The KACTUS View on the 'O' Word *

Guus Schreiber Bob Wielinga Wouter Jansweijer

University of Amsterdam, Social Science Informatics
Roetersstraat 15, NL-1018 WB Amsterdam, The Netherlands
Tel: +31 20 525 6789; Fax: +31 20 525 6896
E-mail: schreiber@swi.psy.uva.nl; URL: http://www.swi.psy.uva.nl

Abstract

This paper describes the view taking in the ESPRIT project "KACTUS" on the notion of ontology. KACTUS is an application-driven project concerned with knowledge reuse in technical doamins. This paper discusses part of the theoretical background of the project. The use of the definitions of "ontology" and related terms is illustrated through an example in the VT elevator-design domain.

1 Introduction

The term "ontology" is currently fashionable in the context of knowledge sharing and reuse. However, no agreement exists on what an ontology is or how it can be used (Guarino & Giaretta, 1995). In this paper we state the view taken by the European ESPRIT project KACTUS on what an ontology is and we illustrate through examples one particular way of using ontologies. The KACTUS project aims at the development of methods and tools for the reuse of knowledge about technical systems during their life-cycle. The project is application-driven: systems are being developed in the domains of preliminary-ship design, oil-production processes, and electrical networks. The paper is organized as follows. In Sec. 2 we define "ontology" and related terms as perceived in KACTUS. Sec. 3 describes the elevator design (VT) case to illustrate one particular use of an ontology for the construction of a knowledge base suitable for performing a configuration task. Sec. 4 summarizes the main conclusions.

2 The Notion of Ontology

2.1 Ontologies and domain theories Gruber (1993) defines an ontology as an "explicit specification of a conceptualization". A conceptualization is a set of definitions that allows one to construct expressions about some application domain. Although this definition has intuitive appeal it leaves a lot unsaid about what exactly a "conceptualisation" is. Guarino and Giaretta clarify

*The research reported here was carried out in the course of the KACTUS project. This project is partially funded by the ESPRIT Programme of the Commission of the European Communities as project number 8145. The partners in the KACTUS project are Cap Gemini Innovation (France), LABEIN (Spain), Lloyd's Register (United Kingdom), STATOIL (Norway), Cap Programmator (Sweden), University of Amsterdam (The Netherlands), University of Karlsruhe (Germany), IBERDROLA (Spain), DELOS (Italy), FINCANTIERI (Italy) and SINTEF (Norway). This paper reflects the opinions of the authors and not necessarily those of the the consortium.

this notion in a recent paper and conclude that one sense of the word ontology is "a logical theory which gives an explicit, partial account of a conceptualization" (Guarino & Giaretta, 1995). In their view a conceptualization accounts for the intended meaning of the terms used in a particular knowledge base. Since the ontologies that are of practical interest for the knowledge engineering community, only partially specify the intended semantics, Gruber's definition should be interpreted as a "partial specification".

The view that we illustrate in this paper is largely compatible with the definition proposed by Guarino and Giaretta with two differences. First, we view the relation between a domain theory and an ontology as an object-meta relation.[1] An ontology is formulated as a *meta-level theory representing a certain viewpoint on a set of possible* domain theories. A domain theory contains a set of expressions that represents a model of some application domain. A second difference is that we do not restrict ontologies to account for "intended semantics" only. An ontology may introduce additional meaning beyond the meaning intended by the creator of a domain theory.

In this section we illustrate this definition of "ontology" through a number of simple examples. Section 3 gives some real-life examples. In Fig. 1 a domain theory is depicted with two sample expressions. The use of a mathematical language in the example is arbitrary: any language would have been fine. The domain theory has some application domain as its referent, which is characterized through an interpretation context[2] (e.g. "task-type = design").

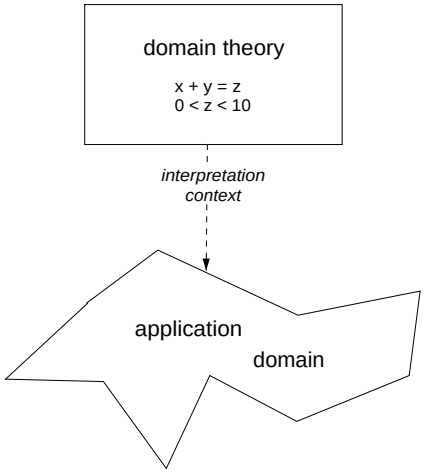

FIGURE 1: Domain theory as a model of some part of an application domain

The reason for defining an ontology as a meta-level viewpoint on a domain theory is that it allows many-to-many relations between ontologies and domain theories. We can use one ontology to describe a spectrum of domain theories, each having its own representation. Similarly, several ontologies can express a viewpoint on one single domain theory, thus allowing for multiple interpretations (and reuse) of the same theory.

Fig. 2 depicts a sample ontology. This particular ontology contains two knowledge-type definitions, namely `parameter` and `constraint-expression`, which together constitute the conceptualization introduced by this ontology. This conceptualization is expressed as a meta-level viewpoint on the domain theory of Fig. 1: the mathematical formulae are interpreted as

[1] we use the term domain theory for that part of the knowledge base that represents knowledge about some state of affairs in the world.

[2] We use the term "interpretation context" in a similar spirit as Shahar (1994).

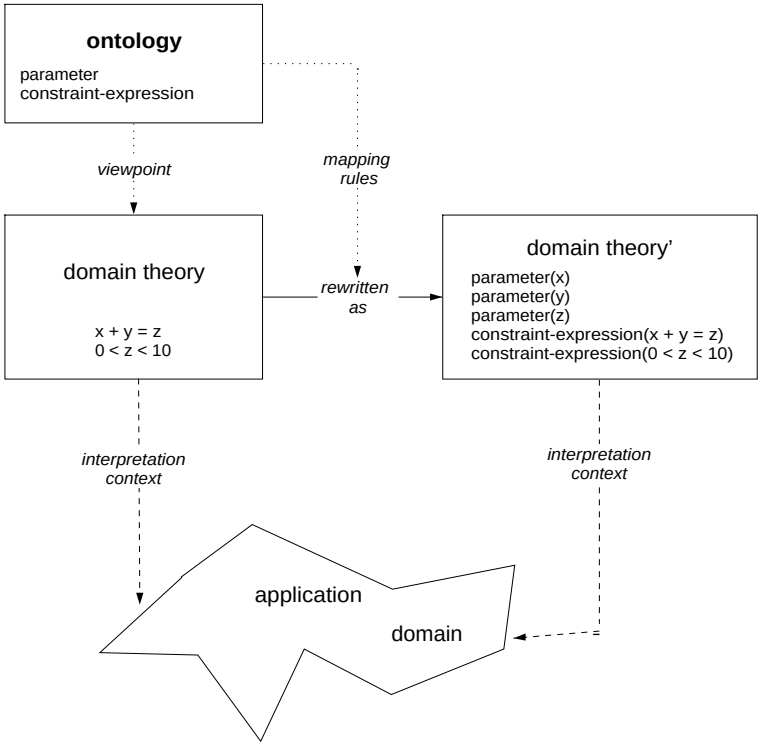

FIGURE 2: Ontology as a meta-model that describes a domain theory

`constraint expressions`; the variables in the formulae as `parameters`. Only one domain theory is shown in Fig. 2 on which the ontology is defined as a viewpoint. In practice, there can be many different theories, for which this viewpoint is expressed. A viewpoint on a particular domain theory can be operationalized in terms of a set of mapping rules that rewrites a domain theory into a form dictated by the ontology (see `domain-theory'` in Fig. 2). The latter theory can be seen as a direct "instantiation" of the ontology, where the relation between ontology and theory is trivialized and not really "meta-level" any more. Builders of knowledge-based systems might say at this point: why didn't you express the domain theory directly in the "instance" format? However, this is only possible if the domain theory is constructed from scratch each time an application is built. Typically, in reusing domain theories we need to do at least some rewriting and/or reinterpretation. An example of a minimal mapping would be to rewrite a domain theory in KIF ground facts into a Prolog-like formalism.

Thus, the KACTUS view on ontology can be summarized in the following definition:

> An ontology is an explicit, partial specification of a conceptualization that is expressible as a meta-level viewpoint on a set of possible domain theories for the purpose of modular design, redesign and reuse of knowledge-intensive system components.

The last part of the definition is included to stress that we view ontologies and domain theories as *engineering* products. Ontological engineering is a discipline that enables modular design, redesign and reuse of knowledge-intensive systems and their components. There is no requirement that ontologies and domain theories represent faithful descriptions of some part of the real world. This implies that the notion of ontology as used in KACTUS is different from the

way the term is used in philosophy. Thus, although expressions in a domain theory are assumed to have a referent in some real or artificial application domain, no formal interpretation function is required. Instead, we enable a characterization of the referent through the specification of an *interpretation context*. This interpretation context typically consists of application domain indices such as task-type, method-type and domain-type.

2.2 Ontology libraries The ontology in Fig. 2 constitutes a simple conceptualization. In most realistic applications, the ontologies are much more complex. In Fig. 3 both the domain theory and the ontology contains additional complexity: components are introduced, and the parameters are related to these components.

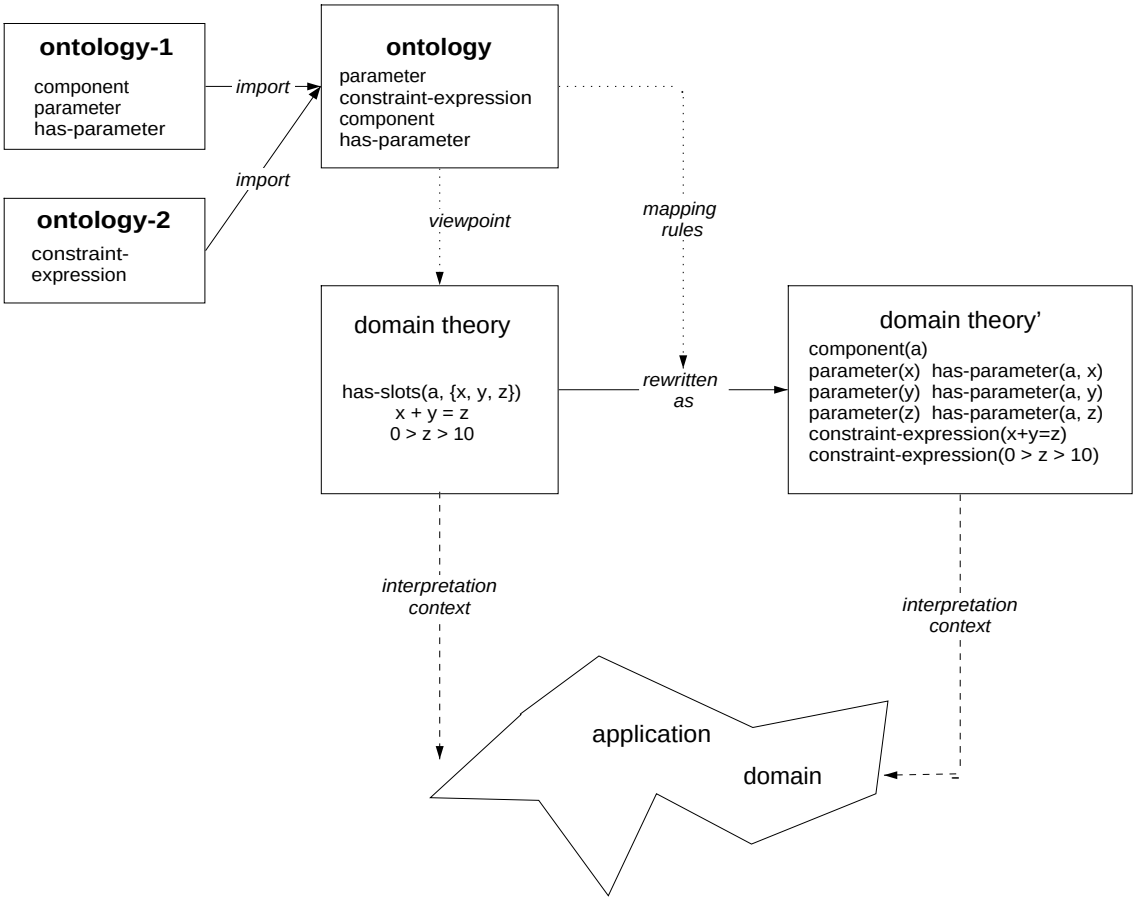

FIGURE 3: Import as an example of compositionality of ontologies

When the ontology becomes more complex, the need arises to modularize ontologies. One key idea behind modularization of ontologies is that the overall ontology of an application (which, not surprisingly, is called "application ontology" in the literature (Gennari *et al.*, 1994)) is at least partially constructed from of a library of small-scale ontologies. Although a promising approach, this also gives rise to a number of research topics that need to be addressed:

- What kind of relations need to be possible between ontological theories?

For example, ONTOLINGUA and CML currently only support "import". EXPRESS[3] also supports "rename" and "export". Other operations have to be considered as well (e.g. "parametrisation").

- How should a library of ontologies be structured?

  Case studies on this issue have been performed (cf. the ONTOLINGUA library and the GAMES-II medical library (van Heijst *et al.*, 1995)).

- What are the primitive ontological categories in a library?

  Research in formal ontologies and in philosophy in general has focused on finding universal ontological categories. Although the results provide an interesting source of ideas, these efforts clearly have not produced a single top-level categorization. A more pragmatic approach is to develop such shared definitions for a class of application domains (e.g. medicine, ships, electrical networks, oil platforms, ...).

- How can one cope with alternative and/or conflicting definitions in ontologies within a library?

  There are multiple ways to conceptualize the world. Assuming that a library will contain ontologies that have proven useful in practice, libraries need to cope with situations where multiple, possibly conflicting, definitions are available for the same construct.

- How can we support the selection of relevant ontological theories from the library?

  To this end we need some sort of indexing schema of ontologies. An indexing scheme that has been proposed in previous publications (Wielinga & Schreiber, 1994) is to characterize the interpretation context in which an ontology can be used through three dimensions:

  - Task type: e.g. diagnosis, prediction, assessment, design, planning, etc.
  - Problem-solving method: e.g. propose-and-revise, skeletal planning, abductive diagnosis.
  - Domain-model type: e.g. structural model, functional model, behavioral model, causal model.
  - Domain-type: e.g. oil production platform, mid-ships, electrical networks, elevators.

2.3 Leveling of Ontologies Ontologies can also have a recursive structure, meaning that an ontology expresses a viewpoint on another ontology. Such a viewpoint entails a reformulation and/or reinterpretation of the other ontology. The VT case study (see the next section) gives a good example of such a leveling of ontologies, and shows how this can support reuse. This layered organization is depicted graphically in Fig. 4. The figure shows a second ontology that expresses a viewpoint on part of the ontology shown previously. The knowledge type `constraint-expression` is split into two sub-types, namely `calculation` and `constraint`. This viewpoint can be operationalized through a second set of mapping rules that enables a partial rewrite of `domain-theory'` to `domain-theory''`. The latter theory has a different

[3] It is the explicit purpose of the KACTUS to be able to support multiple ontology specification languages. Currently, the project is investigating an integrated use of ONTOLINGUA, CML (the CommonKADS expertise model notation) and EXPRESS (a language used for sharing CAD/CAM data).

interpretation context. In this way the closely-related terms `constraint-expression` and `constraint` are associated with different context-specific semantics.

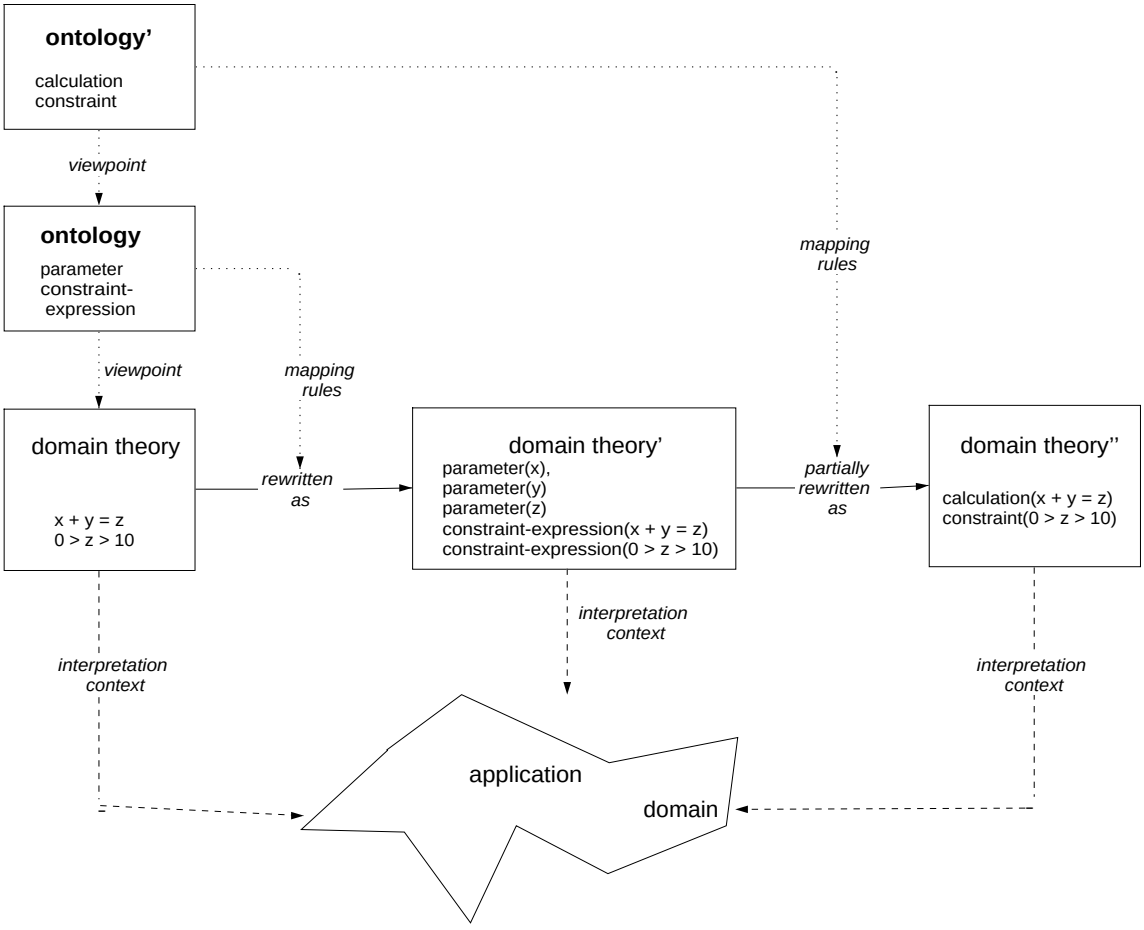

FIGURE 4: Leveling of ontologies: one ontology as a meta-model of another ontology

This multi-level organization raises research questions such as the required expressivity of the mapping formalisms for expressing viewpoints between ontologies. At least two different mapping operations can be identified. The first one is the mapping of the vocabulary of one ontology onto the vocabulary of the other ontology without a change in the semantics of the expressions. The second one entails a change in the semantics of the ontology.

The first type of mapping will occur frequently. The vocabulary used by ontologies can be different while conveying the same meaning (*i.e.* `boat` in one ontology can be mapped on `ship` in another ontology if they refer to the same type of objects in the universe of discourse). In this case the mapping formalism expresses an identity relation. This type of mappings is important for the rewriting of a knowledge base in one formalism (and described by one ontology) in another representional formalism (described by another ontology). The mappings have the effect of representation mappings.

The second type of mapping occurs when we provide an interpretation of an underlying ontology. Sometimes this interpretation can be expressed in an identity relation as well. An example is the mapping of the concept `variable` in a `mathematical-formula` ontology on

the concept `parameter` in a `state` ontology. A slightly different case of this mapping occurs when we identify *roles* for ontology expressions in a problem solving environment. A problem-solving task has an ontology of it's input and output roles which needs to be mapped onto the ontology that describes the domain of the application (for instance, the mapping of the task-notion of `hypothesis` onto a domain notion of `state`).

Often, however, this type of mapping provides a more specific interpretation of an ontology and introduces additional commitments. This is what happened in the mapping examples presented in Fig. 4 where the "=" symbol is interpreted as an assignment operator. Such a change in the semantics of expressions in an ontology also occurs when we subsume (part of) a domain specific ontology under a high level ontology (*e.g.* by connecting an ontology about an electrical network with a high-level ontology about `components` and `connecting-components`) from a library). We will need an expressive mappings formalism for this semantical type of mappings.

3 Example of Ontology Use: The VT Case

Background The V(ertical)T(ransportation) problem is concerned with the the routine design of elevators. The VT example has been used by the knowledge engineering community as a testbed in the so-called Sisyphus experiment (Schreiber & Birmingham, 1994). In this context an Ontolingua ontology and domain theory[4] were developed for this domain. The description of the VT example in this section represents a simplification of the description given in the COMMONKADS Sisyphus contribution (Schreiber & Terpstra, 1995).

3.1 VT ontologies We distinguished two ontologies that describe the domain knowledge required for solving the VT parametric design problem: the *parametric-design* ontology and the propose-and-revise ontology.

Parametric-design ontology This ontology describes a number of basic conceptualizations that are typical for parametric-design problems. The three central constructs introduced by this ontology are:

1. `component`: a part of the artefact to be designed, e.g. "platform", "counter-weight". The term `component-model` is used to refer to a particular component type (e.g. "platform-type 4B").
2. `parameter-slot`: a characteristic of a `component` to which a value is assigned during the design process. An example would be the weight or the length of a component.
3. `constraint-expression`: this term refers to formulae that describe (logical/numerical) dependencies between parameter-slot values.

This ontology is used in VT as a so-called "task-type oriented" ontology: the ontology describes the general ontological commitments required in the context of the parametric-design task. It is assumed not to be biased towards a particular *method* for solving a parametric design problem.

Propose-and-Revise ontology Propose-&-Revise (P&R) (Marcus *et al.*, 1988) is a method used by the VT system to solve the parametric-design problem. The P&R ontology can be seen as a *method-oriented* ontology describing ontological commitments specific for the method selected to solve a parametric design task. The P&R ontology is thus more specific (application-dependent) than the parametric-design ontology.

[4] http://www-ksl.stanford.edu/knowledge-sharing/README.html

It turns out that the P&R ontology can partially be described as a meta-level viewpoint on the parametric design ontology. The main constructs in this part of the P&R ontology are:

Parameters P&R assumes that a design is represented as a flat set of parameters. This means that components for which during design a component model has to be selected need to be reformulated as parameters. Parameter slots are also represented as parameters.

Calculations and constraints The P&R method partitions the set of constraint expressions into two sub-sets that each play a different role in the reasoning process: `calculations` and `constraints`. Calculations contain a `mathematical-expression`: a formula that produces a value for a parameter. Constraints contain a `logical-expression` that evaluates to either true or false.

The two sub-sets correspond to different *roles* that these domain-knowledge fragments play in the problem-solving process dictated by P&R. Calculations are only used to derive design extensions. The constraints are used to verify the design.

In this way, the P&R ontology defines its own interpretation of terms. For example, the term "constraint" in the P&R ontology has a much more restricted meaning than "constraint expression" in the parametric-design ontology. The ontological levels can be seen as *attributing context-specific semantics* to domain knowledge elements. This is in contrast with the traditional logicist's view of model-theoretic semantics, which implies a description of semantics at one level.

3.2 Sharing the VT domain theory In the situation where a KBS is built from scratch it is possible to define one ontology, and view the actual knowledge base as a pure *instantiation* of that ontology. In the light of efforts to share and/or reuse knowledge bases and ontologies, this approach turns out to be insufficient. For example, in the VT domain there was an existing domain theory with its own ontology, specified in ONTOLINGUA. We were able to rewrite the domain expressions in this ontology by a set of mapping rules into statements of the types defined in the parametric-design ontology.

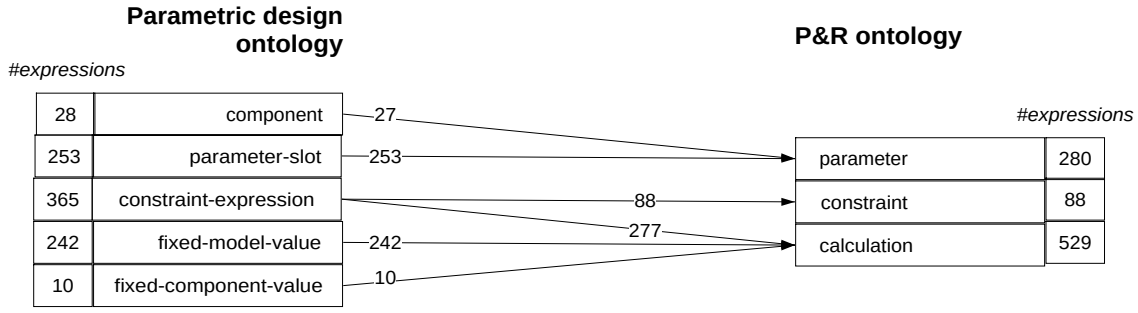

FIGURE 5: Overview of the transformation of expressions in the parametric-design ontology to the P&R ontology. The figures indicate the number of expressions involved. It can be seen that constraint expressions are split into two sub-sets. Only 88 constraint expressions are used as real "constraints". Components are transformed into parameters, if they have associated component models (this is true for all components but one). See (Schreiber & Terpstra, 1995) for more details about the transformation process

Subsequently we have specified a set of ontology mappings that define how the P&R viewpoint on the parametric design ontology can be realized. The mapping is a partial one: not all constructs

in the parametric design ontology are mapped onto constructs in the P&R ontology. The mappings were used in the VT implementation to reuse the VT domain theory within our application. Fig. 5 shows the most important results of the transformation of expressions in the parametric-design ontology to expressions in the P&R ontology. It shows, for instance, that in the P&R ontology only a subset of the domain costructs in the parametric-design ontology are "real" constraints; the others are used to compute values. The details of this transformation process are described elsewhere (Schreiber *et al.*, 1994; Schreiber & Terpstra, 1995).

4 Conclusions

In this paper we have developed the notion of ontology as a meta-level theory about a set of domain knowledge statements. This view on ontologies is in many respects compatible with the recent definition given in (Guarino & Giaretta, 1995). The advantage of choosing the meta-level for defining the ontology is that the ontology itself can be formulated as a first-order theory, rather than as a modal theory as is done in (Guarino *et al.*, 1994). The mapping rules between the object and meta-level terms provide the second order aspects of the ontology. Our approach does not commit to a particular semantic framework in which an ontology is constructed. Rather we propose that the ontology can not only represent aspects of the intended semantics of the domain theory, but that new semantics can be added to a domain theory through ontology mappings. Such additional meaning may be necessary to provide a problem solver with the required knowledge to solve a problem.

The VT example showed how ontology mappings can be applied in sharing knowledge bases. The Ontolingua domain theory, built at another site, on a different platform, using a different representation, could be used within our framework to access the knowledge types defined in the task-type oriented ontology. In the VT case approx. 90% of the total amount of knowledge required for the application could be reused in this way. It actually meant a reduction of a number of weeks with respect to the effort of building the application.[5]

The use of ontologies is not restricted to the construction of task-oriented domain theories. Many operations on domain theories can be performed using the notion of reformulation through ontologies. For example general concepts from a library of general ontologies can be combined to create ontologies and knowledge bases for specific application domains. New interpretations of a data or knowledge base can be generated using the techniques described in this paper. This way of reformulation and reinterpretation of existing knowledge will-we believe- be an important tool for knowledge sharing and reuse.

Acknowledgements

This paper has been extracted from the KACTUS project deliverable DO1b.1 "Framework and Formalism for Expressing Ontologies". It is the result of a series of discussions to which many participants in KACTUS contributed. Helpful comments were received from Anjo Anjewierden, Philippe Gobinet, Frank van Harmelen, Gertjan van Heijst, Rob Martil and Erling Woods. Peter Terpstra contributed to the devlopment of the VT example.

[5] Even taken into account that various bugs were found and repaired in the VT knowledge base.

References

GENNARI, J., TU, S., ROTENFLUH, T., & MUSEN, M. (1994). Mapping domains to methods in support of reuse. *International Journal of Human-Computer Studies*, 41:399–424.

GRUBER, T. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220.

GUARINO, N., CARRARA, M., & GIARETTA, P. (1994). Formalizing ontological commitments. In *Proceedings AAAI'94, Seattle*. Morgan Kaufmann.

GUARINO, N. & GIARETTA, P. (1995). Ontologies and knowledge bases: Towards a terminological clarification. Paper to be presented at the KB&KS'95 Conference, University of Tente, The Netherlnads.

MARCUS, S., STOUT, J., & MCDERMOTT, J. (1988). VT: An expert elevator designer that uses knowledge-based backtracking. *AI Magazine*, Spring:95–111.

SCHREIBER, A. T. & BIRMINGHAM, W. P., editors (1994). *Proceedings of the 8th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop. Volume 3: Sisyphus II – VT Elevator Design Problem*, Calgary, Alberta, Canada. SRDG Publications, University of Calgary.

SCHREIBER, A. T. & TERPSTRA, P. (1995). Sisyohus-VT: A CommonKADS solution. Technical report, esprit project 8145 kactus, University of Amsterdam. Submitted for publication.

SCHREIBER, A. T., TERPSTRA, P., MAGNI, P., & VAN VELZEN, M. (1994). Analysing and implementing VT using COMMON-KADS. In Schreiber, A. T. & Birmingham, W. P., editors, *Proceedings of the 8th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop. Volume 3: Sisyphus II – VT Elevator Design Problem*, pages 44–1 – 44–29, Alberta, Canada. SRDG Publications, University of Calgary.

SHAHAR, Y. (1994). *A Knowledge-Based Method for Temporal Abstraction of Clinical Data*. PhD thesis, Stanford University, Medical Information Sciences.

VAN HEIJST, G., FALASCONI, S., ABU-HANNA, A., SCHREIBER, A. T., & STEFANELLI, M. (1995). A case study in ontology library construction. *Artificual Intelligence in Medicine*. Forthcoming.

WIELINGA, B. J. & SCHREIBER, A. T. (1994). Conceptual modelling of large reusable knowledge bases. In von Luck, K. & Marburger, H., editors, *Management and Processing of Complex Data Structures*, volume 777 of *Lecture Notes in Computer Science*, pages 181–200, Berlin, Germany. Springer Verlag.