Applying Guarino's Aufbau principle for ontologies to REA-based applications

Abstract

Disagreement about the nature of ontologies inhibits full acceptance of ontologies as valuable information systems design artifacts. This paper advocates that seeing ontologies as M1 models facilitates their use as design artifacts in the information systems design process and supports consistent ontology design by smoothening the ontological embedding of ontologies that increases inter-ontological consistency.

Keywords

Ontology, Model Driven Architecture

Authors

Wim Laurier, Geert Poels

1. Introduction

In contemporary information systems research, 'ontology' is a buzzword despite, or maybe because of, a wide variety of ontology definitions [1] and obscurity about what ontologies can do for us. Although ontologies seem to bring disagreement in the information systems research domain, they have proven their usefulness in other scientific domains. In biology, for example, taxonomies (i.e. a weaker form of ontologies[2]) provide biologists with a unified nomenclature for determining and classifying animals. Although the taxonomy of life is not complete and stable yet, it is a valuable instrument in scientific communication including taxonomy redesign.

A mature ontology application in chemistry and physics is the periodic table of elements that is also known as Mendeleev's table. This table identifies all existing chemical elements (i.e. kinds of atoms) and arranges them according to their characteristics. The position of each element in the table characterizes its behavior in the chemical reactions that construct molecules, which are stable aggregates of atoms. The initial table was presented by Dimitri Mendeleev and the contemporary design was obtained after numerous redesigns (e.g. by Marie Curie) that reflected knowledge increments. One of the most interesting characteristics of this ontology is that it contributed to its own construction by identifying undiscovered elements as gaps in the table. Another interesting characteristic is that the periodic table of elements is built upon a theory (i.e. Mandlung's Aufbau principle) that describes how the elements differ from each other and how these differences evoke element interactions.

Both the information systems research and business domain do not have such a *"formal representation of a shared conceptualization"* [3] nor do they have an underlying construction theory like Mandlung's Aufbau principle. For the information systems research domain, Wand and Weber

[4] suggested an ontology based on Bunge's top-level ontology. For the business domain, Geerts and McCarthy [5] introduced what could be called the 'periodic table of business elements' for which they used Sowa's top-ontology. Meanwhile, Guarino [6] presented a framework for ontology decomposition and reuse that is equivalent with Mandlung's Aufbau principle for chemical elements. As Wand and Weber [4] indicate, real-world (e.g. REA) and information-system (e.g. BWW) ontologies are complementary in the sense that they are both needed to conceive business information systems since the former *'provide a basis for modeling the information systems themselves'* and the latter *'identify the basic things that the information system is ought to be able to model'*. Unfortunately, neither of these ontologies (i.e. REA or BWW) can show the level of acceptance that mature ontology applications like the biological taxonomy and the periodic table of elements have attained.

Although the REA ontology has been accepted in accounting education [7] and BWW is virtually a standard for evaluating the expressiveness of formal information system analysis and design grammars [8; 9; 10; 11; 12], we intend to increase their level of acceptance in other information systems research and business administration domains. Therefore, this paper will state how ontologies can alleviate information system design and how these ontologies fit the model-driven architecture (MDA) paradigm. Section 2 will position ontologies in the MDA triangle, in order to elucidate the technical side of ontologies. Section 3 will show how ontologies relate to each other (e.g. how different kinds of ontologies can be (re)used to construct other ontologies), to clarify the semantic side of ontologies as it is discusses by Guarino. Section 4 proposes a method for the ontology-based design of information systems based on the results of sections 2 and 3. Conclusions and future research are presented in section 5.

2. Ontologies in the MDA triangle

Multiple authors have tried to position ontologies in the MDA triangle. Some position ontologies on M1 and M2 level [1] or create parallel but disjoint ontology and modeling spaces[13]. These theoretical approaches do not facilitate the technological support for ontologies, because they do not necessarily respect the MDA paradigm in which joint meta- or meta-meta-models enable model transformations. Therefore, we follow the pragmatic interpretation that sees all (kinds of) ontologies as models at the M1 level[14; 15; 16]. Consequently, all ontology to ontology and ontology (i.e. descriptive reality or information system model) to M1 model (i.e. prescriptive reality or information system model) mappings can be realized as model transformations.

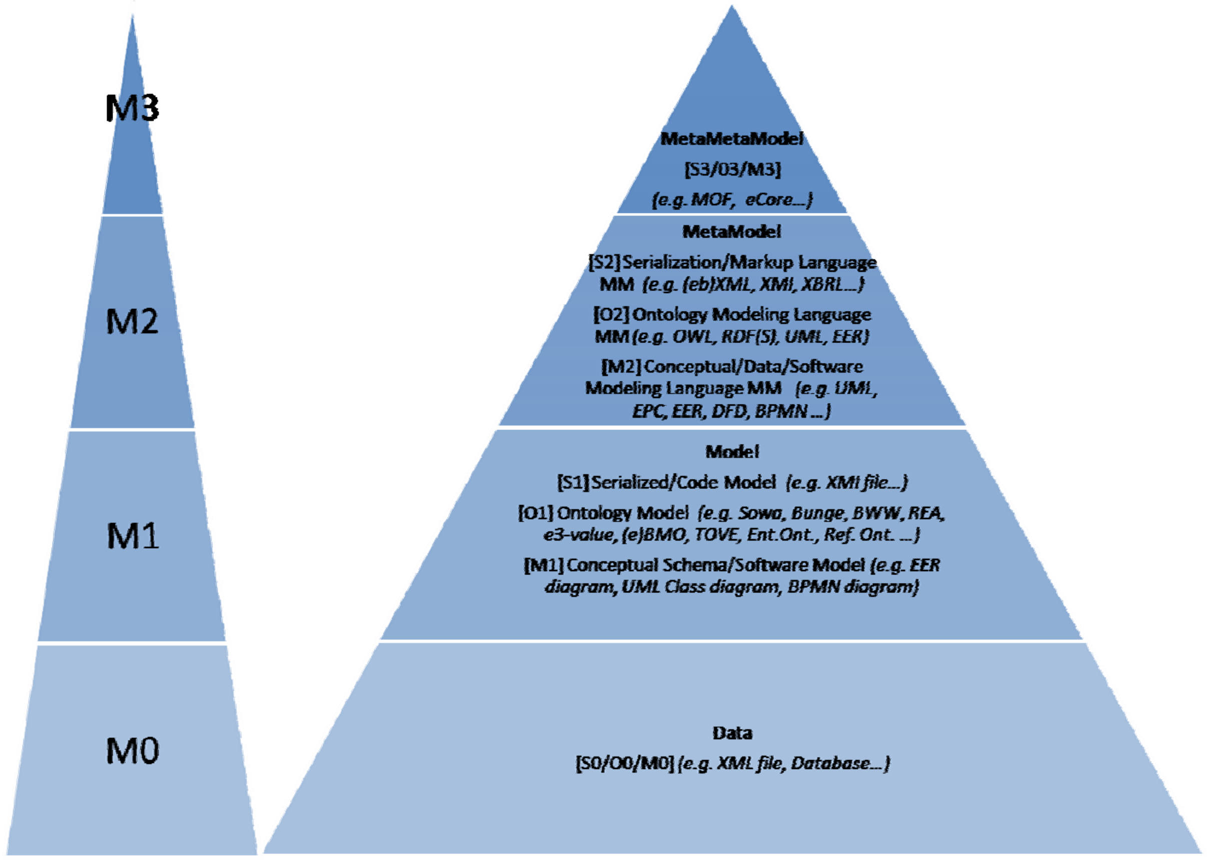
M3
M2
M1
M0
MetaMetaModel
[S3/03/M3]
(e.g. MOF, eCore...)
MetaModel
[S2] Serialization/Markup Language MM (e.g. (eb)XML, XMI, XBRL...)
[O2] Ontology Modeling Language MM (e.g. OWL, RDF(S), UML, EER)
[M2] Conceptual/Data/Software Modeling Language MM (e.g. UML, EPC, EER, DFD, BPMN ...)
Model
[S1] Serialized/Code Model (e.g. XMI file...)
[O1] Ontology Model (e.g. Sowa, Bunge, BWW, REA, e3-value, (e)BMO, TOVE, Ent.Ont., Ref. Ont. ...)
[M1] Conceptual Schema/Software Model (e.g. EER diagram, UML Class diagram, BPMN diagram)
Data
[S0/O0/M0] (e.g. XML file, Database...)

Fig. 1 Ontology position in the MDA triangle

Gasevic et al.[13] distinguish three kinds of modeling spaces (i.e. ontology, model, serialization) for which they define 4 layers in an MDA-triangle-like representation (i.e. O3,O2,O1,O0, M3,M2,M1,M0,S3,S2,S1,S0). Fig. 1 reveals how we see ontologies in the MDA paradigm, following Bézivin et al. [14]. The top of the triangle (i.e. M3) is populated with meta-meta-models that provide an elementary grammar to create the models that prescribe the structure of certain languages. Consequently, we see the O3,M3 and S3 levels as one M3 level with meta-meta-model languages that can be used for the articulation of serialization or markup (S2), ontology (O2) and conceptual, data or software (M2) modeling language meta-models that we see as language specification models at the M2 level. The difference with the Gasevic et al. [13] approach is that we explicitly consider all M2 models equivalent and state that their difference solely lies in the purpose of the language they define (i.e. descriptive representations for the ontology part, prescriptive representations for the conceptual, data or software part and machine-readable representations for the serialization or markup part). Like the M2 meta-models are instances of the M3 meta-meta-models, the M1 models are instances of the M2 meta-models and the M0 data are instances of M1 models. These generalized M1 models can describe ontologies (O1), real-world domains (e.g. business processes), data structures or software architectures/designs (M1) and serialization data or program code (S1).

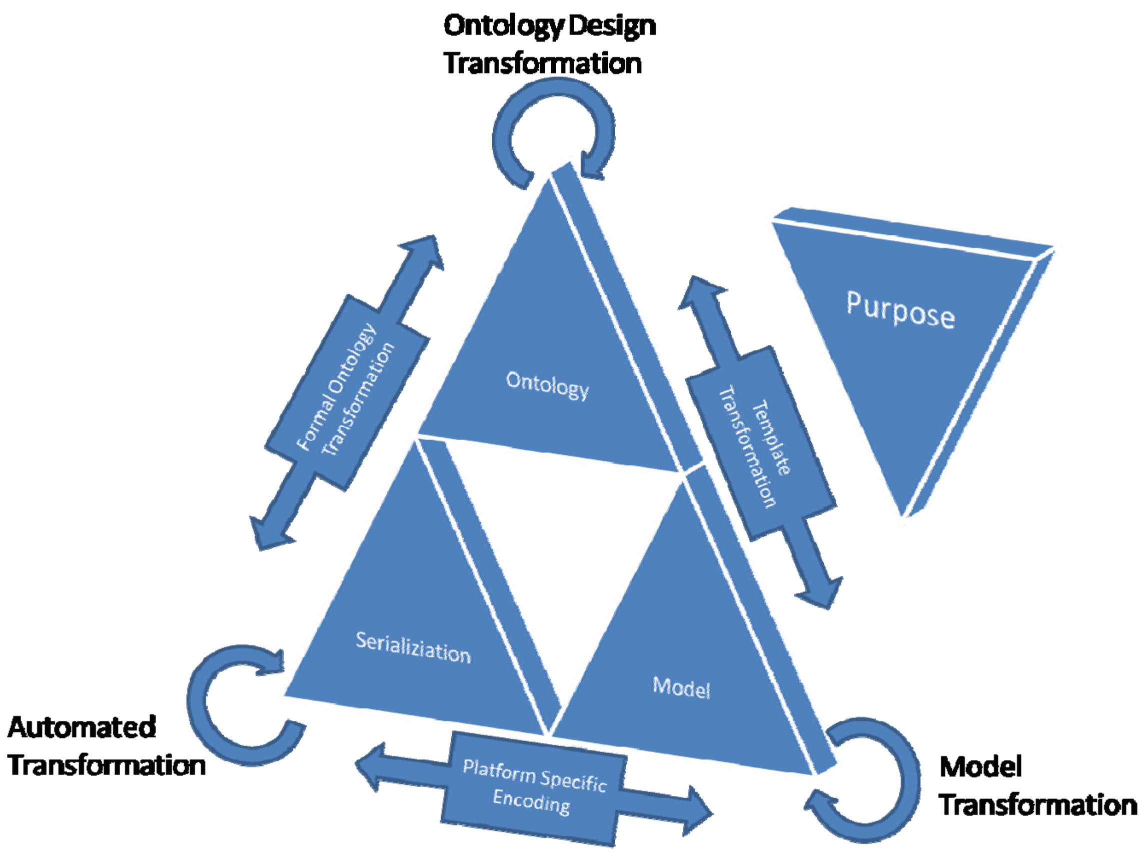

Fig. 2 Model transformations at the M1 level

Fig. 2 summarizes the different transformations at the M1 level, by depicting the three kinds of models (i.e. ontology model (O1), conceptual or software model (M1) and serialized or code model (S1)) and revealing their interrelations. Ontologies relate to each other by means of the ontology design transformations that will be discussed in section 3. Human-readable ontologies can then be turned into machine-readable ontologies with formal ontology transformations (e.g. UML profile versus XMI representations for OWL). Furthermore, ontologies might be used as templates for models in information systems design. These models might then be turned into other models (e.g. PIM to PSM conversions) and for information system implementations, these models need to be converted to model serializations (e.g. XML or program code). Finally, automated transformations convert serializations into other serializations. The ontology to model to serializations transformations will be tackled in section 4.

3. Guarino's Aufbau principle

Guarino [6] distinguishes four kinds of ontologies that describe three levels of detail. These levels enable the reuse of ontological semantics, a consistent buildup of meaning throughout the ontology design process and ontology integration using higher-level ontologies. Meanwhile, the Aufbau principle for ontologies addresses the tradeoff between information specificity and the scope of the domain for which the descriptive information is useful.

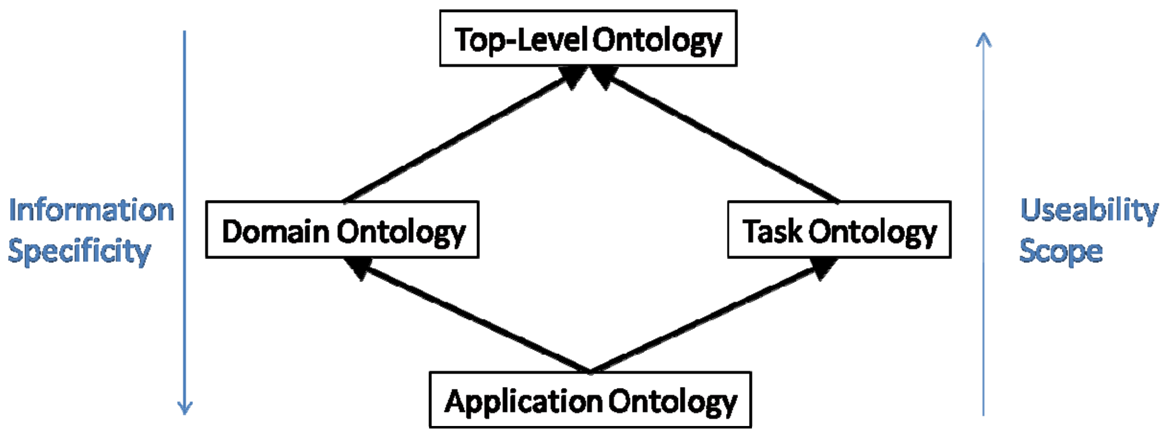

Fig. 3 Guarino's Aufbau principle

Fig. 3 reveals how Guarino[6] sees the relations between top-level, domain, task and application ontologies. *"Top-level ontologies describe very general concepts like space, time, matter, object, event, action, etc. which are independent of a particular problem or domain: it seems therefore reasonable, at least in theory, to have unified top-level ontologies for large communities of users."* [6] This definition clearly shows that top-level ontologies or upper-level ontologies describe a domain with a scope that is as large as possible. Top-level ontologies typically claim to describe the entire world. Describing the world in a limited number of constructs means using extremely generic constructs that provide little specific information.

Domain and task ontologies provide more information than top-level ontologies do. The top-level ontology to domain or task ontology transformation is therefore information adding, while the domain or task ontology to top-level ontology transformation is information abstracting. *"Domain ontologies and task ontologies describe, respectively, the vocabulary related to a generic domain (like medicine or automotive) or a generic task or activity (like diagnosing or selling), by specializing the terms introduced in the top-level ontology."* [6] In this definition Guarino describes a kind of 'good decomposition' of the world into domains, since he requires that every domain or task ontology that describes a specific part of the world describes this part with semantics that are embedded in a higher-level (i.e. top-level) ontology that describes the entire world. In other words, the specialization of top-level constructs in the domain or task ontology provides us with a semantic link between the domain or task and the world the domain or task is part of, by making domain and task ontology constructs specializations of top-level constructs.

The difference between domain and task ontologies is somewhat troublesome since some domain ontologies (e.g. REA [5]) incorporate task or activity constructs (e.g. economic event) in their domain ontology. Therefore it is crucial to determine which parts of the task or activity description belong to the domain ontology and which parts belong to the task ontology. We choose to include all rule and law semantics in the domain ontology and incorporate all configuration semantics in the task ontology. [17] For example, the REA domain ontology describes how each increment event is ought to be coupled with a decrement event and vice versa. Although the domain ontology identifies a causal relationship between an increment and a decrement event, the domain ontology does not reveal which event occurs first since the sequence of events is determined by the application ontology. For this reason the application ontology requires input from the task ontology with construct like sequence, simultaneity, synchronization, parallelism, choice. These configuration constructs are denoted control-flow patterns by van der Aalst et al. [18] More complex configurations for specific domains are discussed[1] in his resource, data and exception handling patterns. These complex configurations discuss task configurations in specific domains, therefore they provide us with an application ontology.

Other patterns (e.g. Fowler's analysis patterns) also provide us with application, domain and task ontologies. *"Application ontologies describe concepts depending both on a particular domain and task, which are often specializations of both the related ontologies. These concepts often correspond to roles played by domain entities while performing a certain activity, like replaceable unit, or spare component."* [6] Guarino also identified a kind of 'good decomposition' for domain and task ontologies into application ontologies since he requires that application ontologies specialize both domain and task ontologies. Consequently, we are able to discern similar tasks in different applications (i.e. same task root, different domain root) and different applications for a domain (i.e. same domain root, different task root).

4. Ontology-based Design

After determining the ontology to ontology transformations in the preceding section, this section will present a framework for information systems design using ontology to model transformations. This framework will distinguish between information system ontologies (e.g. BWW) and real world ontologies (e.g. REA) highlighting their respective use in the different stages of the information system design process.

The design process usually starts with exploring the problem domain. This exploration is usually followed by a problem analysis. The problem analysis can be facilitated using analysis patterns or real world ontologies as analysis templates. The problem analysis is then followed by a design phase in which a solution design is constructed. For information systems, the solution design phase involves conceiving information system representations for problem analysis constructs. The choice of information system constructs can be facilitated using design patterns or information system ontologies. These design patterns and information system ontology constructs are then documented with platform specific encodings (e.g. in Java a property is represented as a data member). These platform specific encodings for design patterns and IS ontology constructs are then used to generate the code for the information system implementation.

In the preceding subsection we identified real world ontologies with analysis patterns and IS ontologies with design patterns. These identifications are oversimplifications since they overlook the fundamental differences between ontologies and patterns. Patterns are empirically validated recurrent solutions for a problem, while ontologies provide theoretically embedded solutions for a problem. However, both ontologies and patterns require the solution to be shared within a community of users (e.g. people that are active in a certain problem domain). The joint characteristic of sharedness makes ontologies and patterns natural allies since the ideal ontology is empirically validated and the ideal pattern is theoretically well-founded. We therefore advocate a meet-in-the-middle between ontologies and patterns.

[1] http://www.workflowpatterns.com/

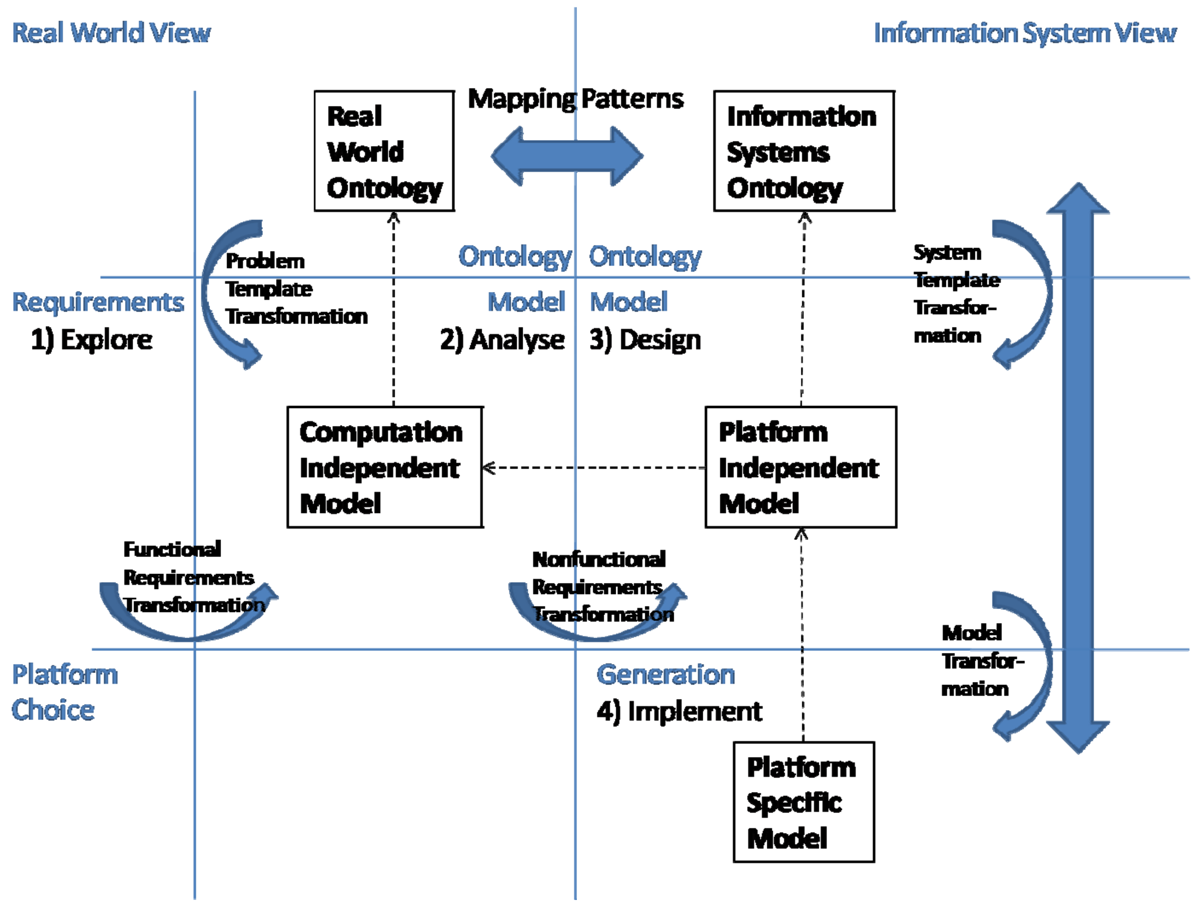
Real World View
Information System View
Real World Ontology
Mapping Patterns
Information Systems Ontology
Problem Template Transformation
Ontology Model
2) Analyse
Ontology Model
3) Design
System Template Transfor-mation
Requirements
1) Explore
Computation Independent Model
Platform Independent Model
Functional Requirements Transformation
Nonfunctional Requirements Transformation
Model Transfor-mation
Platform Choice
Generation
4) Implement
Platform Specific Model

Fig. 4 Stages of the ontology-based information-systems design

Fig. 4 shows that exploring the problem domain involves creating a first outline of the desired system functionalities and a study of the real world process that has to be supported (i.e. system requirements). These requirements can then be mapped with a real world ontology (functional requirements) and the mapping patterns (non-functional requirements). We require the requirements analysis template to be an ontology because the theoretical ground for ontologies mitigates integration conflicts [19] when multiple ontologies need to be integrated in one model, while there is no ground for pattern-crossing construct semantics when integrating analysis patterns.

When selecting the ontology templates for our analysis model (i.e. computation independent model or CIM) we try to meet a maximum of functional system requirements (i.e. maximal ontological completeness) with the selected real-world template ontology or set of real-world template ontologies. The requirements that are not met by this ontology or set of ontologies are input for additional design efforts that conceive analysis patterns that fill the gaps in the CIM and the set of ontologies. These analysis patterns can then be validated theoretically and empirically in order to attain the status of domain, task or application ontology. On the other hand, the original list of functional requirements might contain superfluous requirements that can better be left out. As soon as our CIM reflects a full match between the functional system requirements and a set of template ontologies and analysis patterns, of which the CIM is an instance, we can start the design phase.

The design phase constructs a PIM (i.e. platform independent model) that links CIM constructs to constructs in an information system. Although many real world ontology constructs might have a preferential information system representation (e.g. colors are usually represented as attributes), the representation choices might have serious implications for the characteristics of the resulting information system that are articulated by the non-functional requirements (e.g. representing properties as independent classes might increase retrieval time and slow down the program). Fig. 4 also represents the design choices for a CIM to PIM transformation using an information system ontology and a set of real-world ontology to information-system ontology mappings (e.g. sales can be represented as an independent class or as a method of a resource) as a design template.

The PIM model than represents both the functional requirements represented in the CIM and the design choices made to fulfill the non-functional requirements articulated in information-system ontology constructs. Each of these information-system ontology constructs is the ontological equivalent of a design pattern. Consequently, the IS ontology constructs are documented with platform specific encodings in several programming languages. These platform specific encodings for IS ontology construct then help generating the PSM for the PIM.

5. Conclusions and future research.

In this paper we positioned ontologies in the MDA triangle, specified how ontologies relate to each other according to Guarino and presented a framework that specifies how ontologies can contribute to information systems design. Guarino divided ontologies in four categories in order to increase their (re)usability and inter-ontological consistency using domain and task ontologies as intermediate stages in application ontology design that is based in confirmed top-level ontologies. In this paper, these application ontologies are then classified as real world ontologies that describe the things an information system is ought to represent and information system ontologies that describe information systems. The real world ontologies are then used in ontology to computation independent model transformations that are orchestrated by user defined system requirements. The mapping patterns document the possible mappings between real world ontology constructs and the information system ontology constructs that are used to represent real world ontology constructs in a platform independent model. The platform independent model then represents the analysis in the computation independent model with constructs from the information system ontology. Each of these information system ontology constructs has a specific encoding in a certain platform. Consequently, the information system ontology drives the transformations in model driven architecture. To enable the different kinds of model transformations described in this paper, we started this paper positioning ontology models as M1 models.

In the future we will position the REA domain ontology in Guarino's Aufbau principle for application ontology design using van der Aalst's control flow patterns as a task ontology. Later on, we would like to document business information systems design process with the REA ontology and complementary information system ontologies.

References

[1] U. Assmann, S. Zchaler, and G. Wagner, Ontologies, Meta-Models, and the Model-Driven Paradigm. in: C. Calero, F. Ruiz, and M. Piattini, (Eds.), Ontologies for Software Engineering and Software Technology, 2006.

[2] L. Obrst, Ontologies for semantically interoperable systems, Proceedings of the twelfth international conference on Information and knowledge management, ACM, New Orleans, LA, USA, 2003.

[3] W.N. Borst, Construction of Engineering Ontologies, Centre for Telematica and Information Technology, Enschede, The Netherlands, 1997.

[4] Y. Wand, and R. Weber, An Ontological Model of an Information System. Software Engineering, IEEE Transactions on 16 (1990) 1282-1292.

[5] G.L. Geerts, and W.E. McCarthy, An ontological analysis of the economic primitives of the extended-REA enterprise information architecture. International Journal of Accounting Information Systems 3 (2002) 1-16.

[6] N. Guarino, Formal Ontology and Information Systems, Proceedings of FOIS'98, IOS Press, Trento, Italy, 1998, pp. 3-15.

[7] C.L. Dunn, J.O. Cherrington, and A.S. Hollander, Enterprise information systems : a pattern-based approach, McGraw-Hill/Irwin, Boston, 2005.

[8] P. Green, and M. Rosemann, Integrated process modeling: An ontological evaluation. Information Systems 25 (2000) 73-87.

[9] M. Rosemann, and P. Green, Developing a meta model for the Bunge-Wand-Weber ontological constructs. Information Systems 27 (2002) 75-91.

[10] A. Kayed, and R.M. Colomb, Using BWW model to evaluate building ontologies in CGs formalism. Information Systems 30 (2005) 379-398.

[11] A.L. Opdahl, and B. Henderson-Sellers, A Unified Modelling Language without referential redundancy. Data & Knowledge Engineering 55 (2005) 277-300.

[12] P. Green, M. Rosemann, M. Indulska, and C. Manning, Candidate interoperability standards: An ontological overlap analysis. Data & Knowledge Engineering 62 (2007) 274-291.

[13] D. Gasevic, Model driven architecture and ontology development, Springer, New York, 2006.

[14] J. Bézivin, V. Devedzic, D. Djuric, J.-M. Favreau, D. Gasevic, and F. Jouault, An M3-Neutral Infrastructure for Bridging Model Engineering and Ontology Engineering, Interoperability of Enterprise Software and Applications, 2006, pp. 159-171.

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

[16] C. Atkinson, and T. Kuhne, Model-driven development: a metamodeling foundation. Software, IEEE 20 (2003) 36-41.

[17] M. Rosemann, and W.M.P. van der Aalst, A configurable reference modelling language. Information Systems 32 (2007) 1-23.

[18] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros, Workflow Patterns. Distributed and Parallel Databases 14 (2003) 5-51.

[19] S.R. Rockwell, and W.E. McCarthy, REACH: Automated Database Design Integrating First-Order Theories, Reconstructive Expertise, and Implementation Heuristics for Accounting Information Systems. International Journal of Intelligent Systems in Accounting, Finance & Management 8 (1999) 181-197.