Towards Ontology-Driven Information Systems: Redesign and Formalization of the REA Ontology

Frederik Gailly and Geert Poels

Faculty of Economics and Business Administration, Ghent University
{Frederik.Gailly,Geert.Poels}@UGent.be

Abstract. It is widely recognized that ontologies can be used to support the semantic integration and interoperability of heterogeneous information systems. Resource Event Agent (REA) is a well-known business ontology that was proposed for ontology-driven enterprise system development. However, the current specification is neither sufficiently explicit nor formal, and thus difficult to operationalize for use in ontology-driven business information systems. In this paper REA is redesigned and formalized following a methodology based on the reengineering extension of the METHONTOLOGY framework for ontology development. The redesign is focused on developing a UML representation of REA that improves upon existing representations and that can easily be transformed into a formal representation. The formal representation of REA is developed in OWL. The paper discusses the choices made in redesigning REA and in transforming REA's UML representation into a OWL representation.

Keywords: Business Ontologies, Interoperability, Ontology Engineering, UML, OWL.

1 Introduction

For many years now ontology research has been a growing research domain in Knowledge Engineering and Artificial Intelligence. The most cited definition of an ontology is the definition by Gruber: "an ontology is an explicit specification of a conceptualization" [1, p. 199]. A conceptualization is an intensional semantic structure which encodes the implicit rules constraining the structure of a piece of reality [2]. Ontologies can be used to represent explicitly the semantics of structured and semi-structured information enabling automatic support for maintaining and accessing information [3]. The Gruber definition was modified slightly by Borst [4] who added that the specification must be formal and the conceptualization should be shared. Formal means that a machine must be able to process the specification and shared indicates that the knowledge captured by the ontology is the consensus of a community of experts [5].

Domain ontologies specify a conceptualization of a selected part of reality (i.e. a domain) [6]. They describe the concepts that exist in a domain, the classification of the concepts, the relations between the concepts and their axioms (i.e. basic propositions assumed to be true). For instance, business ontologies (see [7] for an overview) have as 'Universe of Discourse' business, which is "the activity of providing goods and services involving financial, commercial and industrials aspects" (WordNet). Business ontologies have been proposed to support requirements elicitation, modeling and engineering of business applications, enterprise systems and e-collaboration systems [8-11]. For instance, in ontology-driven business modelling, they are used to constrain the contents and structure of business models, thereby helping to identify and organize relevant objects, relationships and other knowledge [12].

The use of ontologies at run-time (i.e. ontology-driven systems instead of ontology-driven development [12]) also offers great potential for business ontologies. Specifically in and between enterprises ontology-driven information systems can be used to create interoperability at different enterprise levels: shop-floor, intra-enterprise and inter-enterprise level. Recently there have been research efforts on how ontologies can be used for information integration [13, 14] and process integration [15-17]. However, real-world applications of business ontologies that explore these opportunities are still scarce.

The Resource Event Agent (REA) ontology [18] is an example business ontology that has been proposed for ontology-driven enterprise systems development. As an 'event ontology' [19], REA focuses on the events occurring within the realm of a company, their participating agents, affected resources, and regulating policies. REA can be used as a reference for modelling a single business cycle (e.g. sales-collection) or a chain of business cycles, connected through resource flows. Applications supported by REA-driven modelling include the design of accounting and operations management systems [20], auditing and internal control (e.g. SOX compliance checking) [21] and conceptual data modelling [19]. REA has been used in a number of international standardization efforts for e-collaboration systems. For instance, REA was the basis for the business process ontology in the UMM business process and information model construction methodology [22], the ECIMF system interoperability enabling methodology [23], and the Open-EDI business transaction ontology which is part of the ISO/IEC 15944-4 standard. REA has further been proposed as a theoretical basis for the reference models that underlie ERP systems [24].

In this paper we investigate REA's potential to be used at run-time (i.e. in ontology-driven information systems). The origin of REA is an accounting data model [25] that has been extended first into an enterprise information architecture and later into a full-scale enterprise ontology. This development followed an ad-hoc process rather than being guided by an Ontology Engineering methodology. The REA developers focused more on the theoretical background of the ontology (events accounting and Micro-Economic theories) than on the representation, formalization and computational correctness of the ontology (although they did perform in [18] an ontological analysis using Sowa's classification [26]). As a consequence, there is no formal representation of REA. Furthermore, the available literature sources on REA (e.g. [20, 21, 27, 28]) present different views on the REA conceptualization of an enterprise and use a variety of different formats (including text, tables and diagrams) to specify the ontology. The lack of a generally accepted conceptualization and a uniform, complete and formal representation of the ontology causes imprecision in its definition and ambiguity in its interpretation. For instance, in [29] we showed that the ontological concepts and the relations between the concepts are not strictly defined and that the ontological axioms, only informally defined in text, are confusing (mixing up types and instances of concepts). There have been attempts to formalize REA (e.g. [30, 31]), but the results obtained were highly dependent on the researchers' subjective interpretation of REA.

As currently REA is not formally represented, not sufficiently explicit, and not based on a demonstrable shared conceptualization, it needs improvement before it can be applied on a wide scale in ontology-driven systems. The aim of this paper is to facilitate the operationalization of REA (i.e. increasing its applicability to be used at run-time) by making it more explicit and formal. We first present the development of a new graphical UML representation of REA that presents a unified and consistent view on the ontology's concepts, relations and axioms. This new conceptual representation is subsequently used as the basis for a more formal representation of REA in OWL. Improving the conceptual and formal representations of REA makes the ontology more understandable, easier to compare to alternative business ontologies, easier to analyse (e.g. for consistency), and more operational (e.g. executable). The paper focuses on representation and formalization aspects, but not on the contents of REA. The new representations will not make REA more accepted per se (in the sense of a having a conceptualization that is widely shared in the business community). However, they do facilitate ontological analysis and the evaluation whether there is agreement upon the ontological definitions and axiomatization.

It is our position that the reengineering of business ontologies to improve their applicability should be guided by proven Ontology Engineering principles and techniques. The two REA reengineering activities presented in this paper are part of a more encompassing business ontology reengineering methodology that we have developed and are currently refining. The proposed reengineering methodology is presented briefly in the next section. Section three presents the new conceptual representation of REA (as a UML class diagram) based on our redesign and synthesis of the currently available representations. Section four then presents the OWL representation of REA and discusses the UML-OWL language mappings employed. Section five ends with conclusions and future work.

2 Reengineering Business Ontologies

An Ontology Engineering methodology can provide guidelines for different purposes. It can prescribe a stepwise ontology development process, provide decision rules to follow during the modelling of the ontology, and address various design, representation and management aspects of the ontology [32]. In [7] we proposed a reengineering methodology for business ontologies. The first part of this section outlines the process, phases and main activities of this methodology. The second part focuses on those activities that are relevant for the reengineering of REA as intended in this paper, i.e. developing an improved conceptual representation of the ontology and formalizing it.

2.1 Business Ontology Reengineering Methodology

The proposed reengineering methodology (see Figure 1) is based on the METHONTOLOGY framework for ontology development [33] which has its roots in Software Engineering and Knowledge Engineering methodologies. METHONTOLOGY is especially useful for our purpose because it has an ontology reengineering extension [34]. The METHONTOLOGY framework defines ontology reengineering as "the process of retrieving and transforming a conceptual model of an existing and implemented ontology into a new, more correct and complete conceptual model, which is reimplemented". Three phases are identified in this reengineering process: reverse engineering, restructuring and forward engineering. These phases provide the core structure of our business ontology reengineering methodology.

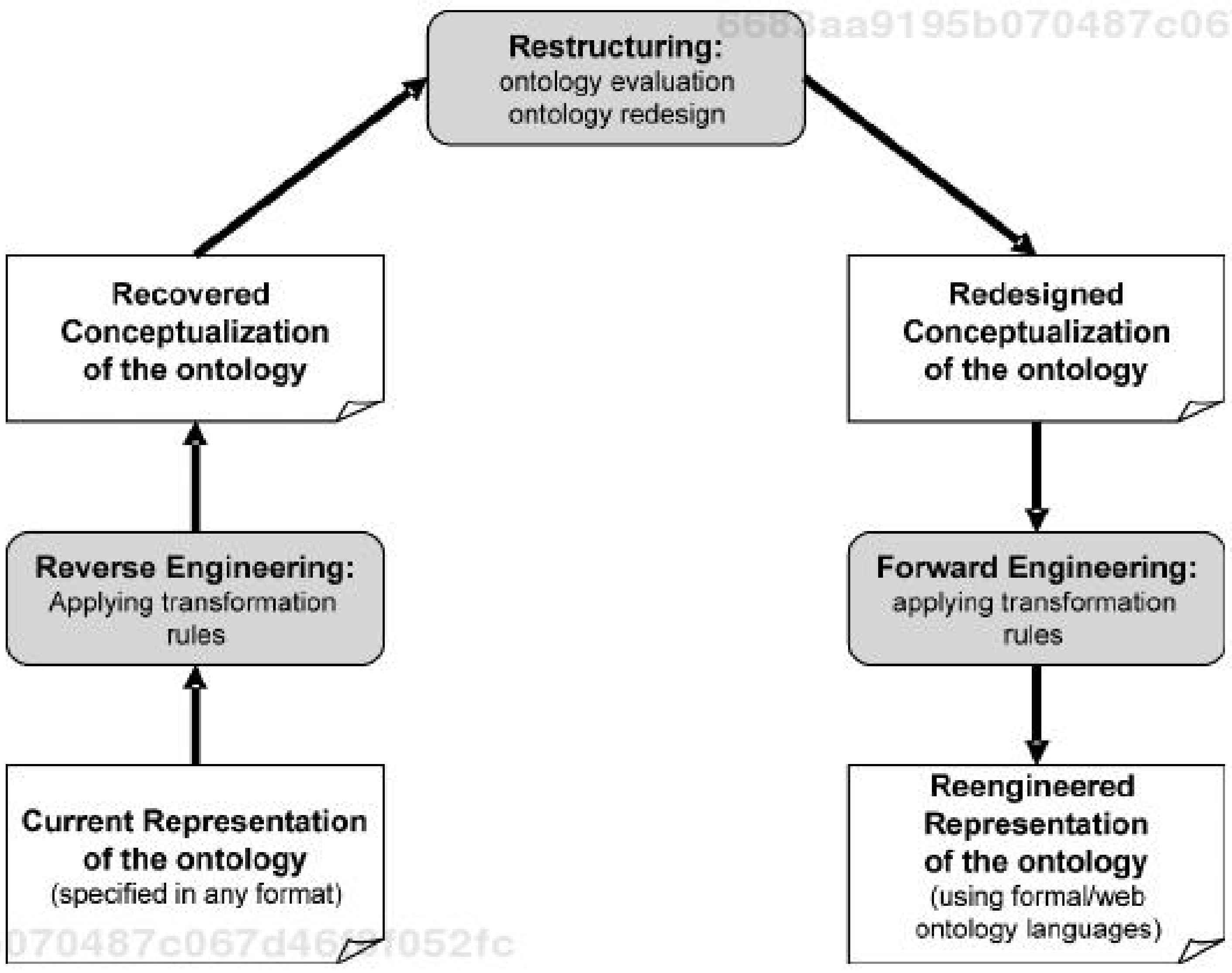
Restructuring:
ontology evaluation
ontology redesign
Recovered Conceptualization of the ontology
Redesigned Conceptualization of the ontology
Reverse Engineering:
Applying transformation rules
Forward Engineering:
applying transformation rules
Current Representation of the ontology
(specified in any format)
Reengineered Representation of the ontology
(using formal/web ontology languages)

Fig. 1. Business Ontology Reengineering Methodology – Process, Activities and Artifacts

The *Reverse Engineering* phase starts from the currently available representation(s) of the ontology as specified in whatever language(s) (i.e. the *Current Representation* artifact). The goal is to recover the original conceptualization specified by the ontology. If only a formal representation of the ontology is available, then this formal representation should be transformed into a conceptual representation by using the right transformation rules for the chosen representation languages. For example, if the ontology is available in OWL, visualization rules that transform OWL into UML can be used to create a graphical representation of the ontology [35]. Sometimes there is neither a formal representation, nor a complete conceptual representation of the ontology (or there exists alternative specifications as with REA). In that case the reverse engineering will mainly be a 'unification' effort, gathering information on the conceptualization selected from the available sources, thereby focusing on the commonalities in the existing representations and underlying interpretations. The result of the reverse engineering activities is referred to as the *Recovered Conceptualization* of the ontology.

During the *Restructuring* phase the recovered conceptualization is evaluated and its specification is redesigned based on the shortcomings identified. The METHONTOLOGY restructuring process distinguishes two distinct sub-phases: analysis and synthesis. During the analysis phase the class hierarchy, the concept definitions and the axioms are evaluated. Studies that compare different ontology evaluation methods can be used for selecting the appropriate analysis method [36, 37]. The synthesis phase then uses the outcome of the analysis to re-specify the conceptualization. Also all changes made to the ontology are recorded.

The restructuring phase results in a *Redesigned Conceptualization* of the ontology. *Forward Engineering* activities transform the conceptual representation of this redesigned conceptualization into a machine readable representation, referred to as the *Reengineered Representation* of the ontology. The transformation rules used depend on the representation format of the redesigned ontology and the target ontology language.

2.2 From Lightweight Ontology to Formal Ontology

REA as it stands can be characterized as a lightweight ontology [38]. To formalize a lightweight ontology a specification is needed that provides a complete view of the conceptualization with as much explicit semantics as possible. Preferably such a specification is developed in a language that can easily be mapped onto an ontology language but at the same time allows for ontology modelling.

Recently, Ontology Engineering researchers have proposed the use of conceptual modelling languages (ER, UML, ORM, …) for modelling ontologies [39-41]. Most conceptual modelling languages propose graphical representations with well-defined syntax and semantics that are close to how humans perceive the world [42]. Specifically in a business context the diagrammatic techniques offered by conceptual modelling languages are known (or can easily be learned) by business domain experts [43]. Conceptual modelling languages therefore provide a more suitable basis for the analysis and refinement of the content of an ontology than the more formal knowledge representation languages as the resulting representation of the ontology will be more natural and hence easier to understand for the domain experts.

REA uses a combination of informal text, table definitions and diagrams. These definitions and diagrams can be found in different sources by different authors. To facilitate the formalization of REA we first developed a representation of REA that unifies these partial (and often imprecise) definitions into a single coherent view with explicit semantics. In case of doubt we referred to the 'official' version of the ontology as described by the REA developers in their most recent papers (i.e. [20, 21]). We developed this new representation in UML (using a class diagram) and refer to it as the *conceptual* representation of REA because its intended users are humans and not machines. UML was chosen as ontology modelling language because it provides a standard and tool-supported notation. Furthermore, the Ontology Definition Metamodel (ODM) request for proposal of the Object Management Group (OMG) has resulted in the proposal of semantically correct mapping rules in both directions between UML and the ontology languages RDF and OWL[44].

The development of the new UML representation of REA corresponds roughly to the reverse engineering and restructuring steps of Figure 1. The currently available representations of REA had to be gathered, analysed and combined to recover to the best possible extent the original conceptualization (i.e. reverse engineering). Unifying the existing definitions, resolving inconsistencies and explicitly representing the recovered semantics in a UML class diagram can be seen as a restructuring activity. The result should however be seen as a redesigned representation of REA rather than a redesigned conceptualization as no fundamental changes to the intended content of the ontology were proposed. The proposed reengineering methodology is intended as an iterative process and in future iterations the new conceptual representation can be used for in-depth analysis of REA's conceptualization of business and subsequent discussion and refinement. In this first iteration, the UML class diagram was primarily meant as a starting point for the formalization of REA.

The purpose of the formalization activity is to map the conceptual representation into a *formal* representation, meaning a machine-readable representation (i.e. the intended users are machines and not humans). The ontology language chosen was OWL because of its wide acceptance as a web ontology language. Also, the availability of ontology tools that support OWL (e.g. Description Logic reasoners) will make it easier to experiment with REA-driven business applications. The OWL representation that results from this forward engineering step corresponds to the reengineered representation artefact in Figure 1.

3 Redesigning REA

REA specifies five basic concepts (Economic **R**esource, Economic **E**vent, Economic **A**gent, Commitment, Contract) in terms of which an enterprise is described. These five concepts (defined in Table 1) are the ontological primitives upon which the other ontological elements (e.g. specialized concepts, typified concepts, relations) are built and in terms of which domain axioms are defined. We also assume these definitions when redesigning the conceptual representation of REA.

Table 1. Definitions of the basic REA concepts

Concept	Definition
Economic Resource	A thing that is scarce and has utility for economic agents and is something users of business applications want to plan, monitor and control.
Economic Agent	An individual or organization capable of having control over economic resources, and transferring or receiving the control to or from other individuals or organizations.
Economic Event	A change in the value of economic resources that are under control of the enterprise.
Commitment	A promise or obligation of economic agents to perform an economic event in the future
Contract	A collection of increment and decrement commitments and terms.

REA further defines and names a number of relations between the basic concepts. For instance, economic resources are associated with the economic events that cause their *inflow* or *outflow* (*stockflow relations*). Economic events that result in resource inflows (e.g. purchases) are paired by economic events that result in resource outflows (e.g. cash disbursements) (*duality relations*). *Participation relations* identify the economic agents involved in economic events. A commitment will eventually be related to an economic event of the specified type by a *fulfilment relation. Reciprocity relations* are analogous to duality relations, but relate commitments instead of economic events.

These and other concept relations are defined informally in text or are depicted as relationships in ER diagrams, UML class diagrams or other graphical formalisms that provide a partial view on the ontology. Implicitly they introduce a number of derived concepts like specializations of basic concepts (e.g. economic events that cause an inflow (*increment* economic events) versus economic events that cause an outflow (*decrement* economic events)) and type images (e.g. economic event type), as well as constraints (e.g. if commitments A and B are linked by reciprocity, and economic events C and D fulfil respectively A and B, then C and D must be linked by duality). Apart from type images (described extensively in [21]) these inferred concepts and constraints are underspecified.

Also a minimal set of ontological axioms is defined, again informally. The definitions of axioms 1 and 2 are literally copied from [20]. Axiom 3 is based on [27] and is more precisely formulated than its equivalent in [20].

- ***Axiom1*** – At least one inflow event and one outflow event exist for each economic resource; conversely inflow and outflow events must affect identifiable resources.
- ***Axiom2*** – All events effecting an outflow must be eventually paired in duality relationships with events effecting an inflow and vice-versa.
- ***Axiom3*** – Each economic event needs to have at least one provide and one receive relationship with an economic agent.

The main problem with these definitions is that it not always clear whether the axioms apply to instances or types. For instance, an enterprise can own an economic resource that it has produced or acquired (e.g. a car) but not sold, used or consumed yet. Clearly the existence of a decrement event for every economic resource under the control of a company is not an axiom. However, following economic rationale (i.e. value creation), we could say that for every type of economic resource there exists at least one type of decrement economic event (e.g. a car manufacturer produces cars to sell them). Axiom 3 further introduces specializations of the participation relation: *provide* and *receive*.

The new conceptual representation that we propose for REA is shown in Figure 2. REA concepts are shown as classes, concept relations are shown as associations or association classes, and axioms are specified by means of multiplicities. For a more detailed and complete account of the shortcomings in the current REA representations and how we resolve them in our UML class diagram, we refer to [7]. The two main improvements are:

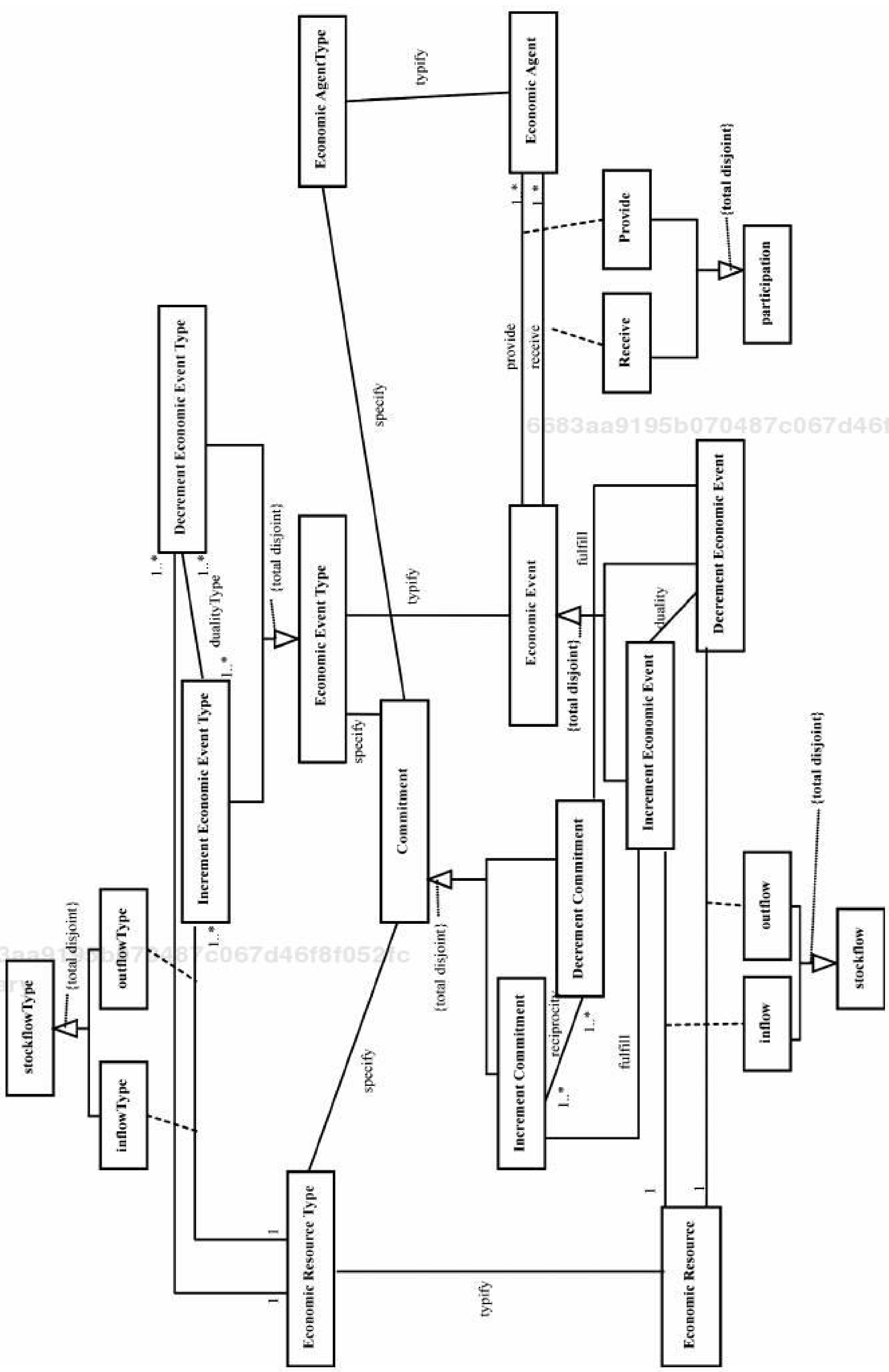

Fig. 2. Conceptual representation of REA as a UML class diagram

- The explicit specification of specializations of basic REA concepts and their type images: Increment Economic Event, Decrement Economic Event, Increment Commitment, Decrement Commitment, Increment Economic Event Type, and Decrement Economic Event Type. Less commonly used in UML is the specialization of association classes (that are used to represent concept relations): inflow and outflow as specializations of the stockflow relation and provide and receive as specializations of the participation relation. These new classifications add formerly implicit semantics to the conceptual representation. The diagram shows, for instance, that inflows relate increment events to resources, outflows relate decrement events to resources, increment events fulfil increment commitments, and decrement events fulfil decrement commitments. Note that Contract is not shown in the diagram, though it can be added by reifying the reciprocity relation. Further, both Contract and Commitment can be typified (again not shown in Figure 2).
- The extensions make it also possible to specify the REA axioms by means of multiplicities. For instance, the first part of axiom 1 can now be stated more precisely by enforcing the participation of every Economic Resource Type object in at least one inflowType link and at least one outflowType link. It is however not required that every Economic Resource object is related to at least one Economic Increment Event and one Economic Decrement Event. Other multiplicities than the ones shown in Figure 2 can be specified, but for the sake of clarity we only included the multiplicities that correspond to the three basic axioms. There is only one exception; the multiplicities shown for the reciprocity relation imply that every increment commitment must be paired with at least one decrement commitment, and vice versa (i.e. the economic reciprocity principle of capitalist market economies that underlies every written or unwritten contract). This example shows that additional domain axioms can easily be integrated into the conceptual representation.

4 Formalizing REA

Different authors have stipulated guidelines for transforming UML class diagrams into a formal representation in an ontology language [39, 40] and vice versa [35]. In the absence of a uniform approach, the recent adoptedOntology Definition Metamodel proposal [44] is used to guide the formalization of REA in OWL. All classes, relationships and constraints presented in the UML class diagram (Figure 2) are transformed into OWL by mapping them to OWL constructs. In most cases these transformations are straightforward but some of them require additional explanation. One of the problems with the ODM specification is that sometimes for the same UML construct different mapping rules can be used and that for some UML constructs no appropriate OWL construct exists. As a result the mapping from UML to OWL depends to some extent on the transformation choices made. The used UML to OWL transformations and corresponding examples are shown in tables 2, 3, 4 and 5. The complete OWL formalization of the conceptual REA representation developed in this paper can be downloaded from http://users.ugent.be/~fgailly/REAontology/.

In Table 2, the UML classes Economic Event (Type), Economic Agent (Type), Economic Resource (Type) and Commitment are mapped onto disjoint OWL classes (i.e. disjoint subclasses of `owl:Thing`). The binary associations and association classes (but not their sub-classes) in Figure 2 are represented by OWL properties (using the ObjectProperty construct). In OWL a property name has a global scope, while in UML the association name scope is limited to the class and subclasses of the class on which it is defined. As a result we have decided to give all associations a unique name. The transformation from UML binary association to OWL properties follows the approach taken in the ODM specification which states that every bidirectional binary association must be translated into a pair of properties where one is inverseOf the other. For an association class the same approach is followed.

Table 2. Transformations UML to OWL (UML class, UML association (class))

UML elements	**OWL elements**
Class	Class, disjointWith

```
<owl:Class rdf:ID="Economic_Agent">
   <owl:disjointWith rdf:resource="#Economic_Agent_Type"/>
   <owl:disjointWith rdf:resource="#Economic_Event"/>
   <owl:disjointWith rdf:resource="#Economic_Resource_Type"/>
   <owl:disjointWith rdf:resource="#Economic_Resource"/>
   <owl:disjointWith rdf:resource="#Economic_Event_Type"/>
   <owl:disjointWith rdf:resource="#Commitment"/>
</owl:Class>
```

Binary association	ObjectProperty, domain, range, inverseOf

```
<owl:ObjectProperty rdf:ID="fulfill">
   <rdfs:domain rdf:resource="#Economic_Event"/>
   <rdfs:range rdf:resource="#Commitment"/>
   <owl:inverseOf rdf:resource="#inverse_of_fulfill"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="inverse_of_fulfill">
   <rdfs:domain rdf:resource="#Commitment"/>
   <rdfs:range rdf:resource="#Economic_Event"/>
   <owl:inverseOf rdf:resource="#fulfill"/>
</owl:ObjectProperty>
```

Association class	ObjectProperty, domain, range, inverseOf

```
<owl:ObjectProperty rdf:ID="stockflow">
   <rdfs:domain rdf:resource="#Economic_Resource"/>
   <rdfs:range rdf:resource="#Economic_Event"/>
   <owl:inverseOf rdf:resource="#inverse_of_stockflow"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="inverse_of_stockflow">
   <rdfs:domain rdf:resource="#Economic_Event"/>
   <rdfs:range rdf:resource="#Economic_Resource"/>
   <owl:inverseOf rdf:resource="#stockflow"/>
</owl:ObjectProperty>
```

UML specializations can be very straightforward transformed into OWL by using the subclass construct for specialized classes (subClassOf) and the subproperty construct for properties (subPropertyOf) for specialized association classes (Table 3 and 4). In UML specialization structures can be declared as being disjoint, meaning each member of a superclass may be a member of no more than one subclass. In OWL this constraint is added by declaring that the subclass must be disjoint with the other subclasses.

Table 3. Transformations UML to OWL (total disjoint subclasses)

UML elements	OWL elements
Total disjoint subclasses	disjointWith, unionOf, subClass

```
<owl:Class rdf:ID="Economic_Event">
   <rdfs:subClassOf>
   <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
   <owl:Class rdf:about="#Decrement_Economic_Event"/>
   <owl:Class rdf:about="#Increment_Economic_Event"/>
   </owl:unionOf>
   </owl:Class>
   </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Decrement_Economic_Event">
   <rdfs:subClassOf rdf:resource="#Economic_Event"/>
   <owl:disjointWith rdf:resource="#Increment_Economic_Event"/>
</owl:Class>
<owl:Class rdf:ID="Increment_Economic_Event">
   <rdfs:subClassOf rdf:resource="#Economic_Event"/>
   <owl:disjointWith rdf:resource="#Decrement_Economic_Event"/>
</owl:Class>
```

Table 4. Transformations UML to OWL (total disjoint subrelations)

UML elements	OWL elements
Total disjoint 'subrelations'	subPropertyOf

```
<owl:ObjectProperty rdf:ID="inflow">
   <rdfs:domain rdf:resource="#Economic_Resource"/>
   <rdfs:range rdf:resource="#Increment_Economic_Event"/>
   <owl:inverseOf rdf:resource="#inverse_of_inflow"/>
   <rdfs:subPropertyOf rdf:resource="#stockflow"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="inverse_of_inflow">
   <rdfs:domain rdf:resource="#Increment_Economic_Event"/>
   <rdfs:range rdf:resource="#Economic_Resource"/>
   <owl:inverseOf rdf:resource="#inflow"/>
   <rdfs:subPropertyOf rdf:resource="#inverse_of_stockflow"/>
</owl:ObjectProperty>
```

Notice that also the stockflow and participation relations are specialized in disjoint 'sub-relations'. The corresponding association classes in the UML class diagram are represented by means of the OWL ObjectProperty construct. The current specification of OWL does not allow declaring subproperties as being disjoint (this issue will probably be solved by the OWL 1.1 extensions). A solution could be reifying the stockflow and participate relations (i.e. declaring them as OWL classes and next declaring disjoint subclasses). A drawback of this approach is that additional OWL properties are needed to represent all associations between the reified associations and the original classes. These additional OWL properties will have no direct counterpart in the REA conceptual representation, so it might be hard to give them meaningful names. That is why we preferred not to reify them. The UML constraints on the specializations in Figure 2 are also total or complete, meaning that all members of a superclass must be members of at least one subclass. This can be formalized be defining a covering axiom on the superclass which states that the superclass is the union of the subclasses.

The formalization of the REA axioms is less straightforward and different approaches can be taken. Based on the ODM proposal we decided to formalize the multiplicities on the association ends by adding OWL minimum and maximum cardinality restrictions to the range of properties where necessary (see Table 5). This might not be the most appropriate solution because of semantic differences between cardinality restrictions in OWL and cardinality constraints in conceptual modeling languages [45]. An alternative for a maximum cardinality of one on the range of the property is to declare the OWL property as functional (see [7] for an example). A property is functional if for a given individual, there can be at most one individual that is related to the individual via the property.

Table 5. Transformations UML to OWL (multiplicities)

UML elements	**OWL elements**
Multiplicities	maxCardinality, minCardinality

```
<owl:Class rdf:ID="Increment_Economic_Event">
<rdfs:subClassOf>
   <owl:Restriction>
   <owl:onProperty rdf:resource="#inverse_of_inflow"/>
   <owl:minCardinality
       rdf:datatype="&xsd;int">1</owl:minCardinality>
   </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
   <owl:Restriction>
   <owl:onProperty rdf:resource="#inverse_of_inflow"/>
   <owl:maxCardinality
       rdf:datatype="&xsd;int">1</owl:maxCardinality>
   </owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

5 Conclusion and Future Research

This paper presents the development of two new representations of the Resource Event Agent (REA) business ontology. This development addresses shortcomings in the current specification that are likely to impede REA's use as a run-time ontology supporting semantic integration and interoperability of heterogeneous business applications and enterprise systems. The first representation is a UML class diagram that can be used by human users as an explicit, unified and uniformly represented specification of REA's conceptualization of business. The second representation is a formal, machine-readable specification obtained by applying UML to OWL mappings. The OWL formalization of REA, which would not be possible without first redesigning its conceptual representation, will facilitate its operationalization in ontology-driven systems as it makes REA executable.

A research contribution of the paper is the embedding of the REA redesign and formalization activities into a comprehensive business ontology reengineering methodology. Other activities suggested by this methodology, such as ontological evaluation and synthesis, can readily be performed based on the proposed conceptual representation. This new UML representation of REA may also serve as an improved basis to compare REA to other business ontologies and to evaluate the degree to which this business conceptualization is shared in the business community. The OWL representation of REA is of practical value for those wishing to explore the use of REA as a run-time ontology. It also allows for formal ontology evaluation, for instance verifying the consistency of the concept definitions using a Description Logic reasoner.

The limitation of this research is its focus on representation and formalization issues. Future research is needed to validate REA and to address ontological content rather than just form. An ontological analysis with respect to an upper-level ontology (e.g. SUMO, Dolce, GFO, BWW, SOWA, …) is part of this validation. Future research may also investigate the use of other languages for business ontology modelling. In particular, enterprise modelling languages may offer the same advantages as UML (given the availability of UML profiles for these languages) whilst allowing semantically richer descriptions of business domain semantics (through domain-specific modelling constructs).

Currently, we are developing a 'proof of concept' REA-driven enterprise information integration application to demonstrate effectiveness (i.e. the use of the reengineered REA as an operational ontology) and feasibility (i.e. REA creating interoperability between heterogeneous business applications). Additionally we are also developing a revised graphical representation of the formalized REA-ontology specification by using the OWL UML profile which is part of the ODM specification. This graphical representation will be further discussed and evaluated in the REA research community.

References

1. Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acquisition 5 (1993) 199-220
2. Guarino, N., Giaretta, P.: Ontologies and Knowledge Bases: Towards a Terminological Clarification. In: Mars, N. (ed.): Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing. IOS press, Amsterdam (1995) 25-32
3. Fensel, D.: Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Springer-Verslag (2001)
4. Borst, W.N.: Construction of Engineering Ontologies. Centre for Telematica and Information Technology, Enschede, The Netherlands (1997)
5. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological Engineering. Springer-Verslag (2004)
6. Guarino, N.: Formal Ontology and Information Systems. Proceedings of FOIS'98, Trento, Italy, IOS Press (1998) 3-15
7. Gailly, F., Poels, G.: Ontology-driven Business Modelling: Improving the Conceptual Representation of the REA-ontology. FEB Working paper series. Faculty of Economics and Business Administration, Ghent University (2007)

8. Assmann, U., Zchaler, S., Wagner, G.: Ontologies, Meta-Models, and the Model-Driven Paradigm. In: Calero, C., Ruiz, F., Piattini, M. (eds.): Ontologies for Software Engineering and Software Technology (2006)
9. Baida, Z., Gordijn, J., Saele, H., Akkermans, H., Morch, A.Z.: An ontological approach for eliciting and understanding needs in e-services. In: Pastor, O., Falcao e Cunha, J. (eds.): Advanced Information Systems Engineering (CAiSE 2005), LNCS 3520, Springer (2005) 400-414
10. Dietz, J.L.G.: System Ontology and its role in Software Development. In: Castro, J., Teniente, E. (eds.): Advanced Information Systems Engineering wokshops (CAiSE 2005), Porto, Portugal, 2, FEUP edicoes (2005) 273-284
11. Grunninger, M.: Enterprise Modelling. In: Bernus, P., Nemes, L., Schmidt, G. (eds.): Handbook on Enterprise Architecture. Springer (2003)
12. Guarino, N.: Understanding, building and using ontologies. International Journal of Human-Computer Studies 46 (1997) 293-310
13. Castano, S., Ferrara, A., Montanelli, S.: Ontology knowledge spaces for semantic collaboration in networked enterprises. Business Process Management Workshops 4103 (2006) 336-347
14. Vujasinovic, M., Marjanovic, Z.: Data level enterprise applications integration. Business Process Management Workshops, LNCS 3812, Springer (2006) 390-395
15. Izza, S., Vincent, L., Burlat, P.: A Framework for Semantic Enterprise Integration. In: Konstantas, D., Bourrières, J.-P., Léonard, M., Boudjlida, N. (eds.): Interoperability of Enterprise Software and Applications. Springer-Verslag (2006)
16. McIlraith, S.A., Son, T.C., Zeng, H.L.: Semantic Web services. IEEE Intelligent Systems & Their Applications 16 (2001) 46-53
17. W3C: Web Service Modeling Ontology (WSMO). (2006)
18. Geerts, G.L., McCarthy, W.E.: An Ontological Analysis of the Economic Primitives of the Extended-REA Enterprise Information Architecture. IJAIS 3 (2002) 1-16
19. Allen, G.N., March, S.T.: The effects of state-based and event-based data representation on user performance in query formulation tasks. MIS Quarterly 30 (2006) 269-290
20. Geerts, G., McCarthy, W.E.: The Ontological Foundation of REA Enterprise Information Systems. (2005)
21. Geerts, G., McCarthy, W.E.: Policy-Level Specification in REA Enterprise Information Systems. Journal of Information Systems Fall (2006)
22. UN/CEFACT: UN/CEFACT Modeling Methodology (UMM) User Guide. (2003)
23. ECIMF: E-Commerce Integration Meta-Framework. Final draft. ECIMF Project Group (2003)
24. O'Leary, D.: Different Firms, Different Ontologies, and No One Best Ontology. IEEE Intelligent Systems Sep/Oct (2000) 72-78
25. McCarthy, W.E.: The REA Accounting Model: A Generalized Framework for Accounting Systems in A Shared Data Environment. The Accounting Review july (1982) 554-578
26. Sowa, J.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Pacific Grove, Brooks/Cole (1999)
27. Hruby, P.: Model-driven design using business patterns. Springer, New York (2006)
28. Dunn, C.L., Cherrington, J.O., Hollander, A.S.: Enterprise Information Systems: A Pattern Based Approach. McGraw-Hill (2005)
29. Gailly, F., Poels, G.: Towards a Formal Representation of the Resource Event Agent Pattern International Conference on Enterprise Systems and Accounting (ICESAcc), Greece (2006)
30. Bialecki, A.: REA ontology. http://www.getopt.org/ecimf/contrib/onto/REA/ (2001)
31. Chou, C.-C.: Using ontological methodology in building the accounting knowledge model – REAP. 2006 AAA mid-year meeting - 2006 AI/ET Workshop (2006)

32. Jarrar, M.: Towards Methodological Principles for Ontology Engineering. STARLAB. Vrije Universiteit Brussel, Brussel (2005)
33. Fernández-López, M., Gómez-Pérez, A., Juristo, N.: METHONTOLOGY: From ontological art towards ontological engineering. Working Notes of the AAAI Spring Symposium on Ontological Engineering, Stanford, AAAI Press (1997)
34. Gómez-Pérez, A., Rojas, M.D.: Ontological Reengineering and Reuse. In: Fensel, D., Studer, R. (eds.): 11th European Workshop on Knowledge Acquisition, Modeling and Management, Dagstuhl Castle, germany, Springer-Verslag (1999) 139-156
35. Brockmans, S., Volz, R., Eberhart, A., Löffler, P.: Visual Modeling of OWL DL Ontologies Using UML. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.): The Semantic Web - ISWC 2004, Hiroshima, Japan, LNCS 3298, Springer (2004) 198-213
36. Hartmann, J., Spyns, P., Giboin, A., Maynard, D., Cuel, R., Suárez-Figueroa, M.C., Sure, Y.: Methods for ontology evaluation. Knowledge Web Deliverable D1.2.3, v. 1.3.1 (2005)
37. Gangemi, A., Catenacci, C., Ciaramita, M., Lehmann, J.: A theoretical framework for ontology evaluation and validation. Proceedings of SWAP 2005, the 2nd Italian Semantic Web Workshop, Trento, Italy, 166, CEUR Workshop Proceedings (2005)
38. Lassila, O., McGuiness, D.L.: The Role of Frame-Based Representations on the Semantic Web. Technical Report. Knowledge System Laboratory, Stanford University, Stanford, California (2001)
39. Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M.K., Smith, J.: UML for ontology development. Knowledge Engineering Review 17 (2002) 61-64
40. Spaccapietra, S., Parent, C., Vangenot, C., Cullot, N.: On Using Conceptual Modeling for Ontologies. Proceedings of the Web Information Systems Workshops (WISE 2004 workshops), LNCS 3307. Springer-Verslag (2004) 22-23
41. Spyns, P.: Object Role Modelling for ontology engineering in the DOGMA framework. In: Robert Meersman et al. (ed.): On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops, Agia Napa, Cyprus, LNCS 3762, Springer (2005) 710-719
42. Mylopoulos, J.: Information modeling in the time of the revolution. Information Systems 23 (1998) 127-155
43. Davies, I., Green, P., Rosemann, M., Indulska, M., Gallo, S.: How do practioners use conceptual modeling in practice? Data & Knowledge Engineering 58 (2006) 358-380
44. OMG: Ontology Definition Metamodel: OMG Adopted Specification (ptc/06-10-11). Object Management Group (2006)
45. de Bruyn, j., Lara, R., Polleres, A., Fensel, D.: OWL DL vs. OWL Flight: Conceptual Modeling and Reasoning for the Semantic Web. World Wide Web Conference (WWW 2005), Chiba, Japan, ACM (2005)