. 1998. *Knowledge Representation: Logical, Philosophical, and Computational Foundations.* PWS Publishing Co. (forthcoming), Boston.
[43] Swartout, B., Patil, R., Knight, K., and Russ, T. 1996. Toward distributed use of large-scale ontologies. In *Proceedings of 10th Knowledge Acquisition for Knowledge-Based Systems Worskhop*. Banff, Canada.
[44] Uschold, M., Clark, P., Healy, M., Williamson, K., and Woods, S. 1998. Ontology Reuse and Application. In N. Guarino (ed.) *Formal Ontology in Information Systems*. IOS Press (this volume).
[45] Uschold, M. and Gruninger, M. 1996. Ontologies: Principles, Methods and Applications. *The Knowledge Engineering Review*, **11**(2): 93-136.
[46] Uschold, M., King, M., Moralee, S., and Zorgios, Y. 1998. The Enterprise Ontology. *The Knowledge Engineering Review (to appear).*
[47] Van de Riet, R., Burg, H., and Dehne, F. 1998. Linguistic Issues in Information Systems Design. In N. Guarino (ed.) *Formal Ontology in Information Systems*. IOS Press (this volume).
[48] Van Heijst, G., Schreiber, A. T., and Wielinga, B. J. 1997. Roles aren't Classes: a Reply to Nicola Guarino. *International Journal of Human and Computer Studies*, **46**: 311-318.
[49] Van Heijst, G., Schreiber, A. T., and Wielinga, B. J. 1997. Using Explicit Ontologies in KBS Development. *International Journal of Human and Computer Studies*, **46**: 183-292.
[50] Varzi, A. 1998. Basic Problems of Mereotopology. In N. Guarino (ed.) *Formal Ontology in Information Systems*. IOS Press (this volume).
[51] Wand, Y. 1989. A Proposal for a Formal Model of Objects. In W. Kim and F. H. Lochovsky (eds.), *Object-Oriented Concepts, Databases, and Applications.* Addison Wesley, Reading, MA: 537-559.
[52] Wand, Y. and Weber, R. 1990. An Ontological Model of an Information System. *IEEE Transactions On Software Engineering*, **16**(11): 1282-1292.
[53] Weber, R. 1997. *Ontological Foundations of Information Systems.* Coopers and Lybrand.
[54] Welty, C. 1998. The Ontological Nature of Subject Taxonomies. In N. Guarino (ed.) *Formal Ontology in Information Systems*. IOS Press (this volume).
[55] Wiederhold, G. (ed.) 1996. *Intelligent Integration of Information*. Kluwer Academic Publishers, Boston, MA.