The Journal of Systems and Software 84 (2011) 301–313

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Bridging metamodels and ontologies in software engineering

B. Henderson-Sellers *

School of Software, Faculty of Engineering and Information Technology, University of Technology, Sydney, P.O. Box 123, Broadway, NSW 2007, Australia

ARTICLE INFO

Article history:
Received 27 May 2010
Received in revised form 27 May 2010
Accepted 13 October 2010
Available online 27 October 2010

Keywords:
Ontology
Metamodel
Software engineering

ABSTRACT

Over the last several years, metamodels and ontologies have been developed in parallel isolation. Ontological thinking, largely from the research field of artificial intelligence, has been increasingly investigated by software engineering researchers, more familiar with the idea of a metamodel. Here, we investigate the literature on both metamodelling and ontologies in order to identify ways in which they can be made compatible and linked in such a way as to benefit both communities and create a contribution to a coherent underpinning theory for software engineering. Analysis of a large number of theoretical and semi-theoretical approaches using as a framework a multi-level modelling construct identifies strengths, weaknesses, incompatibilities and inconsistencies within the extant literature. A metamodel deals with conceptual definitions while an ontology deals with real-world descriptors of business entities and is thus better named "domain ontology". A specific kind of ontology (foundational or high-level) provides "metalevel" concepts for the domain ontologies. In other words, a foundational ontology may be used at the same abstraction level as a metamodel and a domain ontology at the same abstraction level as a (design) model, with each pair linked via an appropriate semantic mapping.

1. Introduction

Over the last decade there has been an increasing interest in creating a more rigorous underpinning to various aspects of software engineering (SE). This has led to explorations of, for instance, the role of category theory in object technology (Whitmire, 1997), and the creation of modelling languages such as the Unified Modeling Language (UML) (OMG, 1999) and the OPEN[1] Modelling Language (OML) (Firesmith et al., 1997). In the latter case, the underpinning formality is that of metamodelling (see, for instance, Gonzalez-Perez and Henderson-Sellers, 2008) complemented, in the case of UML, by a logic-based language, the Object Constraint Language (OCL: Warmer and Kleppe, 1999). More recently, and in parallel to the growth of metamodelling, interest has grown in the use of ontologies (Uschold, 2005). Both metamodels and ontologies appear to have a similar aim and this paper provides an analysis of these two approaches in order to discover their similarities and differences and to provide a bridge between them. The context here is on providing a solid basis in order to optimize the benefit from the use of ontologies in software engineering – particularly with respect to multi-level frameworks commonly used in SE, i.e. the application of ontological research to SE – we do not, however, attempt a theoretical assessment of all ontological research.

To fully explore the potential use of metamodels and ontologies in software engineering, a broad understanding of the subtle overlaps between the views of the various communities working on metamodels and on ontologies is both instructive and vital. While software engineering metamodels have played a large part in standards communities such as that of the Object Management Group (OMG), much of our understanding of ontologies has been derived from the AI (artificial intelligence) community. This leads to the need to evaluate the relationship between ontologies and a metamodelling hierarchy such as that of the OMG (2005a) or the International Organization for Standardization (ISO) (ISO/IEC, 2007). Indeed, Favre et al. (2007) note the lack of a general, systematic technique to map between metamodels and ontologies and that is the topic of this paper. We seek to establish a formal relationship between metamodels and ontologies in order that the adoption and integration of ontological thinking and theory into software engineering will result in theoretically sound software development methodologies that are also practical for industry usage.

In Section 2, we describe the features of metamodels, primarily from the software engineering literature. In Section 3, we identify current ontological thinking from literature with a software engineering bias and in Section 4 we combine the two "paradigms" in order to identify links between these two current areas of interest to software engineers. Section 5 outlines related work other than

* Tel.: +61 2 9514 1687; fax: +61 2 9514 4535.
E-mail address: brian.henderson-sellers@uts.edu.au

[1] OPEN is an acronym for Object-oriented Process, Environment and Notation (Henderson-Sellers and Graham, 1996).

0164-1212/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2010.10.025

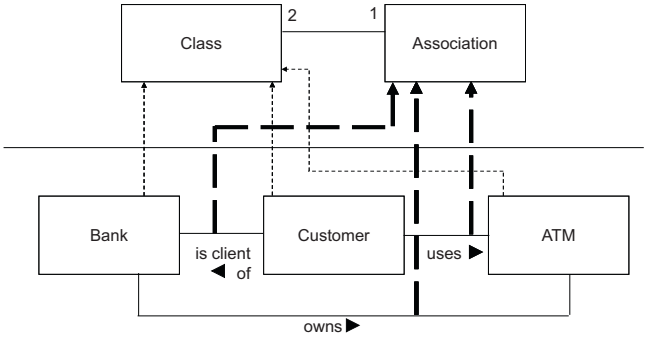

Fig. 1. An example of part of a metamodel with classes Class and Association together with a simple banking domain model that conforms to this metamodel. Rectangles represent concepts, solid lines represent association links and the connections indicating the conformances are dashed. (Heavier lines are used for the associations simply to aid visual discrimination.)

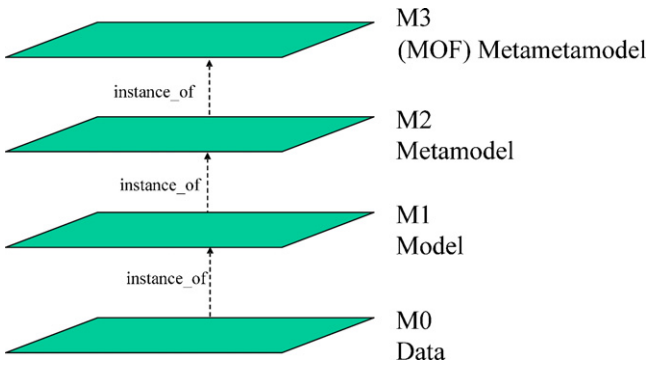

Fig. 2. The four level hierarchy of the OMG, itself based on ANSI (1989) (after Henderson-Sellers & Unhelkar, 2000)© Pearson Education Limited.

that used in earlier sections; while Section 6 draws the discussion to a close with suggestions for future research.

2. Models and metamodels

A model is an abstraction that represents some view on reality, necessarily omitting details, and for a specific purpose—for example, Guizzardi (2005) defines it formally as "A model is an abstraction of reality according to a certain conceptualization". OMG's Model-Driven Architecture (MDA) guide (OMG, 2003) states that "A model of a system is a description or specification of that system and its environment for some specific purpose" and Kühne (2006a) defines a model as "an abstraction of a (real or language-based) system allowing predictions or inferences to be made" – although Hesse (2006) and Kühne (2006b) recommend replacing the word "system" in this definition – by "original (Hesse, 2006) or "subject" Kühne (2006b). A model may be used to document existing situations (descriptive mode) or to describe situations that have yet to eventuate (prescriptive mode) (e.g. Bézivin, 2005a; Kühne, 2006a; Ludewig, 2003; Seidewitz, 2003; Gonzalez-Perez and Henderson-Sellers, 2007). Furthermore, a model refers to a specific domain of discourse and usually adheres to the closed world assumption (which states that anything that has not been specified is disallowed (Aßmann et al., 2006)). Karagiannis et al. (2008) differentiate between iconic (non-linguistic) and linguistic models, arguing that the latter are more common in computing. A detailed discussion of the underpinning theory of models is given in Gonzalez-Perez and Henderson-Sellers (2007) and is not repeated here—see also Muller et al. (2009) and Jackson (2009) who discuss not only models but also the act of modelling.

A metamodel is also a model. It is generally defined as a "model of models" (Flatscher, 2002; OMG, 2003) or, equally, "a model of a set of models" (Favre, 2004) or perhaps "a [prescriptive] model of a modelling language" (Seidewitz, 2003; Favre, 2004). In other words, it is both a model and a modelling language (Gonzalez-Perez and Henderson-Sellers, 2008, p. 18; Bertoa and Vallecillo, 2010). There is a one-to-many relationship between any element in a metamodel to corresponding elements in the model—often describing a homomorphism relationship (Gonzalez-Perez and Henderson-Sellers, 2007). A model is said to conform to its metamodel when each element in the model maps to a corresponding and definitional element in the metamodel. For example, if Class exists in the metamodel, then a Bank class, a Customer class and an ATM class can all exist in the model space. It can then be said that each of these three classes (Bank, Customer and ATM) map to the Class concept in the metamodel (Fig. 1) and thus the model is said to conform to the metamodel. Bézivin (2005a) notes that this conformance relationship has some similarity to the standard object-oriented relationship of instantiation. However, instantiation is only relevant between a class and an object (as an instance of the class) whereas here both model elements and the corresponding metamodel elements are all classes. The importance of using conformance for model-metamodel links rather than instantiation is also stressed by Gašević et al. (2007).

In metamodelling in current SE, one of two possible stacked architectures is commonly used (Figs. 2 and 3). It is worth noting that the OMG architecture in Fig. 2 is based on strict metamodelling (Atkinson, 1997, 1998) wherein the only inter-level relationship permitted is said to be "instance of" (but see comment above re "conformance"). In OMG standards, an M0 object is said to be an instance of a class in level M1; a class in level M1 is said to be an instance of a metaclass in level M2 (e.g. Corcho et al., 2006); and so on. However, since entities in level M2 are actually classes, the instantiation relationship read in reverse implies that M1-level entities must be objects, i.e. they are classes in relation to level M0 but objects in relation to level M2. This paradox is explored in detail in Gonzalez-Perez and Henderson-Sellers (2007) where it is suggested that a better model is to consider M1-level entities as clabjects as defined by Atkinson (1998)—a clabject being both a class and an object at the same time. Clabjects are consistent with the notion of powertypes (Odell, 1994) – a powertype being a class that has instances that are subtypes of another class/type – and it is this powertype pattern (Henderson-Sellers and Gonzalez-Perez, 2005; Gonzalez-Perez and Henderson-Sellers, 2006b) (Fig. 4) that underpins the alternative multi-level architecture shown in Fig. 3 as used in ISO/IEC 24744 (ISO/IEC, 2007). Fig. 5 shows an example in which the Document/DocumentKind powertype – part of the metamodel domain of Fig. 3 – is used to create the RequirementsSpecification document clabject, which resides in the Method Domain. The object facet of the clabject provides attributes for Method Domain entities while the class facet provides specification of attributes that are given a value in the Endeavour Domain. Furthermore, the use of powertypes permits both instance-of and generalization relationships between levels. Powertype conformance can thus be defined by extending Bézivin's (2005a) definition by enforcing Method Domain clabjects to map to both a powertype (an instance-of mapping) and a partitioned type (a generalization mapping) (Fig. 5). Indeed, as observed in several papers summarized in Gonzalez-Perez and Henderson-Sellers (2008), application of the four level hierarchy used by the OMG to methodologies results in several contradictory situations—hence the creation of the newer architecture in Fig. 3. For our purpose, we can say that a metamodel describes a domain that is representative of more than one instance in a less abstract domain (i.e. it is a model of models) and, at the same time, it is the core of a modelling language used to describe those instances (Gonzalez-Perez and Henderson-Sellers, 2008). Furthermore, it should be also be noted

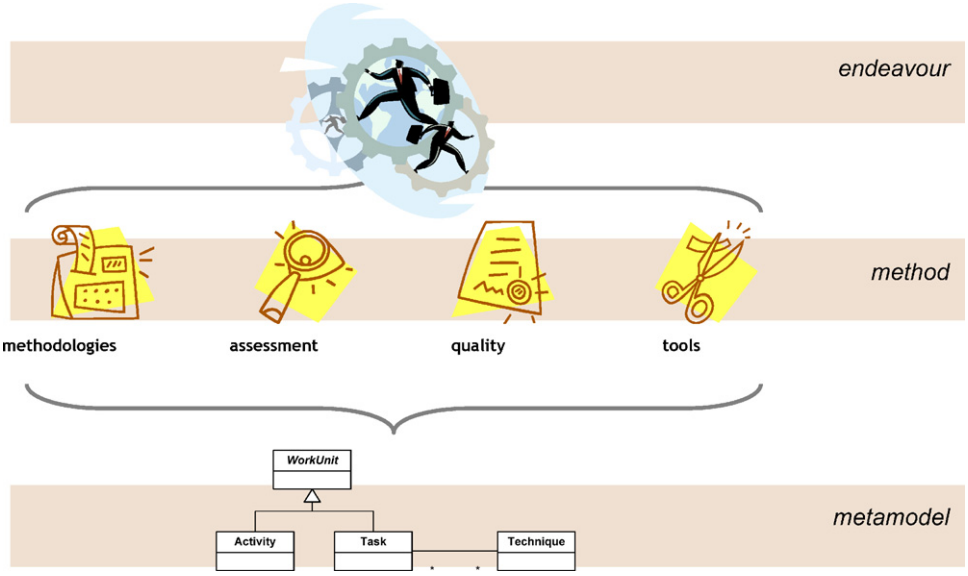

Fig. 3. Three layer architecture of ISO/IEC 24744 International Standard (after Henderson-Sellers, 2006).

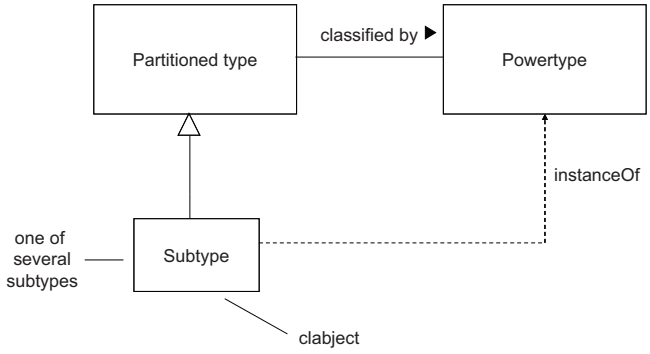

Fig. 4. The elements of a powertype pattern. The "Subtype" is a clabject that inherits from the "Partitioned type" and is an instance of the "Powertype". (Only a single subtype is depicted although there are likely to be several.)

that each model/metamodel usually describes a different domain of discourse, such that the language used for the metamodel domain and the language used for the model domain (although relative) are usually distinct.

It should also be noted that these architectures, particularly that of Fig. 2, are, however, frequently abused—several examples are given by Henderson-Sellers (2007). This means that, in analyzing the published literature for bridges between metamodels and ontologies, care must be taken that a consistent multi-level architecture underpins each published proposal. For example, a recent example of such imprecision is given in Fig. 7 of Pastor et al. (2008), which places the ontology (and also a generic information systems (IS) model) in the M2 level while acknowledging that it may or may not be regarded as a metamodel; while the model-level diagrams of van de Weerd et al. (2007) are (mis)labelled as metamodels.

3. Ontologies

In philosophy, the terms ontology and epistemology are well defined as the study of (a) existence and (b) of (human-created) knowledge, respectively. Unfortunately perhaps, other writers in software engineering (actually the majority) have taken to using the term ontology to mean epistemology (in the sense of a study of human knowledge) or, sometimes, as a noun describing the output of that aforementioned study (e.g. Guizzardi, 2007): the hierarchical or network representation of a domain of knowledge for the banking industry is shown in Fig. 6, i.e. a work product rather than a process of study. Indeed, Corcho et al. (2006) and Guizzardi (2007) stress the difference between "ontology" and "an ontology", the latter referring to this aforementioned work product. Lenat (2005) also notes the addition in software engineering ontology research of generalization and instantiation relationships to philosophical ontology, as well as noting the need to consider probabilistic attributes. In this section, we overview the various perspectives on ontology with an intent to bridge, in Section 4, the modelling and metamodelling efforts in software engineering to the rest of the ontology-related research in knowledge engineering.

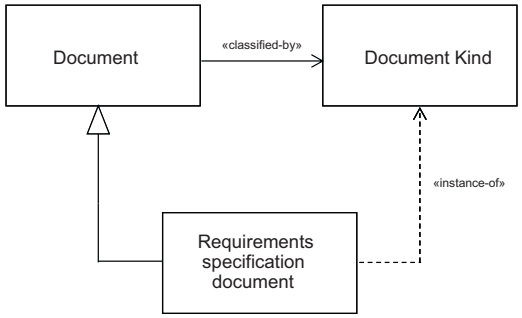

Fig. 5. A typical powertype pattern in software engineering.

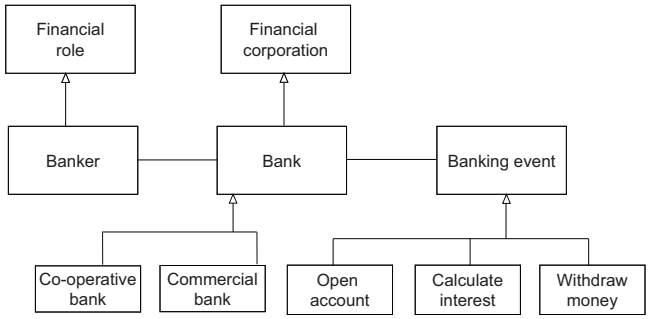

Fig. 6. Part of an ontology pertinent to the banking domain.

According to Gruber (1993), an ontology has two key attributes: firstly, it is an explicit formal construction; secondly, it is agreed upon by many domain experts. The first ensures that an ontology is easy to incorporate within software logic. The second ensures that an ontology is indeed a reusable knowledge unit (Fig. 6). Because ontologies can be *understood* by both humans and computers, they can be used to mediate communication within an IT system, between people themselves or between people and an IT system.

The introduction of ontologies into computing (focussing here on information systems and, primarily, software engineering), while occurring over the last two decades (e.g. Guarino, 1998), has flourished in the last half decade (e.g. Wyssusek and Klaus, 2005). Many of these IS/SE-focussed application areas are brought together in Green and Rosemann's (2005a) volume on business systems analysis. They note the growth not only in application areas and publications but also ontology papers in both specialist and wider IS/SE conferences such as ICIS, CAiSE and ER. Currently, there are many freely available repositories of various domain ontologies, e.g. *Swoogle* (http://swoogle.umbc.edu/), DAML repository (http://www.cse.dmu.ac.uk), *Ontolingua* (http://www.ksl.stanford.edu/software/ontolingua/). Interest in the applied use of ontologies is further demonstrated by new journals (e.g. *Applied Ontology*, launched in 2006) and various ongoing large projects in Europe, America and Australia.

Several studies in the software engineering literature use a definition of ontology based on the treatise of Bunge (1977, 1979), often as elaborated by Wand and Weber (1988, 1993, 1995) into what is commonly referred to as the BWW (or Bunge–Wand–Weber) ontological framework. This framework adapts Bunge's comprehensive philosophical ontology to the IS field by defining "a set of core concepts that can be used to describe the structure and behavior of an information system" (Wand and Weber, 1990, p. 1282). The BWW framework consists of three models (representation, state-tracking, decomposition) (Wand, 1996) of which only the first is generally used for IS/SE ontological assessments (Recker et al., 2007). Example applications of the BWW framework include studies of the ontological completeness (or otherwise) of object-oriented (OO) modelling languages including the UML (Opdahl and Henderson-Sellers, 2000; Opdahl and Henderson-Sellers, 2002) and of the whole-part relationship in OO modelling languages (Opdahl et al., 2001). However, Wyssusek and Klaus (2005) argue that the Bunge approach to ontology is not representative of contemporary discussions in the philosophical ontology research arena, despite its widespread adoption in information systems and software engineering.

The more commonly encountered studies in software engineering, however, tend to adopt the epistemological approach. For instance, the OMG (2005b) state that an ontology defines the common terms and concepts (meaning) used to describe and represent an area of knowledge. An ontology can range in expressivity from a Taxonomy (knowledge with minimal hierarchy or a parent/child structure), to a Thesaurus (words and synonyms), to a Conceptual Model (with more complex knowledge), to a Logical Theory (with very rich, complex, consistent and meaningful knowledge). While this is clearly consistent with the philosophical definition of epistemology, we will bow to common SE usage, described by Guizzardi (2005, p. 61) as a "liberal meaning", and from hereon use the term ontology as being a product (or process) associated with (human-derived) knowledge representation.

"Ontology" in association with "software engineering" is becoming increasingly commonplace in the literature (the term "ontology-driven information systems" being much older, e.g. Guarino, 1998). While paper titles such as that of Pastor et al. (2008) stress the ontological content, this is still in its infancy. More substantial integration is achieved, for instance, in Hesse (2008a),

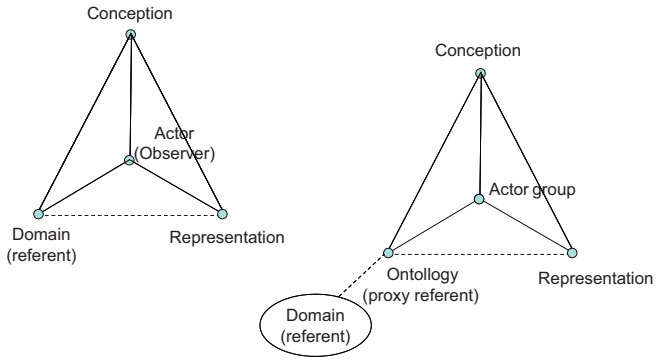

Fig. 7. The original (left) and modified (right) semiotic tetrahedron (after Hesse, 2008a, figures 5 and 6, ©2008) With kind permission of Springer Science + Business Media.

who links ontologies with the FRISCO framework (Falkenberg et al., 1998) by taking its semiotic tetrahedron and replacing the Domain (referent) by a linkage to Ontology, which, in turn, is more solidly linked to Representation (Fig. 7); in the Ontology-based software Development Environment (ODE) of Falbo et al. (2005); and in the MOBMAS methodology of Tran et al. (2006) and Tran and Low (2008). Beydoun et al. (2005, 2006b) argue that an ontology-based methodology, like MOBMAS, can provide an integrative platform across other methodologies as well as improving the quality of both product and process. These ideas are then applied to the early requirements phase in Beydoun et al. (2008). Other ontology-focussed methods are summarized in Corcho et al. (2006).

When used in software engineering, additional characteristics for an ontology are generally required. Corcho et al. (2006) point out that, from a computer science viewpoint, an ontology, to be useful, must be "understandable" by a computer—this is usually termed "formal". OMG (2005b) interprets this to mean that there must be a set of definitional rules, as specified in a metamodel: for example, by the Ontology Definition Metamodel (ODM); although Guizzardi (2005, p. 5) points out that "formal" here really means "having form" rather than meaning precise or mathematical. Rilling et al. (2007) also underline the formal nature of an ontology but equate it to the context of a mental model, citing the cognitive science underpinning of Johnson-Laird (1983). A second required characteristic is that it should represent *shared* knowledge (e.g. Gruber, 1993; Noy and McGuinness, 2001). Finally, although Corcho et al. (2006) quote Neches et al. (1996) as describing an ontology in terms of its *vocabulary*, Ruiz and Hilera (2006, p. 51) stress that, since it is essential that an ontology is natural language independent, it should not be equated with a vocabulary (or any other construct in a particular natural language). On the other hand, as noted by Neches et al. (1996) and Gruber (1993), an ontology can be used to define a vocabulary. Guarino (1998) uses this notion to differentiate between an ontology linked to a representation in a specific vocabulary. Aßmann et al. (2006) also note that an ontology has the property of upholding the so-called open-world assumption, i.e. anything not explicitly expressed by an ontology is unknown.

Some additional clarity is added in the various classification schemes that have been published. A number of classification schemes have been proposed, some based on structural considerations (e.g. Van Heijst et al., 1997), others on the areas of likely application (e.g. Jurisica et al., 1999) or on a set of binary characteristics (Ermolayev et al., 2008). Other kinds of ontologies have been proposed in very specific domains; for example, the use of service ontologies for designing multi-agent systems (e.g. Nodine and Unruh, 1998). Gomez-Perez et al. (2004) add the concept of the "richness of the internal structure" (e.g. whether the ontology is a controlled vocabulary, a glossary, a thesaurus, a frame, a formal

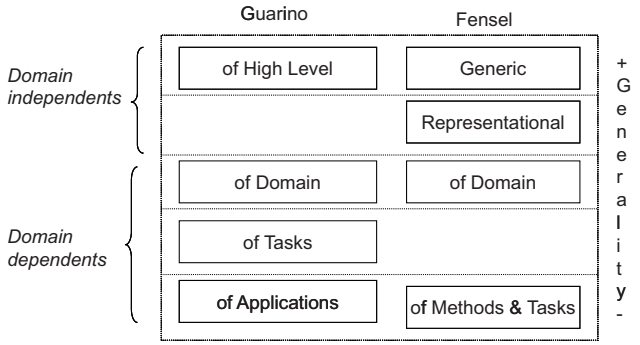

Fig. 8. Kinds of ontologies according to the generality level (after Ruiz and Hilera, 2006, figure 2.1, ©2006) With kind permission of Springer Science + Business Media.

hierarchy and so on). This range is also reflected in the ontology discussion of Pidcock (2009). Following Guarino (1998), Ruiz and Hilera (2006) and Guizzardi (2005, p. 67) identify four general kinds of ontologies. The first, *high-level ontologies*, correspond directly with those identified above as concomitant with Bunge's work. The other three (domain ontologies, task ontologies, application ontologies) are more aligned with the epistemological approach outlined above.

Much of the ontological literature is focussed on the notion of a domain ontology and, in many publications, it is this category that is given the generic appellation *ontology*. A particular kind of domain ontology is a reference ontology (Guizzardi (2005, p. 90)—see also (Guarino, 1998)), although a *domain reference ontology* is equated by Guizzardi (2005, p. 37) to a domain ontology. In contrast, Kazmierczak and Milton (2005) use the term reference ontology for the philosophical meaning of ontology, i.e. the shared conceptualization (using "conceptualization" in the ontological sense as also used here), as noted above (Guarino, 1998). In a very different vein, Menzel (2003) equates a reference ontology with a foundational ontology, thus illustrating once again the very disparate semantics of another ontology-related term.

Our view is that ontologies constitute part of the context description that has to be taken into account for successful system development. In other words, ontologies can convey useful information for software developers to understand system requirements as well as information that may be useful for a project manager to comprehend subtleties that can impact the choice of the best methodology and/or modelling language to use or adapt; for instance, by means of *situational method engineering* (e.g. Brinkkemper et al., 1996; Henderson-Sellers, 2003) although the usefulness of ontologies in method engineering is largely untapped.

4. Ontologies in software engineering: a bridge between models, metamodels and ontologies

Having established the relationship between models and metamodels (Section 2) and given an overview of ontologies (Section 3), we can now ask to which of these (or both) it is useful, both theoretically and pragmatically, to link our software engineering-defined "ontology" requirement. In particular, we consider domain ontologies as an exemplar of the epistemological approach identified in Section 3, together with Guizzardi's (2005, 2007) high-level ontologies. In other words, in this paper we will classify the ontological literature into just two broad areas (high-level ontologies and domain ontologies) since, as we shall shortly show, it is these that have most relevance to both modelling/metamodelling and to software engineering theory – this is in accordance with Fig. 2.1 of Ruiz and Hilera (2006) – here reproduced as Fig. 8 – who compare two classification schemes (of Guarino (1998) and Fensel (2004)) and differentiate between domain-independent ontologies and domain-dependent ontologies: the same discrimination that we adopt for the remainder of this paper.

We now need to address the following question: How might a domain ontology and a high-level ontology be linked to a model and/or a metamodel? We examine possible links to models in Section 4.1 and to metamodels in Section 4.2. In Section 4.3, we examine areas of the literature where the metamodel/model and/or ontology/domain ontology boundaries are blurred or ambiguous. Although, initially, we will attempt to align "ontology" with one or other of the (Mx) levels of the OMG 4-level architecture of Fig. 2, we propose an architecture that provides an underpinning for the approaches discussed in Sections 4.1–4.2 and attempt to resolve the ambiguities identified in this section.

4.1. Domain ontologies and models

Atkinson et al. (2006) ask a similar question regarding whether ontologies and models are different technologies. We will first explore this as a lemma that we can later use to answer our main question regarding a bridge between ontologies and metamodels. Atkinson et al. argue that ontologies and models appear to be derived from different subfields of computing and knowledge representation. They note several projects, for example within the Object Management Group and the World Wide Web Consortium (W3C), aimed at producing a bridge between the technologies. These authors identify a number of differences:

- Models focus on realization; ontologies do not.
- Ontologies are typically, but not necessarily, for run-time knowledge exploitation; they are formal and can support reasoning. In contrast, they argue that models do none of these – although current research into executable models could easily negate this claim.
- Models use the closed world assumption, ontologies the open world assumption.

Their conclusion, as noted also by Guizzardi (2005, p. 6) and Aßmann et al. (2006, p. 256), is that ontologies are a subset of models since ontologies fulfil the criteria for being models but have additional characteristics, i.e. they are specializations in the OO sense. This observation would invalidate Atkinson et al.'s conclusion that ontologies are redundant (as noted by Gonzalez-Perez, 2006) since specializations possess characteristics in addition to those of their "parent".

While noting that much of ontology design originated in OO design, Noy and McGuinness (2001) suggest that OO stresses operational properties of classes rather than the structural properties of classes, which are the focus of ontology design. In contrast, OMG (2005b) take a broader meaning to the term "conceptual model" than the database-restricted use of Ruiz and Hilera (2006, p. 63). They state (OMG, 2005b, p. 47) that "An ontology is a conceptual model, and shares characteristics with more traditional data models". The OMG ODM standard (OMG, 2005b) gives reasons why UML is unsatisfactory for documenting ontologies. It notes some missing concepts in the UML—in particular, the treatment of disjoint classes, set intersection and set complement. This, they argue, prevents the use of automated reasoning for UML models, which (they argue) is important for ontologies. They further argue that ontology instances may also be required without the prior definition of a class—not permissible when using UML, since UML depicts classes primarily and only secondarily supports the notion of an object.

Many other authors equate ontologies with models (e.g. Nirenburg, 2004; Gašević et al., 2007) although noting the difference in intent, i.e. that an ontology is descriptive and a model in SE is typically (but not always) prescriptive, e.g. Wand and Weber (2005). For example, Gruber (1993) states that "Ontologies are

also like conceptual schemata in database systems" which "provide a logical description of shared data"; Aßmann et al. (2006) noted that ontologies do not describe systems in the sense of specific problem/solution environments, only domains in the much broader sense, and can therefore be regarded as analysis models but not design or implementation models. They go on to suggest that, if the intention is to use an ontology for system design, then it is no longer an ontology but rather is a prescriptive or specification model. Guarino (1998) suggests that "In some cases, the term "ontology" is just a fancy name denoting the result of familiar activities like conceptual analysis and domain modelling"—clearly indicating that he regards an ontology as belonging to the model domain and not the metamodel domain.

In some contrast to the notion of ontologies being focussed at their specification level, i.e. the metamodel, most "ontologies" found by a web search and documented, for example in Protégé, are hierarchies of terms in a specific (often commercial) domain. For instance, we have located an ontology for newspaper publishing containing elements such as editor, journalist and printing press; an ontology for health care with concepts including doctor and patients. Such an ontological hierarchy bears a good correspondence to a UML model (M1) that might be constructed if one were building software as opposed to the ontological usage of knowledge representation (of the real world).

Guarino (2008) notes that most conceptual modelling approaches are ontologically neutral, in accordance with proposals to link ontologies to modelling languages—for instance, the addition of an Ontology class in the agent-oriented metamodel of Beydoun et al. (2006a) and the simpler merger at the requirements and analysis stage proposed by Hesse (2008b, Fig. 2). Guarino further notes that conceptual models and ontologies belong to the same category—that of human artefacts, while Hesse (2008b) similarly compares ontologies and conceptual models.

4.2. Ontologies and metamodels

As well as comparisons of ontologies and models in the literature, several authors directly investigate comparisons of ontologies and (conceptual) metamodels—as we do here. However, many authors (e.g. Ruiz and Hilera, 2006, p. 64) note a continuing confusion between the terms metamodel and ontology. They posit that this is because they are often depicted with the same language. They suggest differences based on focus (metamodels seek to improve similar but different models whereas an ontology does this for knowledge models) and, in their mind most importantly, on the nature of their application, arguing that an ontology is descriptive (of the problem domain) whereas a metamodel is prescriptive, belonging to the solution domain. Aßmann et al. (2006) also note this terminological confusion, arguing against both Devedzic's (2002) statement that "an ontology is a meta-model describing how to build models" and Gruber's (1993) suggestion that ontologies are specifications since ontologies, being shared knowledge, must always be descriptive and never prescriptive (e.g. Hesse, 2006).

In an example of an apparently inconsistent approach, Dillon et al.'s (2008) appendices, graphically depicting their ontologies, would appear to be at the M2 level (see Fig. 3), with classes such as Swimlane and Activity; yet the application of stereotypes, only valid at the M1 level, is therefore not in accordance with the UML language specification. Their Fig. 1 also appears to mix M2 and M1 concepts together within a single domain, e.g. concepts with project-specific data. This raises the question (analyzed below) of whether it is permissible for elements in an ontology to belong to more than one Mx level. A similar apparent confounding of levels is evident in the diagrams of Falbo et al. (2005) that describe their "ontology", as used in ODE (Falbo et al., 2002, 2003). From the names of the elements in Fig. 9 (e.g. Process, Activity, Artifact),

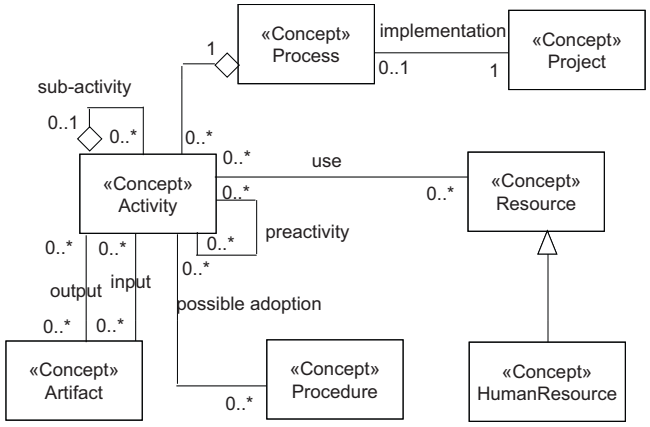

Fig. 9. Part of the software process ontology (after Falbo et al., 2005).

which depicts their "software process ontology", one can conclude it is at the "M2" or metalevel. However, this figure uses stereotypes, which, as noted above, are only valid in M1 models. This usage suggests that Fig. 9 is at the model level, i.e. they equate ontology and model—a contradiction existing within the single diagram. Indeed, the authors go on to explain that a class like "Human Resource" in this figure, which is an instance of the Concept class in their so-called "meta-ontology class model" (Fig. 10), "originates two classes: *KHumanResource* and *HumanResource*". Their examples show that this is in fact a clabject (Atkinson, 1997) and that their approach is compatible with the powertype approach explained by Gonzalez-Perez and Henderson-Sellers (2006b) and used in ISO/IEC (2007)—see earlier discussion. They link the class facet of the clabject to the "metalevel" and the object facet to the "base level" and the elements in Fig. 10 as belonging to the ontological level (the model being described as a "meta-ontology"). It is not clear, however, that these three levels correspond to either the OMG (Fig. 2) or the ISO (Fig. 3) multi-level architecture since, in these two architectures, a concept like Activity occurs at the metamodel level (M2 in OMG) rather than the (apparently) model (M1) level as described by Falbo et al. (2005).

Similarly, but without comment, Guizzardi (2005, p. 62 et seq.) suggests that ODE and LINGO *ontologies* bear a strong resemblance to *metamodels* of the OMG (e.g. metamodels for quality and for process). However, ontologies need an ontology representation language and it is this language that is the ontology metamodel (similar to the role of the UML in OO design).

Guizzardi and Wagner (2005a), in the context of agent modelling languages, propose a unified foundational ontology (UFO). This is a merger of the General Formal Ontology (GFO) developed by the OntoMed research group (Degen et al., 2001) and the OntoClean ontology (Welty and Guarino, 2001) plus the Descriptive Ontology

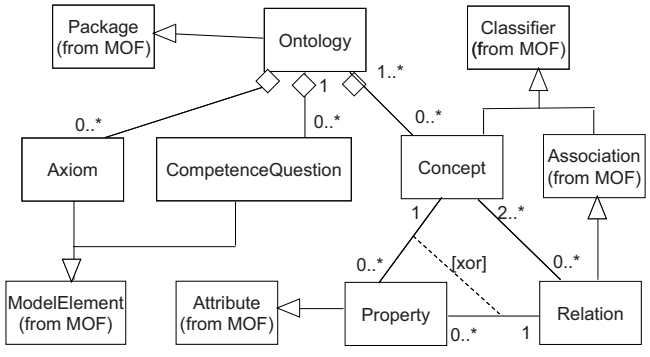

Fig. 10. Part of ODE's meta-ontology class model (after Falbo et al., 2005).

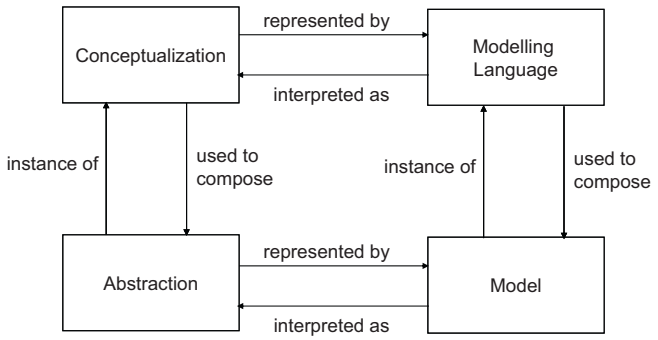

Fig. 11. Relations between Conceptualization, Abstraction, Model and Modelling Language (after Guizzardi, 2007). Reprinted with permission from IOS Press.

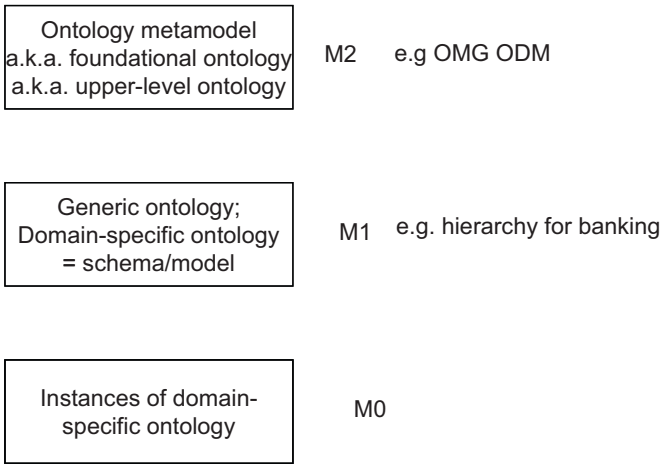

Fig. 12. Three level ontology architecture suggested by the Ontology Definition Metamodel of the Object Management Group.

for Linguistic and Cognitive Engineering (DOLCE) by the ISTC-CNR-LOA research group (http://wonderweb.semanticweb.org/). The UFO is categorized as an upper level ontology (a.k.a. foundational ontology), equivalent in "level" to the BWW ontology described above. An application to business modelling is given in Guizzardi and Wagner (2005b) and to goals in agent technology by Guizzardi et al. (2007). Guizzardi (2005, p. 86) states that a foundational ontology is a *meta-ontology*. Since he, and others, effectively equates "ontology" with "model", then we must conclude that a meta-ontology can be effectively equated with metamodel, at least in the OMG sense. Indeed, in Guizzardi and Wagner (2005a), it is clearly stated that a foundational ontology can be represented as a MOF model, MOF being a language for defining modelling languages, i.e. it is used as a metamodelling language. In other words, a foundational ontology is at the metamodel level in that it is equivalent to the UML or the ER (Entity-Relationship) definition. This means that we need to reassess Fig. 8 because "domain independence" is also seen as a feature of a meta-ontology while, in contrast, a generic model is widely recognized as *not* being at this meta level (see e.g. Henderson-Sellers, 2007). Thus, the domain independent kinds of ontology can be further sub-classified into domain independent and foundational in contrast to domain independent (and not foundational).

In a later study, Guizzardi (2007) clearly differentiates between a cognitive framework and a modelling framework and how these are linked. Fig. 11 shows that real world abstractions (mental models for specific problems) are specific instances of very broadly applicable conceptualizations and that these are represented by a model and a modelling language, respectively, wherein the abstract syntax of the modelling language is a metamodel. He further argues that this approach allows the comparison of conceptualizations as intensional structures and metamodels as represented by logical theories. The real world abstractions are related to a domain ontology while the conceptualizations can be depicted using a foundational ontology.

4.3. Ambiguities and a possible resolution framework

In the context of knowledge modelling and agent messages, Cranefield et al. (2000) note that "the distinction between an ontology and an abstract syntax (which defines the concepts that can be expressed in a language) becomes blurred" (a blurring also evident in Gonzalez-Perez and Henderson-Sellers, 2006a). Nevertheless, Cranefield et al. clearly identify (their Fig. 6) domain ontologies as belonging to level M1 and, interestingly, that all these domain ontologies are representable in UML.

In a section entitled "combining metamodels and ontologies to achieve semantic interoperability" – words suggesting that ontologies belong to the metalevel – Karagiannis et al. (2008) go on to describe "semantic mappings between metamodel elements and ontology concepts". Arguably this latter statement, at odds with the former, can be interpreted as ontology concepts being the classes in the ontology metamodel—as for instance documented in the OMG's ODM (OMG, 2005b).

Contrasting several chapters from the same book (Calero et al., 2006), we see that, while the software maintenance ontology of Anquetil et al. (2006) and the software development environment ontology of de Oliveira et al. (2006) clearly discuss a domain ontology, the ontology for software measurement of Bertoa et al. (2006), the ontology for SQL:2003 of Calero and Piattini (2006) and the ontology for software development methodologies and endeavours (Gonzalez-Perez and Henderson-Sellers, 2006a) are all clearly defined in terms of a metamodel. Indeed, OMG (2005b) clearly differentiate between the Web Ontology Language (OWL) *metamodel* that allows users to define ontology models (p. 87) and the ontology that is "generally specified as a system of classes and properties (the structure) which is populated by instances (the extents)" (p. 57). Hence, the Universe of Discourse is described by a set of ontologies (OMG, 2005b, p. 42) where ontologies are used to enhance the target system and be complementary to UML modelling artefacts (p. 44). In other words, ontologies belong to the M1 level (Fig. 2) or Method Domain (Fig. 3) since (OMG, 2005b, p. 47 & 51) an ontology is a conceptual model, sharing characteristics with more traditional data models. The parallels between metalevels of the OMG and those required for ontologies are endorsed by Bézivin et al. (2005). This OMG ODM approach suggests a multi-level ontology architecture as depicted in Fig. 12. Here, the "M2" level is equivalent with Guizzardi and Wagner's (2005a) foundational ontology, with the OMG's ODM and with the term "upper level ontology" such as that represented by the BWW model. The "M1" level includes not only domain-specific ontologies (such as that for, say, a banking domain) but also a domain-independent generic ontology. Instances of elements of a domain-specific ontology (Fig. 12) are discussed by Noy and McGuinness (2001). They argue that the depth in the ontology hierarchy at which this instantiation occurs is dependent upon context, making no attempt to align with the strict metamodelling architecture of Fig. 2. Guizzardi (2005, p. 68) describes instances of an ontology as "concrete engineering artifacts".

Fig. 13 positions upper-level ontologies and domain ontologies within the typical multi-level metamodelling framework as adopted, for instance, by the OMG's standards such as UML (Fig. 2). Aßmann et al. (2006) argue that a philosophically focussed "world ontology" encompasses both the conceptualization level (upper-level ontology) as well as domain ontologies at a lower level in this pyramid. They depict the relationship between the two

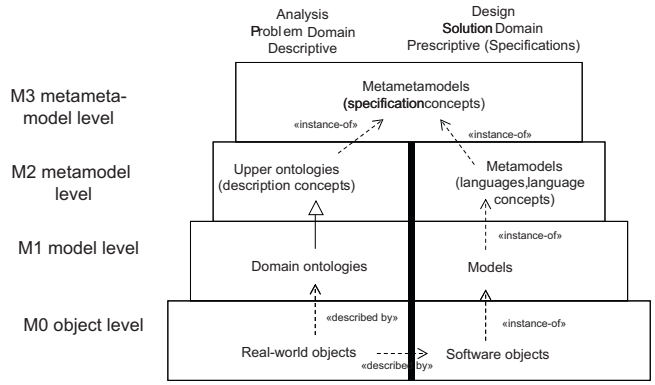

Fig. 13. Proposed integration of an ontological view with a four-level metamodel architecture (after Aßmann et al., 2006, figure 9.7, ©2006). With kind permission of Springer Science + Business Media.

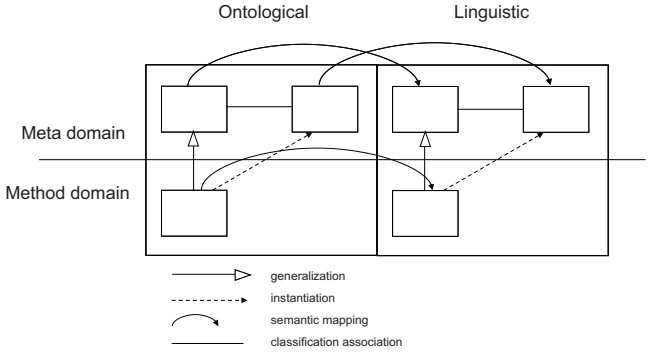

Fig. 14. Two levels that illustrate the integration of the powertype pattern and a semantic mapping between the model and the ontology.

as a specialization relationship rather than the expected instantiation relationship (more strictly a conformance relationship). They suggest that this has an historical basis, perhaps linked to the difference between descriptive versus prescriptive modelling (although no citation or validation is given). They then argue that specialization may well be an unnecessary constraint, suggesting instead the use of a "described-by" relationship. This links well to the modelling-focussed arguments of Gonzalez-Perez and Henderson-Sellers (2007), Seidewitz (2003) that the OMG's use of strict metamodelling and its insistence on only using "instance-of" relationships between the levels of this pyramid is overly restrictive.

Further answers may be found in the following (Gruber, 1993): "An *ontology* is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an ontology is a systematic account of Existence. For knowledge-based systems, what "exists" is exactly that which can be represented. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the Universe of Discourse. This set of objects and the describable relationships among them, are reflected in the representational *vocabulary* with which a knowledge-based program represents knowledge." [It should be noted that, in philosophy, Ontology is not dependent upon human representation or our ability to conceptualize things that exist in the universe.]

Thus, from the Gruber quotation above, we can deduce that if an ontology (as opposed to Ontology in the pure philosophy sense) refers to a Universe of Discourse and to conceptualization, then the term "(an) ontology" would appear to be equally applicable to either M1 or M2 (although not both simultaneously), in just the same way that the term "model" can be applied to a M1 UML visualization (e.g. a system design) or to a M2 visualization (e.g. the UML metamodel). This may explain the ambiguity regarding whether an ontology is an M1 or M2 thing.

Thus, for example, when Green and Rosemann (2005b) state that they converted the Bunge–Wand–Weber ontology, as well as its modelling grammar, into an ER-based metamodel, they imply both are M2. Their concepts, such as "thing", are clearly compatible with elements of Bunge's approach as well as the concepts listed in Guizzardi and Wagner's (2005a) foundational ontology. Furthermore, Holten et al. (2005) state, immediately after a discussion of metamodels, that "The analysis of a modelling language and a metalanguage leads to the concept of an ontology".

While meta-pyramids, such as that of Aßmann et al. (2006) (Fig. 13) unite the modelling aspects and the ontology aspects only through the M3 level and the M0 (real world) level, Saeki and Kaiya (2007) stress the need to provide a semantic mapping at each level, expressing this in terms of the architecture of Fig. 2. However, if we now introduce the architecture of Fig. 3, we also introduce powertype patterns for several of the concepts in the Metamodel Domain, wherein each powertype pattern leads to a clabject in the Method Domain. Combining these ideas with Saeki and Kaiya's semantic mapping creates a generic framework for future refinement for use in software engineering to create a synergism between the modelling world and the ontology world. The two most relevant "levels" (domains) are shown in Fig. 14. These powertype patterns are most useful for process-focussed methodology elements and are less useful when dealing with modelling languages or ontological domains expressing static bodies of knowledge.

In summary, we can conclude that the term "ontology" can be applied at several "metalevels", as can the term "model". Of specific interest to software engineering are domain ontologies (that parallel analysis and design models) and foundational ontologies (a.k.a. meta-ontologies) that have similar characteristics to modelling languages and metamodels. The metalevels may be aligned with the OMG 4 level architecture shown in Fig. 2 or the domain-focussed architecture of Fig. 3. The former is sufficient for discussing modelling languages and data-focussed ontologies whereas the latter is needed for broader methodology characterization. It is also generally agreed that both a model and a domain ontology (M1 or Method Domain) must be represented by a language, itself defined by a metamodel (M2). Ontologies and models are connected using a semantic mapping (Saeki and Kaiya, 2007). Metamodel-defined languages commonly used for ontology descriptions include DL (Description Logics) (Rilling et al., 2007) or RDF[2]/OWL (OMG, 2005b) or a proposed extension to UML (Evermann, 2005; Hesse, 2008a). Such languages provide reasoning support not possible in software modelling languages such as UML or ER (OMG, 2005b) but necessary for ontologies.

5. Related work

Most of the related work has already been cited in the discussion in Sections 2–4. Work like that of Guizzardi (2007) comes nearest to the work described in this paper. However, Guizzardi's work is aimed at introducing metamodels to the ontology researcher, his own background, e.g. Guizzardi (2005) being solidly in ontology. In contrast, our aim here is to answer the question "How can software engineers benefit from introducing ontologies into their thinking". While authors such as Hesse (2008a) offers an answer to this question at the broad software engineering level, here we focus at the detailed level of the formality of the metamodel in order to

[2] Resource Description Framework.

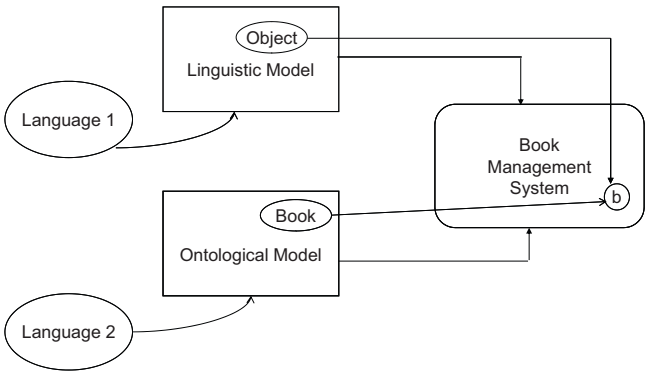

Fig. 15. An example in which the Book Management System is the System under Study (SUS) for two different models: the Ontological Model (which contains elements such as Book) and the Linguistic Model that contains elements such as Object. Each of these two models is depicted with a different language. The object b in the Book Management System is represented by Book in the ontological model but by Object in the linguistic model (adapted from Gonzalez-Perez and Henderson-Sellers, 2007).

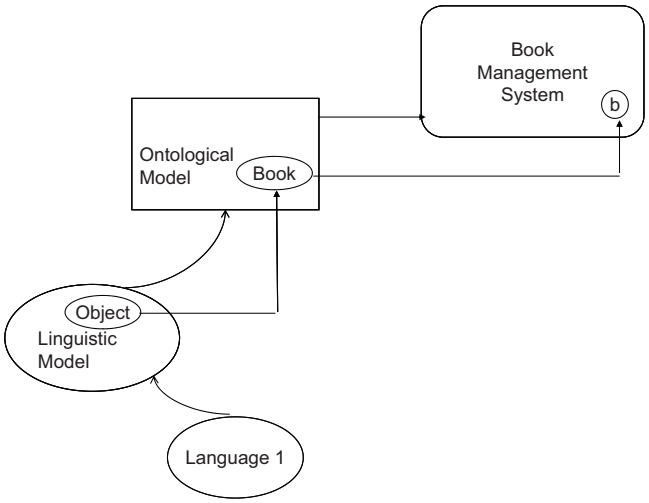

Fig. 16. A redrawn version of Fig. 15 that shows that in fact the Linguistic Model *is* Language 2 (adapted from Gonzalez-Perez and Henderson-Sellers, 2007).

identify a "bridge" between the conceptualizations represented by metamodels and those by ontologies, particularly foundational and domain ontologies—and identifying the difference between these latter two in the software engineering/modelling context.

Less directly related are papers debating linguistic versus ontological metamodelling, as originally discussed by Atkinson and Kühne (2003). They describe the modelling/metamodelling approach that focusses on domain concepts (at different abstraction levels) as "ontological" in contrast to the adoption of concepts like class, object, relationship which they label as "linguistic" (earlier called logical and physical, respectively, e.g. Atkinson and Kühne, 2002); pointing out that the OMG strict metamodelling hierarchy is most closely aligned with the linguistic approach and is weak in support of ontological modelling (see also Laarman and Kurtev, 2010).

In our previous study, aimed at identifying a conceptual foundation for models and modelling languages (Gonzalez-Perez and Henderson-Sellers, 2007) and based on an assumption of a homomorphism between model and the real system, we argued that these two viewpoints (ontological and linguistic) actually produce two different and distinct models for any specific situation in which each of these two distinct models is described using a different language (Fig. 15). Since, in this figure, Object in the linguistic model is mapped to Book in the ontological model, it was demonstrated that in fact the linguistic model and Language 2 (in Fig. 15) are the same thing (Fig. 16).

In his analysis of ontologies and metamodels from the ontological engineer's viewpoint, Guizzardi (2007) applies the pattern of Fig. 11 to a specific example in order to create the metamodel for a domain specific language. For the real world genealogical domain, he creates a model of concepts in that specific domain depicted using a "M1" level UML diagram, which he describes as an ontology. In this ontology, each class is conformant to a class in a version of UML 2.0 that he has redesigned based on his Foundational Ontology. From this (domain) ontology, he then creates a number of valid models. This can be pictured as a double application of Fig. 11 (as shown in Fig. 17); clearly this does not fit into the OMG-style metalevel hierarchy since the lowest level of Fig. 17 corresponds to M1 and the top level to M2, thus leaving the middle level of Fig. 17 unmapped. Alternatively, the instance-of/used-to-compose relationships in the upper part of this diagram could perhaps be represented more usefully as generalizations/specializations, thus paralleling Fig. 13—a topic for future study (see Section 6). Rather, the solution of Fig. 11 is more aligned with the two-dimensional representations used by Atkinson and Kuhne and others in terms of their varying attempts at resolution of the ontologic/linguistic metamodelling challenge—although the author (Guizzardi) does not couch his argument in such terms.

A similar approach is taken by Laarman and Kurtev (2010) who use the work of Guizzardi applied to the linguistic/ontological paradox highlighted by Atkinson and Kühne. These authors create a "four-category ontology" on which they build an Ontology Grounded Metalanguage (OGML) that incorporates both classical linguistic elements (as found for instance in the UML/MDA world) and ontological elements—specifically Individual (with two subtypes of Substantial and Moment) and Universal (with two subtypes of Substantial Universal and Moment Universal). In OGML, they show that both linguistic and ontological instantiations can be treated equally with a single "instance-of" relationship. Notwithstanding, they acknowledge that this interpretation appears different to that of other authors (e.g. Kühne, 2006a,b; Gašević et al., 2007) – although possibly reconcilable – and that the relationships between linguistic instantiation, ontological instantiation and metamodels are still debatable.

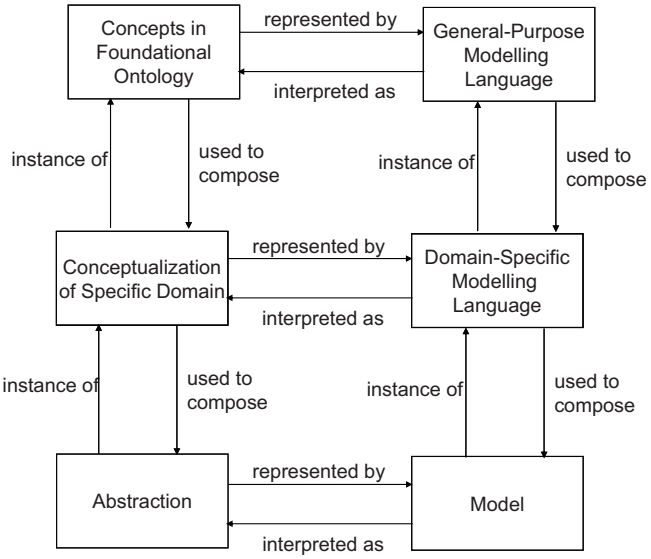

Fig. 17. Three "levels" created by the double application of the pattern of Fig. 11 illustrating the need for both a domain specific (modelling) language and a general purpose modelling language, both of which are underpinned by a metamodel.

A further line of research closely allied to our topic is the introduction of a "megamodel" (Favre, 2005), which has arisen largely from studies of model-driven engineering (MDE). Using the definition (Bézivin, 2005b) that "a megamodel is a model which elements represent models, metamodels and other global entities", Favre (2006) goes on to describe megamodel patterns and uses the megamodel idea to try to obviate the megamuddle noted by Solberg et al. (2005). These ideas are then linked to ontologies by Gašević et al. (2007), who suggest that a megamodel can be regarded as an ontology of MDE. However, they note that Favre's megamodel does not distinguish between the intension and extension of a class and that, overall, the approach is still largely linguistic such that extensions are necessary in order to capture all necessary concepts for its use for ontological metamodelling. They also conclude that an ontological metamodel should be conformant to the linguistic metamodel of the modelling language being used—which is in agreement with the statement in Fig. 16 that the linguistic model acts as the (modelling) language for the ontological model.

In addition to the work of Hesse (2008a,b), Pastor et al. (2008) and Tran et al. (2006) that are focussed on integrating ontological ideas into software engineering, other authors have found a specific focus area—for instance, Höfferer (2007) investigates the use of metamodels and ontologies in the context of the interoperability of business processes with suggestions on the use of ontologies in service-oriented architectures; Zhu and Feng (2010) explore how to use an ontology in data mining focussing on a database context. Happel and Seedorf (2006) consider the overlap of semantic web technologies and software engineering and identify the various lifecycle stages at which ontologies may be useful. For each stage, from requirements analysis through to testing, they identify a problem that ontologies may help to address and itemize the individual advantages of such an approach.

6. Conclusions and future work

Despite much ambiguity in the literature regarding the term "ontology", there are two specific kinds of ontology that are useful for software engineering: (i) domain ontologies, which are used to create a vocabulary for a specific application domain (e.g. banking, health) and are vital to ensure that elements in the model have well-defined semantics; and (ii) meta-ontologies or foundational ontologies, which are equivalent in nature to the metamodel of a modelling language and thus encapsulate the concepts needed for creating domain ontologies. Both kinds of ontology need a language with which they can be described. Often a modelling language such as UML or ER is found to be useful, although several authors note that, without extensions (as proposed by Guizzardi, 2005), these modelling languages do not provide adequate reasoning support for use as a complete ontology description language.

Although we have identified two different "flavours" of ontologies (upper-level/foundational and domain), for which we have identified an appropriate "metalevel" in an architecture such as those depicted in Figs. 2 and 3, we can also offer a unifying statement in conclusion. We propose that, since an ontology is agreed to be a shared conceptualization, "an ontology" in software engineering is a formal, often taxonomic organization of concepts. This definition can then be applied at the highest abstraction level to give foundational ontologies, metamodels and modelling languages; at a domain-specific level (e.g. for a health domain, a banking domain), leading to the idea of a domain ontology and to domain-specific languages (DSLs); to problem-specific models with an associated subset of a domain ontology; or, finally, to an instance of a domain ontology to depict the state of affairs present in reality (the last option is not discussed here—but see Guizzardi, 2007, Fig. 14). In other words, the "ontology concept" can be applied at various "metalevels" in just the same way that the "model concept" can be.

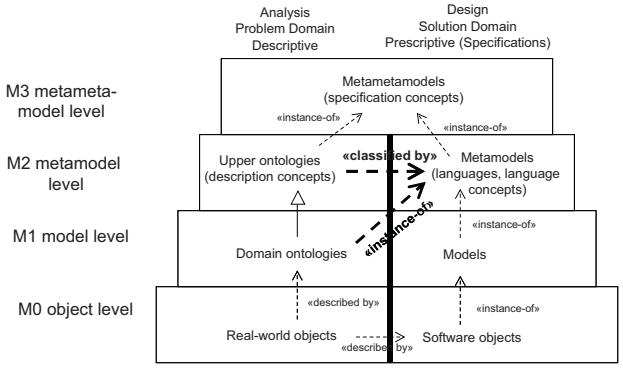

Fig. 18. Introduction of two additional relationships (a classified-by between Upper Ontologies and Metamodel and an instance-of relationship between Domain Ontologies and Metamodels) that together create a powertype that is superimposed on the pyramid structure of Aßmann et al. (2006) as depicted in Fig. 13. (These additions have been emboldened for clarity.)

Of particular interest for *future work* in this area is the potential application of the ISO/IEC architecture of Fig. 3 to the representation of an ontology and its corresponding metamodel. Such a "hybrid" architecture helps to resolve both the internal contradiction of the ODE approach of Falbo et al. (2005) discussed above and also the architecture proposal by Aßmann et al. (2006) as demonstrated in Fig. 13. It is conjectured here (but is a topic for future work) that a combination of instance-of and specialization, in the form of the powertype relationship (Gonzalez-Perez and Henderson-Sellers, 2006b), may provide a more useful representation—see Fig. 18, where we postulate an additional "classification" relationship between the elements in an upper ontology and the corresponding elements in the metamodel. Completing the powertype pattern (Fig. 5) requires also the addition of an instantiation relationship from domain ontologies to metamodels. If this can be substantiated by further research, then the hybrid proposal of Aßmann et al. (2006) will have been reconciled with the powertype pattern as used in ISO/IEC 24744 (ISO/IEC, 2007). Additionally, the underpinning rationale of Figs. 13 and 18 may provide an alternative avenue of reconciliation between the ontological and the linguistic metamodelling approaches promulgated by Atkinson and Kühne (2002, 2003), for instance as synopsized in Gonzalez-Perez and Henderson-Sellers (2008).

A second topic for future extension to the work reported here is solving the challenge posed in Beydoun et al. (2008) in the creation of a multi-agent system that simultaneously provides knowledge requirements to different Platform Specific Models within an OMG-like model-driven architecture (MDA) (OMG, 2003). Although these authors propose a set of six requirements for developing a multi-agent system (MAS) using an ontology-based software engineering approach, this is but a first step towards further development.

Finally, an interesting topic for future work is further reconciliation and integration of the megamodel work of Favre (2004, 2005, 2006) and Gašević et al. (2007) in enhancing our understanding of the best way to develop a future modelling language that gives equal weight to ontological metamodelling and linguistic metamodelling (see also Saeki and Kaiya, 2007). At the same time, researchers could reconsider whether this dichotomy of ontological versus linguistic metamodel(ling) is worth preserving or whether a more metaphysical approach to conceptual modelling based on endurants and perdurants (e.g. Guizzardi, 2005) might be a more fruitful avenue for exploration. The addition of these more abstract concepts to the theory of conceptual modelling (and thus of metamodelling) should also lead to our ability to produce more flexible toolsets (e.g. Gonzalez-Perez, 2005) to support the ontology/modelling bridge that has been the focus of this paper.

Acknowledgements

I wish to thank Charlotte Hug, Graham Low, Ghassan Beydoun and Cesar Gonzalez-Perez for their suggestions regarding earlier versions of this paper. Thanks also to the anonymous reviews for their useful comments. This research is funded by the Australian Research Council under grant number DP0878172. This is contribution number 10/01 of the Centre for Object Technology Applications and Research within the Centre for Human Centred Technology Design of the University of Technology, Sydney.

References

Anquetil, N., de Oliveira, K.M., Dias, M.G.B., 2006. Software maintenance ontology. In: Calero, C., Ruiz, F., Piattini, M. (Eds.), Ontologies for Software Engineering and Software Technology. Springer, Berlin, pp. 153–173.

ANSI, 1989. Information Resource Dictionary System. American National Standards Institute.

Aßmann, U., Zschaler, S., Wagner, G., 2006. Ontologies, meta-models, and the model-driven paradigm. In: Calero, C., Ruiz, F., Piattini, M. (Eds.), Ontologies for Software Engineering and Software Technology. Springer, Berlin, pp. 249–273.

Atkinson, C., 1997. Metamodelling for distributed object environments. In: Procs First International Enterprise Distributed Object Computing Workshop (EDOC'97) Brisbane, Australia.

Atkinson, A., 1998. Supporting and applying the UML conceptual framework. In: Bézivin, J., Muller, P.-A. (Eds.), The Unified Modeling Language: Beyond the Notation, LNCS 1618. Springer-Verlag, Berlin, pp. 21–36.

Atkinson, C., Kühne, T., 2002. Rearchitecting the UML infrastructure. ACM Transactions on Modeling and Computer Simulation 12 (4), 290–321.

Atkinson, C., Kühne, T., 2003. Model-driven development: a metamodelling foundation. IEEE Software 20 (5), 36–41.

Atkinson, C., Gutheil, M., Kiko, K., 2006. On the relationship of ontologies and models. In: Meta-Modelling and Ontologies. Proceedings of the 2nd Workshop on Meta-Modelling, WoMM 2006 LNI Volume P-96, pp. 47–60.

Bertoa, M.F., Vallecillo, A., Garcia, F., 2006. An ontology for software measurement. In: Calero, C., Ruiz, F., Piattini, M. (Eds.), Ontologies for Software Engineering and Software Technology. Springer, Berlin, pp. 175–196.

Bertoa, M.F. and Vallecillo, A., 2010. Quality attributes for software metamodels, Proceedings of the Workshop on Quantitative Approaches to Object-Oriented Software Engineering (QAOOSE 2010), Malaga, Spain, 2 July 2010.

Beydoun, G., Low, G., Tran, N., Henderson-Sellers, B., 2005. Preliminary basis for an ontology-based methodological approach for multi-agent systems. In: Akoka, J., Liddle, S.W., Song, L.-Y., Bertolotto, M., Comyn-Wattiau, I., vanden Heuvel, W.-J., Kolp, M., Trujillo, J., Kop, C., Mayr, H.C. (Eds.), Perspectives in Conceptual Modeling: ER2005 Workshops CAOIS, BP-UML, CoMoGIS, eCOMO and QoIS, Klagenfurt, Austria, October 24–28, 2005. Proceedings, LNCS 3770. Springer-Verlag, Berlin, pp. 131–140.

Beydoun, G., Gonzalez-Perez, C., Henderson-Sellers, B., Low, G.C., 2006a. Developing and evaluating a generic metamodel for MAS work products. In: Garcia, A., Choren, R., Lucena, C., Giorgini, P., Holvoet, T., Romanovsky, A. (Eds.), Software Engineering for Multi-Agent Systems IV: Research Issues and Practical Applications, LNCS 3914. Springer-Verlag, Berlin, pp. 126–142.

Beydoun, G., Tran, N., Low, G., Henderson-Sellers, B., 2006b. Foundations of ontology-based methodologies for multi-agent systems. In: Kolp, M., Bresciani, P., Henderson-Sellers, B., Winikoff, M. (Eds.), LNCS 3529. Springer-Verlag, Berlin, pp. 111–123.

Beydoun, G., Krishna, A.K., Ghose, A., Low, G.C., 2008. Towards ontology-based MAS methodologies: ontology based early requirements. In: Barry, C., Lang, M., Wojtkowski, W., Wojtkowski, G., Wrycza, S., Zupancic, J. (Eds.), The Inter-Networked World: ISD Theory, Practice, and Education. Springer-Verlag, New York.

Bézivin, J., 2005a. On the unification power of models. Software and Systems Modelling 4, 171–188.

Bézivin, J., 2005b. Model engineering: from principles to platform, In: WIT-Kolloquium, Technical University of Vienna, March, 15, 2005. Available from: http://wit.tuwien.ac.at/events/bezivin/index.html (accessed 25.05.10).

Bézivin, J., Devedzic, V., Djuric, D., Favreau, J.-M., Gasevic, D., Joualt, F., 2005. An M3-neutral infrastructure for bridging model engineering and ontology engineering. In: Konstansas, D., Bourrières, J.-P., Léonard, M., Boudjlida, N. (Eds.), Interoperability of Enterprise Software and Applications. Springer-Verlag, Berlin, pp. 159–171.

Brinkkemper, S., Lyytinen, R., Welke, R. (Eds.), 1996. Method Engineering: Principles of Method Construction and Tool support. Chapman & Hall, London.

Bunge, M., 1977. Treatise on Basic Philosophy: vol. 3: Ontology I: The Furniture of the World. Reidel, Boston.

Bunge, M., 1979. Treatise on Basic Philosophy: vol. 4: Ontology II: A World of Systems. Reidel, Boston.

Calero, C., Piattini, M., 2006. An ontological approach to SQL:2003. In: Calero, C., Ruiz, F., Piattini, M. (Eds.), Ontologies for Software Engineering and Software Technology. Springer, Berlin, pp. 197–215.

Calero, C., Ruiz, F., Piattini, M., 2006. Ontologies for Software Engineering and Software Technology. Springer, Berlin.

Corcho, O., Fernandez-Lopez, M., Gomez-Perez, A., 2006. Ontological engineering: principles, methods, tools and languages. In: Calero, C., Ruiz, F., Piattini, M. (Eds.), Ontologies for Software Engineering and Software Technology. Springer, Berlin, pp. 1–48.

Cranefield, S., Purvis, M., Nowostawski, M., 2000. Is it an ontology or an abstract syntax? Modelling objects, knowledge and agent messages, In: Proceedings of the Workshop on Applications of Ontologies and Problem-Solving Methods, 14th European Conference on Artificial Intelligence (ECAI) http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.107.8964&rep=rep1&type=pdf (accessed 25.05.10).

de Oliveira, K.M., Villela, K., Regina Rocha, A., Horta Travassos, G., 2006. Use of ontologies in software development environments. In: Calero, C., Ruiz, F., Piattini, M. (Eds.), Ontologies for Software Engineering and Software Technology. Springer-Verlag, Berlin, pp. 275–309.

Degen, W., Heller, B., Herre, H., Smith, B., 2001. GOL: Towards an axiomatized upper level ontology. Paper presented at the Procs. FOIS'01, Ogunquit, Maine, USA.

Devedzic, V., 2002. Understanding ontological engineering. Communications of ACM 45 (4), 136–144.

Dillon, T.S., Chang, E., Wongthongtham, P., 2008. Ontology-based software engineering – software engineering 2.0. In: Procs. 19th Australian Software Engineering Conference ASWEC. 2008 Perth, Australia.

Ermolayev, V., Keberle, N., Matzke, W.-E., 2008. An upper level ontological model for engineering design performance domain. In: Li, Q. (Ed.), ER 2008, LNCS 5231. Springer-Verlag, Berlin, pp. 98–113.

Evermann, J., 2005. Thinking ontologically: conceptual vs. design models in UML. In: Green, P., Rosemann, M. (Eds.), Business Systems Analysis with Ontologies. IGI Group Publishing, Hershey, PA, USA, pp. 82–104.

Falbo, R.A., Guizzardi, G., Duarte, K.C., 2002. An ontological approach to domain engineering. In: Procs. 14th Int. Conf. on Software Eng. and Knowledge Eng. (SEKE), Ischia, Italy.

Falbo, R.A., Natali, A.C.C., Mian, P.G., Bertollo, G., Ruy, F.B., 2003. ODE: Ontology-based software Development Environment. In: Paper presented at the Proc. IX Argentine Congress on Computer Science, La Plata, Argentina.

Falbo, R.A., Ruy, F.B., Moro, R.D., 2005. Using ontologies to add semantics to a software engineering environment. In: Procs. SEKE 2005, Skokie, IL, USA.

Falkenberg, E., Hesse, W., Lindgreen, P., Nilsson, B.E., Oei, J.L.H., Rolland, C., Stamper, R.K., van Assche, F.J.M., Verrijn-Stuart, A.A., Voss, K. (Eds.), 1998. FRISCO – A Frame-work of Information System Concepts – The FRISCO Report, IFIP WG8.1 Task Group FRISCO, Available from: http://www.mathematik.uni-marburg.de/~hesse/papers/fri-full.pdf (accessed 25.05.10).

Favre, J.-M., 2004. Foundations of meta-pyramids: languages vs. metamodels. Episode II. Story of Thotus the Baboon, Procs. Dagstuhl Seminar 04101 "Language Engineering for Model-Driven Software Development.

Favre, J.-M., 2005. Foundations of Model (Driven) (Reverse) Engineering: Models. Episode I: Stories of The Fidus Papyrus and of The Solarus. In: Bézivin, J., Heckel, R. (Eds.), Language Engineering for Model-Driven Software Development, 29. February – 5. March 2004. Dagstuhl Seminar Proceedings 04101, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI). Schloss Dagstuhl, Germany.

Favre, J.-M., 2006. Megamodelling and etymology. A story of words: from MED to MDE via MODEL in five millenniums. In: Cordy, J.R., Lämmel, R., Winter, A. (Eds.), Transformation Techniques in Software Engineering, 17–22 April 2005. Dagstuhl Seminar Proceedings 05161, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI). Schloss Dagstuhl, Germany 2006.

Favre, J.-M., Gašević, D., Lämmel, R., Winter, A., 2007. 3rd International Workshop on Metamodels, Schemas, Grammars and Ontologies. In: Kühne, T. (Ed.), Models in Software Engineering (MoDELS 2006 Workshops), LNCS 4364. Springer-Verlag, Berlin, pp. 52–55.

Fensel, D., 2004. Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, 2nd ed. Springer-Verlag, Berlin.

Firesmith, D., Henderson-Sellers, B., Graham, I., 1997. OPEN Modeling Language (OML) Reference Manual, SIGS Books, New York. Cambridge University Press, New York, 1998.

Flatscher, R.G., 2002. Metamodeling in EIA/CDIF – meta-metamodel and metamodels. ACM Transactions Modeling and Computer Simulation 12 (4), 322–342.

Gašević, D., Kaviani, N., Hatala, M., 2007. On metamodeling in megamodels. In: Engels, G. (Ed.), MoDELS 2007, LNCS 4735. Springer-Verlag, Berlin, pp. 91–105.

Gomez-Perez, A., Fernandez-Lopez, M., Corcho, O., 2004. Ontological Engineering. Springer-Verlag, London.

Gonzalez-Perez, C., 2005. Tools for an extended object modelling environment. In: Procs. 10th IEEE International Conference on Engineering of Complex Computer Systems. IEEE Computer Society, Washington, DC, pp. 20–23.

Gonzalez-Perez, C., 2006. email, Personal communication, 13 December 2006.

Gonzalez-Perez, C., Henderson-Sellers, B., 2006a. An ontology for software development methodologies and endeavours. In: Calero, C., Ruiz, F., Piattini, M. (Eds.), Ontologies in Software Engineering and Software Technology. Springer-Verlag, Berlin, pp. 123–152.

Gonzalez-Perez, C., Henderson-Sellers, B., 2006b. A powertype-based metamodelling framework. Software and Systems Modeling 5 (1), 72–90.

Gonzalez-Perez, C., Henderson-Sellers, B., 2007. Modelling software development methodologies: a conceptual foundation. Journal of Systems Software 80 (11), 1778–1796.

Gonzalez-Perez, C., Henderson-Sellers, B., 2008. Metamodelling for Software Engineering. J. Wiley & Sons, Chichester.

Green, P., Rosemann, M., 2005a. Business Systems Analysis with Ontologies. IGI Group Publishing, Hershey, PA.

Green, P., Rosemann, M., 2005b. Ontological analysis of business systems analysis techniques: experiences and proposals for an enhanced methodology. In: Green, P., Rosemann, M. (Eds.), Business Systems Analysis with Ontologies. IGI Group Publishing, Hershey, PA, pp. 1–27.
Gruber, T.R., 1993. A translation approach to portable ontology specifications. Knowledge Acquisition 5, 199–220.
Guarino, N., 1998. Formal ontology and information systems. In: Procs. Int. Conf. on Formal Ontology in Information Systems – FOIS'98, Trento, Italy.
Guarino, N., 2008. Ontology-driven conceptual modeling: the first fifteen years, presentation at Dagstuhl Seminar on Conceptual Modelling. (preprint on conference website: http://drops.dagstuhl.de/opus/volltexte/2008/1598).
Guizzardi, G., 2005. Ontological Foundations for Structural Conceptual Models. Enschede, The Netherlands.
Guizzardi, G., 2007. On ontology, ontologies, conceptualizations, modeling languages, and (meta)models. In: Frontiers in Artificial Intelligence and Applications Volume 155. Proceedings of the 2007 Conference on Databases and Information Systems IV: Selected Papers from the Seventh International Baltic Conference DB&IS2006. IOS Press, Amsterdam, pp. 18–39.
Guizzardi, G., Wagner, G., 2005a. On the ontological foundations of agent concepts. In: Bresciani, P., Giorgini, P., Henderson-Sellers, B., Low, G., Winikoff, M. (Eds.), Agent-Oriented Information Systems II, LNCS 3508. Springer-Verlag, Berlin, pp. 113–128.
Guizzardi, G., Wagner, G., 2005b. Some applications of a Unified Foundational Ontology in business modeling. In: Green, P., Rosemann, M. (Eds.), Business Systems Analysis with Ontologies. IGI Group Publishing, Hershey, PA, pp. 345–367.
Guizzardi, R.S.S., Guizzardi, G., Perini, A., Mylopoulos, J., 2007. Towards an ontological account of agent-oriented goals. In: Software Engineering for Multi-Agent Systems V, LNCS 4408. Springer-Verlag, Berlin, pp. 148–164.
Happel, H.-J., Seedorf, S., 2006. Applications of ontologies in software engineering. In: Procs.2nd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2006).
Henderson-Sellers, B., 2003. Method engineering for OO systems development. Communications of ACM 46 (10), 73–78.
Henderson-Sellers, B., 2006. Method engineering: theory and practice. In: Karagiannis, D., Mayr, H.C. (Eds.), Information Systems Technology and its Applications. 5th International Conference ISTA 2006, LNI P-84. Gesellschaft für Informatik, Bonn, pp. 13–23.
Henderson-Sellers, B., 2007. On the challenges of correctly using metamodels in method engineering. In: Fujita, H., Pisanelli, D. (Eds.), Keynote paper in New Trends in Software Methodologies, Tools and Techniques. Proceedings of the sixth SoMeT_07. IOS Press, pp. 3–35.
Henderson-Sellers, B., Gonzalez-Perez, C., 2005. Connecting powertypes and stereotypes. Journal of Object Technology 4 (7), 83–96.
Henderson-Sellers, B., Graham, I.M. with additional input from Atkinson, C., Bézivin, J., Constantine, L.L., Dué, R., Duke, R., Firesmith, D., Low, G., McKim, J., Mehandjiska-Stavrova, D., Meyer, B., Odell, J.J., Page-Jones, M., Reenskaug, T., Selic, B., Simons, A.J.H., Swatman, P., Winder, R., 1996. OPEN: toward method convergence?, IEEE Computer 29(4), 86–89.
Henderson-Sellers, B., Unhelkar, B., 2000. OPEN Modeling with UML. Addison-Wesley.
Hesse, W., 2006. More matters on (meta-)modeling: remarks on Thomas Kühne's "matters". Software and Systems Modeling 5 (4), 387–394.
Hesse, W., 2008a. Engineers discovering the "real world" – from model-driven to ontology-based software engineering. In: Kaschek, R., Kop, C., Steinberger, C., Fliedl, G. (Eds.), UNISCON 2008, LNBIP 5. Springer-Verlag, Berlin, pp. 136–147.
Hesse, W., 2008b. From conceptual models to ontologies. Dagstuhl Seminar on Conceptual Modelling.
Höfferer, P., 2007. Achieving business process model interoperability using metamodels and ontologies. In: Procs. 15th European Conf. on Information Systems (ECIS 2007), pp. 1620–1631.
Holten, R., Dreiling, A., Becker, J., 2005. Ontology-driven method engineering for information systems development. In: Green, P., Rosemann, M. (Eds.), Business Systems Analysis with Ontologies. IGI Group Publishing, Hershey, PA, USA, pp. 174–217.
ISO/IEC, 2007. ISO/IEC 24744. Software Engineering – Metamodel for Development Methodologies. ISO, Geneva.
Jackson, M., 2009. Some notes on models and modelling. In: Borgida, A.T., et al. (Eds.), Mylopoulos Festschrift, LNCS 5600. Springer-Verlag, Berlin, pp. 68–81.
Johnson-Laird, P.N., 1983. Mental Models: Towards a Cognitive Science of Language, Inference and Consciousness. Harvard University Press, Cambridge, Mass., USA.
Jurisica, I., Mylopoulos, J., Yu, E., 1999. Using ontologies for knowledge management: an information systems perspective. In: Procs 62nd Annual Meeting of the American Society for Information Science (ASIS99), pp. 482–496.
Karagiannis, D., Fill, H.-G., Höfferer, P., Nemetz, M., 2008. Metamodelling: some application areas in information systems. In: Kaschek, R., Kop, C., Steinberger, C., Fliedl, G. (Eds.), UNISCON 2008, LNBIP 5. Springer-Verlag, Berlin, pp. 175–188.
Kazmierczak, E., Milton, S., 2005. Using a common-sense realistic ontology: making data models better map the world. In: Green, P., Rosemann, M. (Eds.), Business Systems Analysis with Ontologies. IGI Group Publishing, Hershey, PA, USA, pp. 218–248.
Kühne, T., 2006a. Matters of (meta-)modeling. Software and Systems Modelling 5 (4), 369–385.
Kühne, T., 2006b. Clarifying matters of (meta-) modeling: an author's reply. Software and Systems Modelling 5 (4), 395–401.
Laarman, A., Kurtev, I., 2010. Ontological metamodelling with explicit instantiation. In: van den Brand, M., Gašević, G., Gray, J. (Eds.), SLE 2009, LNCS 5969. Springer-Verlag, Berlin, pp. 174–183.
Lenat, D.B., 2005. Applied ontology issues. Applied Ontology 1 (1), 9–12.
Ludewig, J., 2003. Models in software engineering – an introduction. Software and Systems Modelling 2, 5–14.
Menzel, C., 2003. Reference ontologies – application ontologies: either/or or both/and? Available from: http://www.bioontology.org/wiki/images/d/d9/Menzel (accessed 26.05.10).
Muller, P.-A., Fondement, F., Baudry, B., 2009. Modeling modeling. In: Schürr, A., Selic, B. (Eds.), MODELS 2009, LNCS 5795. Springer-Verlag, Berlin, pp. 2–16.
Neches, R., Fikes, R.E., Finin, T., Gruber, T., Patil, R., Senator, T., Swartout, W.R., 1996. Enabling technology for knowledge sharing. AI Magazine 12 (3), 36–56.
Nirenburg, S., 2004. Ontology Tutorial, Available from: http://ilit.umbc.edu/Ontology-tutorial-content.pdf (accessed 25.05.10).
Nodine, M.H., Unruh, A., 1998. Facilitating open communication in agent systems: the InfoSleuth infrastructure. In: Singh, M.P., Rao, A., Wooldridge, M.J. (Eds.), Intelligent Agents IV Agent Theories, Architectures, and Languages 4th International Workshop, ATAL'97 Providence, Rhode Island, USA, July 24-26, 1997 Proceedings LNCS 1365. Springer-Verlag, Berlin, pp. 281–295.
Noy, N.F., McGuinness, D.L., 2001. Ontology development 101: a guide to creating your first ontology, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880.
Odell, J., 1994. Power types. Journal of Object-Oriented Programming 7 (2), 8–12.
OMG, 1999. OMG Unified Modeling Language Specification, Version 1.3, June 1999.
OMG, 2003. MDA Guide Version 1.0.1, omg/2003-06-01.
OMG, 2005a. OMG: Unified Modeling Language Superstructure, Version 2.0, formal/05-07-04.
OMG, 2005b. Ontology definition metamodel, ad/2005-08-01.
Opdahl, A., Henderson-Sellers, B., 2000. Evaluating and improving OO modelling languages using the BWW-model. In: Dampney, C.N.G. (Ed.), Procs. of the Information Systems Foundations Workshop – Ontology, Semiotics and Practice 1999. Lighthouse Press, Macquarie University, Sydney, pp. 31–38.
Opdahl, A., Henderson-Sellers, B., 2002. Ontological evaluation of the UML using the Bunge–Wand–Weber model. Software and Systems Modelling 1 (1), 43–67.
Opdahl, A.L., Henderson-Sellers, B., Barbier, F., 2001. Ontological analysis of whole-part relationships in OO models. Information and Software Technology 43 (6), 387–399.
Pastor, O., España, S., Gonzalez, A., 2008. An ontological-based approach to analyze software production methods. In: Kaschek, R., Kop, C., Steinberger, C., Fliedl, G. (Eds.), UNISCON 2008, LNBIP 5. Springer-Verlag, Berlin, pp. 258–270.
Pidcock, W., 2009, What are the differences between a vocabulary, a taxonomy, a thesaurus, an ontology, and a meta-model? Available from: http://infogrid.org/wiki/Reference/PidcockArticle (accessed 6.09.10).
Recker, J., Rosemann, M., Green, P., Indulska, M., 2007. Exending the scope of representation theory: a review and proposed research model. In: Hart, D.N., Gregor, S.D. (Eds.), Information Systems Foundations: Theory, Representation and Reality. ANU E Press, Canberra, pp. 93–114.
Rilling, J., Zhang, Y., Meng, W.J., Witte, R., Haarslev, V., Charland, P., 2007. A unified ontology-based process model for software maintenance and comprehension. In: Kühne, T. (Ed.), Models in Software Engineering, LNCS 4364. Springer-Verlag, Berlin, pp. 56–65.
Ruiz, F., Hilera, J.R., 2006. Using ontologies in software engineering and technology. In: Calero, C., Ruiz, F., Piattini, M. (Eds.), Ontologies for Software Engineering and Software Technology. Springer-Verlag, Berlin, pp. 49–102.
Saeki, M., Kaiya, H., 2007. On relationships among models, meta models and ontologies. In: Procs. 6th OOPSLA Workshop on Domain-Specific Modeling.
Seidewitz, E., 2003. What models mean. IEEE Software 20, 26–32.
Solberg, A., France, R., Reddy, R., 2005. Navigating the metamuddle. In: Proceedings of the 4th Workshop in Software Model Engineering Montego.
Tran, Q. -N. N., Low, G., 2008. MOBMAS: A methodology for ontology-based multi-agent systems development. Information and Software Technology 50 (7–8), 697–722.
Tran, Q.-N.N., Low, G., Beydoun, G., 2006. A methodological framework for ontology centric oriented software engineering. International Journal of Computer Science and Engineering 21 (**2**), 117–132.
Uschold, M., 2005. An ontology research pipeline. Applied Ontology 1, 13–16.
van de Weerd, I., de Weerd, S., Brinkkemper, S., 2007. Developing a reference method for game production by method comparison. In: Ralyté, J., Brinkkemper, S., Henderson-Sellers, B. (Eds.), Situational Method Engineering: Fundamentals and Experiences. Proceedings of the IFIP WG 8.1 Working Conference. 12–14 September 2007, Geneva, Switzerland IFIP Series, vol. 244. Springer, Berlin, pp. 313–327.
Van Heijst, G., Schereiber, A.T., Wielinga, B.J., 1997. Using explicit ontologies in KBS development. International Journal of Human and Computer Studies 46 (2/3), 293–310.
Wand, Y., 1996. Ontology as a foundation for meta-modelling and method engineering. Information and Software Technology 38, 281–287.
Wand, Y., Weber, R., 1988. An ontological analysis of some fundamental information systems concepts. In: Proceedings of the Ninth International Conference on Information Systems, Minneapolis, 30 November to 3 December 1988.
Wand, Y., Weber, R., 1990. An ontological model of an information system. IEEE Transactions on Software Engineering 16 (11), 1282–1292.
Wand, Y., Weber, R., 1993. On the ontological expressiveness of information systems analysis and design grammars. Journal of Information Systems 3, 217–237.

Wand, Y., Weber, R., 1995. On the deep structure of information systems. Information Systems Journal 5 (3), 203–223.
Wand, Y., Weber, R., 2005. Introduction: setting the scene. In: Green, P., Rosemann, M. (Eds.), Business Systems Analysis with Ontologies. IGI Group Publishing, Hershey, PA, USA, pp. xii–xv.
Warmer, J., Kleppe, A., 1999. The Object Constraint Language. Precise Modeling with UML. Addison-Wesley.
Welty, C., Guarino, N., 2001. Supporting ontological analysis of taxonomic relationships. Data and Knowledge Engineering 39 (1), 51–74.
Whitmire, S.A., 1997. Object Oriented Design Measurement. J. Wiley and Sons, Inc.
Wyssusek, B., Klaus, H., 2005. Ontological foundations of information systems analysis and design: extending the scope of the discussion. In: Green, P., Rosemann, M. (Eds.), Business Systems Analysis with Ontologies. IGI Group Publishing, Hershey, PA, pp. 322–344.
Zhu, S., Feng, J., 2010. Using an ontology to help reason about the information content of data. Journal of Software Engineering and Applications 3, 629–643.

Professor Brian Henderson-Sellers is Director of the Centre for Object Technology Applications and Research within the Centre for Human-Centred Technology Design and Professor of Information Systems at the University of Technology, Sydney (UTS). He is author or editor of over 30 books and is well-known for his work in OO and AO methodologies (MOSES, COMMA, OPEN, OOSPICE, FAME), OO metrics and metamodelling. He has chaired workshops at OOPSLA on method engineering and AOIS on agent-oriented methodologies and was General Chair of the IFIP WG8.1 Working Conference on Method Engineering (Geneva, 2007). He is Co-Editor of the ISO/IEC 24744 International Standard, is Editor of the International Journal of Agent-Oriented Software Engineering and on the editorial board of Journal of Object Technology, Software and Systems Modelling and International Journal of Cognitive Informatics and Natural Intelligence. In July 2001, Professor Henderson-Sellers was awarded a Doctor of Science (DSc) from the University of London for his research contributions in object-oriented methodologies and, in 2010, the Consensus IT Professionals Award.