Contexts for Concepts: Information Modeling for Semantic Interoperability

Paul Oude Luttighuis[1], Roel Stap[2], and Dick Quartel[1]

[1] Novay, P.O. Box 589, NL-7500 AN Enschede, The Netherlands
[2] TNO, Colosseum 27, NL-7521 PV Enschede, The Netherlands
{Paul.OudeLuttighuis,Dick.Quartel}@novay.nl, Roel.Stap@tno.nl

Abstract. Conceptual information modeling is a well-established practice, aimed at preparing the implementation of information systems, the specification of electronic message formats, and the design of information processes. Today's ever more connected world however poses new challenges for conceptual information models, as different models should enable mutual connection and reconciliation, even when developed in totally different situations. This paper argues that the 'vertical' bias of today's conceptual information modeling practice diverts models from meeting this new, 'horizontal' need. As an alternative, a conceptual information modeling approach is described that is simultaneously unconventional as well as interoperable with existing approaches. The key to this approach is conceptual context-awareness. It is based on ideas from the Metapattern work [31].

Keywords: enterprise interoperability, semantic interoperability, conceptual modeling, context-awareness, enterprise integration, information modeling.

1 Introduction

Conceptual information modeling practice has grown in the field of information systems design. Conceptual information models — CIMs, for short — mark the border between the worlds of the user (the problem domain) and the information system (the solution domain), or between what is called requirements engineering and systems engineering in information systems design [17]. Therefore, CIMs play a double-faced role in being both descriptive as well as prescriptive.

Figure 1 illustrates this view by positioning conceptual information modeling — CIMing, for short — in a design process chain. The vertical arrows do not imply a waterfall-type of process, but may involve iterative design cycles. This vertical positioning of CIMing is challenged by today's trend towards networked information processes and systems. In fields like organizational science and software engineering, these trends have led to the rise of new paradigms, such as network organizations [7] and service-oriented architecture (SOA) [10]. These paradigms share a shift from 'hierarchical' towards 'networked' thinking. In SOA, services are not part of other services, but linked through contracts. And, participants in a network organization are not casted in a classical hierarchy, but involved in peer-to-peer or market-type relations.

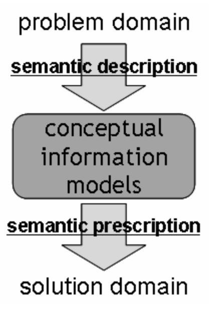

Fig. 1. Design roles of CIMs

Somehow though, these developments have not led to the adoption of new or evolved paradigms in the CIMing realm, although at least one is available [31]. Such situation may suggest that this realm has been ready for large-scale information networking. This paper argues otherwise. Although some of the current CIMing approaches do have the required relational ingredients at hand, these approaches are generally used to establish CIMs with predominant hierarchical structure. With regard to SOA, this may seem surprising, because major service characteristics such as self-containment, loose coupling and reusability are not primarily technical but semantic. No service will be more reusable than the information it deals with.

A naive approach to deal with the trend towards networked systems and processes would be to enlarge the modeling scope. This approach however assumes the fundamental possibility of grasping all possible domains into a single object of design. It also assumes the fundamental possibility of an absolute, objective, context-independent perception of the world. Even for those who think this is theoretically possible, it will still prove to be practically unfeasible.

Therefore, one is left with the challenge to connect and reconcile CIMs from different sources and circumstances. CIMs in practice, though, vary considerably in the extent to which they allow for mutual connection and reconciliation. Semantic reconciliation is already subject of research in the ontology engineering field [1].

This paper describes a context-aware approach to CIMing. The approach is inherently networked and horizontal, in contrast to common approaches that are predominantly hierarchical and vertical.

Rather than working up from a philosophy or formal semantics, the paper introduces the approach by letting it materialize from a set of confrontations between pairs of CIMs. Each of these confrontations represents a typical semantic interoperability problem. Sections 2 and 3 start with basic notions. Section 4 presents the conceptual confrontations. Section 5 explains the approach – Contextual CIMing – in terms of a semi-formal definition and some basic principles. Section 6 compares and combines Contextual CIMing with a range of other approaches, viz. UML, ERM, ORM, OWL, MERODE, relational database modeling, and large-scale e-business standardization. Section 7 discusses the business rationale behind this work. And section 8 ends with conclusions and ideas for further work.

2 Modeling Options

Experienced conceptual information modelers will acknowledge that the same real-world phenomenon may be modeled in alternative ways ([29], Ch. 9; [25], Ch. 7, 8). Should enrollment be modeled as an action (to enroll) or as an entity (enrollment)? Should a property be modeled as an attribute or as a relation? Should a specialized type be modeled as a subtype or as a role type? Should a relation have its own lifecycle? Such dilemmas are typically settled by a modeler's professional experience, sometimes using explicit modeling patterns ([11]; [15]). Often, they are settled with a claim that one alternative is more natural or more concise than the other.

Naturalness. A claim of naturalness suggests that one modeling alternative is somehow closer to the real world than the other. Yet, the assumption that the only connection between the CIM and the real world is via the interpreters of the model, makes any such claim relative to the specific interpreters. The only way to let such a claim extend beyond a single interpreter is by confrontation with models of others. Then, establishing a CIM is not merely a matter of objective problem analysis, it is primarily a matter of communication about the problem with stakeholders and subjective confrontations of their views.

This also sheds a light on the use of natural language as a basis for CIMs. For instance, NIAM/ORM [12] is well-known for the use of natural language for information analysis. Yet, although natural language may be seen as a powerful and flexible means of human communication, there is little reason to assume that the structure of any specific natural language utterance, articulated by a specific writer or speaker in a specific context, would be any more natural than another. In fact, comparable real-world phenomena may be expressed by a wide variety of different natural language sentences, the choice between which may be guided by criteria considered important in the particular context at hand, such as efficiency, bias, perspective, or habit.

Conciseness. As for conciseness claims, two types may be distinguished, referring to the two roles of CIMs depicted in Figure 1. First, *prescriptive conciseness* is the quality of leading to more efficient implementations in targeted software platforms. Clearly, modeling decisions based on such considerations carry the risks of implementation bias. Second, *descriptive conciseness* is the conciseness of the problem description, i.e., of the CIM itself. Modeling decisions may lead to smaller or less complex models. For example, one might say that in a given problem domain, no country will ever have multiple capitals. This may lead to a model with the *capital* type subordinate to the *country* type. This however is a context-specific decision. It excludes contexts in which countries may have multiple capitals.

In general, conciseness counteracts variation and distinction in CIMs, and thus their expressivity. Should CIMs be prepared for semantic interoperability, and hence for variable contexts, they should strive to refrain from prescriptive and descriptive conciseness for specific contexts. Especially for descriptive conciseness, this will not be easy, as users, business people, and project clients will generally accept CIMs only if they address their own specific contexts concisely.

3 Context

This paper presents an approach to CIMing, called *Contextual CIMing*, that uses context-aware CIMs. A CIM is considered context-aware if it models the linguistic context of each of its concepts explicitly, i.e., the concepts on which the concept semantically depends. Context-awareness improves the openness of CIMs, i.e., its receptivity to change and reconciliation, and thus its interoperability. The approach is not meant to substitute other CIMing approaches, but to enhance and prepare them for the confrontation, connection, and reconciliation with other CIMs. Our aim is to preserve a formal relation with other approaches, so that transformations are possible both ways.

As Figure 2 shows, Contextual CIMing involves context-aware CIMs that provide a common context at which connection and reconciliation is feasible. It should be said however, that there is no a priori stringent separation between context-aware and context-unaware models. Rather, context-awareness is a quality that CIMs may have in different grades. The only way to experience a model's level of context-awareness is by confronting it with other models. And even then, the revealed level only holds with respect to the other models encountered.

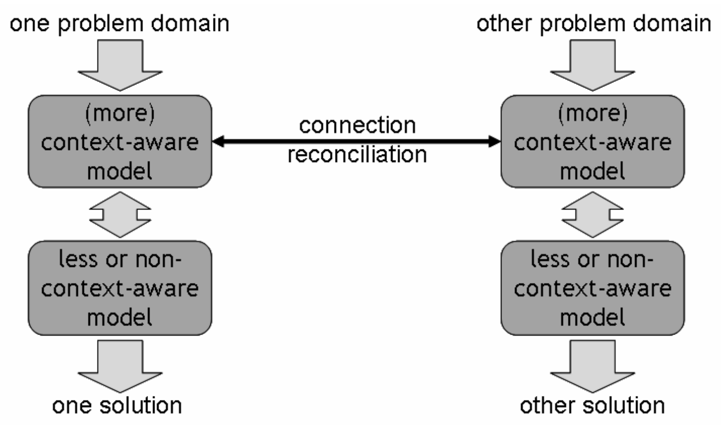
one problem domain
other problem domain
(more) context-aware model
connection
reconciliation
(more) context-aware model
less or non-context-aware model
less or non-context-aware model
one solution
other solution

Fig. 2. Reconciliation requires context-aware CIMs

As a consequence, context-awareness should be included in the CIM itself, rather than a separate model extension. In other words, there should not be an a priori distinction between information and meta-information, as such distinction would produce its own bias. This approach is the basis for a model-based semantic interoperability process, depicted in Figure 3. In this process, CIMs of parties participating in the interoperability problem are first made (more) context-aware and then connected or reconciled, leading to a shared CIM. This shared model retains its relation with the particular models of the participants, and is the basis for the realization of an intermediate information process or system ([9]; [22]). If such intermediation is not wished for, the shared model may at least be the basis for communication formats.

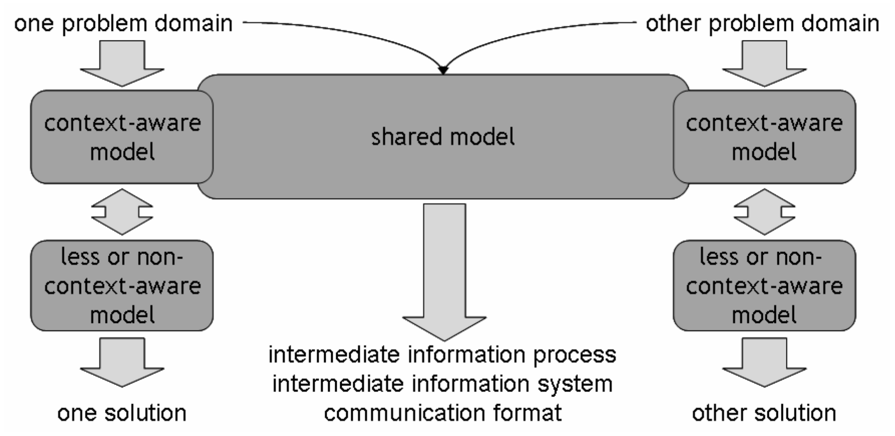

Fig. 3. Semantic interoperability process map

The map of Figure 3 allows for different semantic interoperability routes. On either side, such route may comprise:

- the tabula rasa composition of a context-aware model from the outset; in the map, this corresponds to a route from the problem domain down;
- the transformation of an existing non- or less context-aware CIM to a context-aware CIM, in order to prepare for reconciliation; in the map, this corresponds to a route from the non- or less context-aware model upwards;
- in case a solution is in place, but no CIM to go with it, the reverse engineering of a CIM out of the solution; in the map, this corresponds to a route from the solution upwards.

Concluding, rather than two, as depicted in Figure 1, CIMs should play at least four roles, as illustrated in Figure 4: semantic description of the problem, semantic prescription for the solution, semantic interoperability with other models, and semantic adjustability for future models.

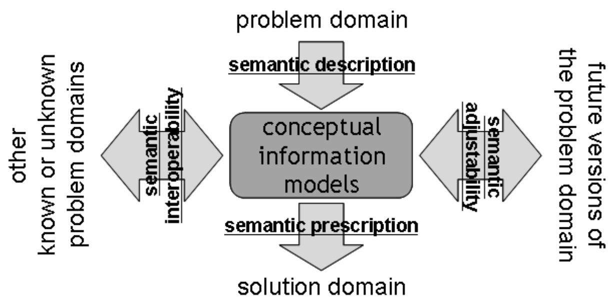
problem domain
semantic description
other known or unknown problem domains
semantic interoperability
conceptual information models
semantic adjustability
future versions of the problem domain
semantic prescription
solution domain

Fig. 4. Four roles of CIMs

4 Conceptual Confrontations

It is worthwhile to revisit how different types of semantic interoperability are often conceived. Pokraev ([22], Section 2.3) gives a classification of seven types of semantic interoperability problems: "*Different systems use …*":

1. *"the same symbol to represent concepts with disjoint meanings."*
2. *"the same symbol to represent concepts with overlapping meanings."*
3. *"the same symbol to represent concepts with more general/specific meanings."*
4. *"different symbols to represent the same concept."*
5. *"different symbols to represent concepts with overlapping meanings."*
6. *"different symbols to represent concepts with more general/specific meanings."*
7. *"different definitions of the same concept."*

For the purpose of this paper, interoperability problem types 1 and 4 pose little problems. Confusion about symbols (syntax) can in principle be dealt with easily by changing names or using name spaces. In terms of the well-known semiotic triangle [21], this problem can be dealt with in the symbolic corner. Using the same argument, we do not distinguish between types 2 and 5, nor between types 3 and 6. Furthermore, type 7 can also be dealt with in the symbolic corner, because if parties agree that the same concept is at stake, they can simply use either definition, or both.

This leaves us with two specifically semantic interoperability problems:

a) *"Different systems use concepts with overlapping meanings."*
b) *"Different systems use concepts with more general/specific meanings."*

These two distinguish conceptual overlap from conceptual generalization/specialization. The latter though, can be seen as a special case of the former. Just like *woman* and *president* may be seen as having overlapping meanings, *dog* and *Great Dane* can be seen so. Hence, for this paper, a semantic interoperability problem occurs when *two parties use different concepts with related meanings*. Notice that this statement swapped *system* for *party*, in order to avoid the idea that only software agents are at stake, and *overlapping* for *related*, in order to avoid the preassumption of a first-order set-theoretic formal semantics, where *overlap* implies conjoint instance sets.

Obviously, the parties use different concepts, as there would be no semantic problem otherwise, except for the symbolic problems mentioned before. Equally obviously, these different concepts should be related, as otherwise there would be no sense in interoperating. The interoperability challenge is precisely to find that relation. The conceptual differences might be large, but the shared context should reveal a relation, even if it would take a range of intermediate concepts to bridge the gap. So, semantic interoperability is about conceptual reconciliation and hence, in model-based approaches, about the confrontation of different CIMs. This requires context-awareness.

The following paragraphs will address four typical types of interoperability problems that arise when concepts from different CIMs are related. Each of these confrontations leads to reconciliation using the same modeling construct. This modeling construct is the basis for Contextual CIMing.

4.1 First Confrontation: Static and Dynamic

The distinction between static structure and dynamic structure is felt as very intuitive by many. Conceptual modeling approaches embed such a distinction in their modeling language, or even use it to scope their language or diagramming technique. For example, business process modeling [23] is dedicated to dynamic phenomena, and data modeling to static phenomena. Some methodologies [25] involve both, but separate them a priori. Apparently, this distinction is felt to be very fundamental. This also causes many to see semantic interoperability as a data issue (see e.g. [16]), whereas semantics in fact applies to processes as much as it does to data.

In Contextual CIMing therefore, classifying a phenomenon as either static or dynamic is a matter of perspective. One CIM might see *to enroll* as a business activity, where the other sees *enrollment* as a business object. Natural language does not settle the dispute. Even though many modern natural languages have a subject-verb-object-type of grammar, they generally offer a grammatical construct for switching between the verb and the noun for the same phenomenon. In English, for instance, this is called *gerund* [4]. So, any reconciliation of two CIMs that have, given their contexts, made different modeling choices in this respect, requires the ability to match *to enroll* with *enrollment*, assuming they have agreed to address the same phenomenon.

So, rather than objects or entities on the one hand, and events or actions on the other, one needs a notion that transcends this distinction. This is possible by first having all dynamic phenomena materialize into concepts themselves and, second, awarding any concept its dynamic aspect. This is convenient, as many would agree that virtually no concept is entirely static. Dynamics are an aspect of concepts, not a separate category.

4.2 Second Confrontation: Properties

It is a well-known transformation in information modeling to release an attribute type from the object type that contains it. Instead of saying that a *person* has an attribute called *birth country* that always carries a value drawn from a *country* type, one might also model *persons* to be related to *countries* by a *birth country* type.

There may be several reasons for choosing the attributive variant in this case. One is that it allows for more efficient implementations (prescriptive efficiency). Another is that a birth country is often felt to be attributive to the person, rather than to the country (naturalness). But, why would any of the two be more natural than the other?

The attributive variant of modeling properties can be seen as a special case of the relational variant, because the relational variant enables the property to have properties itself, whereas the attributive variant doesn't. So, the reconciliation of the two variants is found in the relational variant itself.

4.3 Third Confrontation: Roles

Suppose an organization wants to have their employee information system interoperate with their knowledge management system. In the first, *engineer* occurs in the CIM as a subtype of *employee*. In the second, *engineer* occurs in the CIM as a subtype of *professional*. Now, suppose also that both occurrences of *engineer* would be found to address the same real-world phenomenon, but *employee* and *professional* don't. Still, both the *employee*-perspective as well as the *professional*-perspective on engineers should be retained. See the leftmost model fragment in Figure 5.

In case subtyping would involve inheritance — that is, the proliferation of any properties from the supertype to the subtype ([25], Section 8.2) — the shared *engineer* type might inherit the properties from both sides. However, proliferation is awkward here, because the context-specific properties of *engineer* should remain local. The shared engineer type should be the semantic intersection of the *engineer-as-an-employee* and the *engineer-as-a-professional*. Property proliferation would make *engineer* the semantic composition of both and propagate context-specific properties.

Contextual CIMing therefore refrains from any property proliferation and looks at subtyping as a one-to-one relationship between the subtype and the supertype. This one-to-one relationship can be identified as an existence dependency relationship ([25], Section 4.2). Rather than saying that an instance of a subtype always is also an instance of the supertype, both types are taken to have disjoint instance sets, mutually tied by existence dependence.

This is a well-known transformation used in object-relational mapping [2]. In the example, the context-specific *engineer* types at both ends are made context-aware: they can be seen as replaced by "*engineer-as-an-employee*" and "*engineer-as-a-professional*". In addition, both are existence dependent on the shared *engineer* type.

In this model, *engineer*, *engineer-as-an-employee* and *engineer-as-a-professionals* all have their own set of instances. They have become role types that tie the shared fragment of the conceptual model to the private fragments of the model. Even if some might object that these types still address the same "real-world" person, they are different linguistic categories and therefore span different linguistic instance sets. These sets are tied by existence dependency, instead of by property proliferation.

This may be depicted by the rightmost model fragment in Figure 5. The *engineer-as-an-employee* type is existence dependent on both the *employee* and the *engineer* type. For brevity, the labels on the existence dependency relations are omitted, as well as the multiplicity of the existence dependency, which is single everywhere in this example. Figure 5 uses a notation that is also used in our case studies (see below), but other notations might be used instead, such as UML association classes or ERD associations. For a comparison with these and other approaches, we refer to section 6.

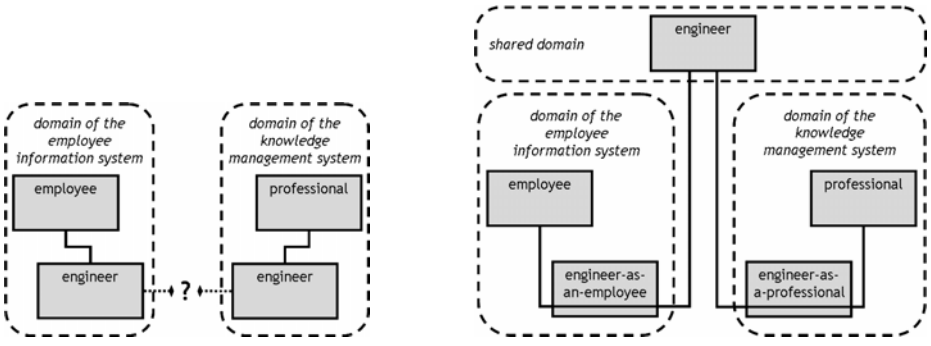
domain of the employee information system
employee
engineer
?
domain of the knowledge management system
professional
engineer
shared domain
engineer
domain of the employee information system
employee
engineer-as-an-employee
domain of the knowledge management system
professional
engineer-as-a-professional

Fig. 5. Before (left) and after (right) reconciliation with role types

Every type is represented by a rectangle. A line that runs through a rectangle represents that the existence of the instance of the represented type depends on the instances of the connected types. This example uses two-way associations, but one-way or more-way associations may be used as well.

In this way, Contextual CIMing enables loose coupling between shared and private information types, which is a much wished-for quality in interoperability. The adoption of property proliferation would imply tighter semantic coupling.

The choice between role types and subtypes is also discussed by [24] (Section 4.10.4). They state that "*the role entity class is usually the neatest solution […] when there are significant differences […]*". Given that preparation for interoperability involves uncertainty about new contexts, there will probably always be significant differences, so that the role type is to be preferred. They however see "*a danger […] of blurring the distinction between subtypes and […] relationships*". But, in Contextual CIMing, subtypes are contextually restrained role types. They represent a model optimization that can only be justified within restrained contexts.

4.4 Fourth Confrontation: Higher Context

Above, the confrontation between subtypes in different contexts made them become role types, which relate the shared semantic domain to the private semantic domains. Now, this is taken a step further to show that this can be done for any type.

Suppose that two government agencies in a given country operate different information systems. On the one hand, municipalities operate a citizen registration system that involves a type *person*. On the other, the tax department operates a CRM system, involving a type *person* as well. The parties agree that there is ample reason to communicate about these persons, but discover that the populations addressed by each of their registrations are different. These differences are not accidental, they reflect the context (in particular, the business process) for which the registrations were designed originally. See the leftmost model fragment in Figure 6.

To start, the parties notice that their *person* types are semantically different, by the bare fact that they represent different populations. Only persons that are residents of the country are represented in the citizen registration. And, only persons that are taxable persons with respect to some legal taxation system are recognized to be represented in the CRM system.

The next step should be *not* to stress that some persons may be both taxable and residents. Instead, *residents* and *taxable persons* should be considered as different, but related. These types are both found to be specializations of *natural person*. From there, the same approach as with subtypes can be used: resident and taxable person are seen as relations, connecting specific contexts (the citizen registration and the CRM system) with the shared type *natural person*. Looking back at the original CIMs of each system apart, their original *person* types appear to have been context-specific.

Again, in Contextual CIMing, the resulting *resident* and *taxable person* types do not have overlapping instance sets. Rather, specific pairs of *residents* and *taxable persons* may be existence dependent to the same *natural person*. The real-world counterpart of such a *natural person* may henceforth have at least three linguistic manifestations: one for each linguistic category present in the reconciled CIM.

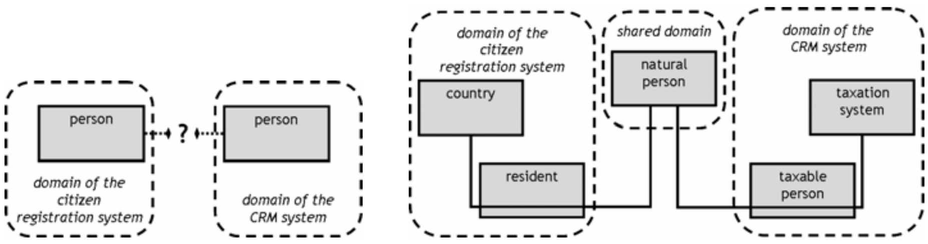

Fig. 6. Before (left) and after (right) reconciliation with context

This may be depicted as in the rightmost model fragment in Figure 6. Notice the resemblance with Figure 5. There is also an important difference though: the particular models at both ends have grown: they have become more context-aware. This is because the predecessors-before-reconciliation of the *resident* and *taxable person* types where "top types", who were assumed to "simply exist". Reconciliation though has revealed that their semantics is context-dependent. The context categories should be introduced into the CIM, so that the model gains context-awareness. This example chose to model a *resident* as a *natural person* with respect to a *country*, and a taxable person as a natural person with respect to a legal system for taxation.

5 Contextual CIMing

Even though there may be many more types of confrontation than those addressed above, out of every confrontation came the same modeling construct as the one best prepared for reconciliation. It is an association-oriented type that has its own set of instances, but at the same time relates instances of other types. It therefore always has object-like as well as relational characteristics. Every type is existence dependent to the other types it relates. Furthermore, the type may represent any real-world phenomenon, be it felt as static or dynamic.

The Contextual CIMing language involves only this one single construct, called *contextual specialization*. Contextual specialization connects any type to the one or more other types that constitute its context. There is no type without context, i.e., any type is a contextual specialization of one or more other types [31].

In order to enable finiteness of models, the modeler may close or conclude the model with a special type, called the *contextual horizon* ([32], Paragraph 31). This type has no context, that is, not one that is represented in the given model. Adjustment or confrontation with other models may make this horizon shift, because confrontation of models leads to the widening of context. So, the contextual horizon shows that any model is blinkered, but is aware of that.

As a semiformal definition of a contextual model in Contextual CIMing, the following may serve. A Contextual CIM is a set C of concepts, a special $\perp \notin C$ (the horizon) and the context function $c: C \rightarrow (C \cup \{\perp\})^+$. The context function labels each contextual connection, so that they may be told apart.

Quality before quantity. A crucial principle of Contextual CIMing is that concepts are firstly defined qualitatively, that is, as a feature-as-such, without their "sets of instances". Such sets constitute the quantitative aspect of a concept, which is seen as a refinement of the qualitative aspect. Hence, contextual specialization does not include cardinality restrictions, for instance. These, though, can be added when the modeler decides to refine a model to the quantitative level.

So, contextual specialization is a qualitative relation between concepts. As such, it can guide conceptual model reconciliation and evolution.

This implies that Conceptual CIMing needs a formal semantics beyond classical set-theoretic semantics, that associates a set of instances with each concept. Instead, the concepts-as-such (and their contextual specialization relations) should be the backbone of the formal semantics.

Still, at the quantitative level, each concept is awarded its set of instances. Yet, each two of these sets (for different concepts) are disjoint. Rather than overlapping, instances may be related, but only if their concepts are involved in a contextual specialization relation at the qualitative level. These quantitative relations may be governed by cardinality restrictions. So, again, quality enables quantity, not reversely.

Intensional, not extensional. Second, Contextual CIMing involves the arrangement of concepts, where each concept simultaneously relates other concepts, and adds its own (qualitative) substance to the model. In a geometrical analogy, concepts would simultaneously be points and lines. This implies that Contextual CIMing requires more than a first-order formal semantics. It requires a more recursive semantics.

Even stronger, Contextual CIMing does not contain a fixed first layer, in which concepts can be defined extensionally, that is, by referring to a set of "real world" objects, thought to be collectively represented by the concept. The model horizon is only a temporary break in a contextual generalization process, that may asymptotically converge, but will never end definitely.

Instead, all concepts are defined intensionally, that is, by identifying the contexts from which they take meaning. These contexts are themselves conceptualized, so that they are easily included in the model.

Synthetic, not analytic. By rigorously pointing out that any concept only takes meaning from its context, any absolute objectivity of concept definitions is abandoned. There is no a priori real-world structure, no "semantic coordinate system", on which a conceptual model can be projected. This implies that conceptual models become in fact synthetic, rather than analytic, because there is no natural structure against which any analysis can be carried out.

So, if one would continue to use the term *information analysis* for the process that leads to conceptual information models, it would not be information itself that is being analyzed. Instead, the objects of analysis would be the perspectives of the stakeholders on their domains. Information itself, or at least the conceptual models used to represent it, would be synthesized, not analyzed.

Structuralism and atomism. These principles can be seen as to arise from a philosophical choice. Contextual CIMing appeals to linguistic structuralism of De Saussure, more than to Russell's and Wittgenstein's logical atomism. Logical atomism teaches that language (hence CIMs) is built from primitive elements (types, facts, terms), that carry meaning in themselves. Contrarily ([14]), "*[…] a crucial feature of Saussurean structuralism is the idea that structure itself creates the units and their relations to one another. […] Linguistic structure is not an assemblage: it is not built up piecemeal.*" In structuralist thinking, information takes the structure of the language (hence, the CIM) used to express it, rather than "the structure of the real world". The only way to approach "real-world structure" is socially, in communication — by having interpreters reconcile their models. In philosophy, structuralism enjoys more appreciation than in CIMing, probably because the latter is dominated by technology-oriented professionals.

6 Related Approaches

This paper describes an approach to CIMing that is simultaneously unconventional as well as familiar. It is unconventional because it positions quality over quantity, it reconciles modeling options that, for many, are intuitively felt as irreconcilable. It reconciles the static and dynamic natures attributed to real-world phenomena, it reconciles subtyping and relations, as well as objects and relations. In the end, it is the (objectified) relation that comes out as the most important, or even only, modeling construct. This may come as no surprise, as it has been driven by the demand for semantic *inter*-operability, rather than the design of a single information system or process.

At the same time though, relational modeling is not new in itself. Virtually any information modeling approach used in practice has at least some notion of relation types. Often though, relations are seen as second- or third-class citizens in information modeling.

Sometimes, the subordination of relations is embedded in the modeling language itself. Sometimes, as the remainder of this paper will show, the language itself essentially enables the predominant use of relations. But then, these languages are often so rich — they supply so many modeling constructs — that, in practice, one hardly sees predominantly relational CIMs. In fact, modeling tools for these languages may even decide to not support the entire relational capabilities of the language and hence disable contextual CIMing. It is important to notice, though, that the reuse of existing modeling languages for the purpose of Contextual CIMing is only possible as far as these languages allow for structuralist semantics.

ERM and UML. In ERM [3], entity types are the first-class citizens, but relation types are of course also included. In order to use ERM for contextual CIMing, the associative relation, or associative entity, should be promoted to the first class. This is unconventional, as general practice only uses this modeling construct in case a many-to-many relation appears. For UML class diagrams, the exact same points can be made. Here, one needs the promotion of UML's *association class* [19] to first class or even to the only construct used. So, ERM and UML can both be used for contextual CIMing, provided that a very restricted part of the language is used. This part can then be given a structuralist semantics.

ORM. Object Role Modeling (ORM) [12] does not include the attributive variant of property modeling, but uses the relational variant only, as Contextual CIMing would do. Yet, ORM has a built-in distinction between entities and relations. Relations are called *predicates* and may relate any number (one or more) of other types. Still, ORM includes an associative relation type, much like ERM and UML. It is called *objectification*, which conveniently expresses the reconciliation between object (entities) and relations. It is also called *nesting*, suggesting that a relation is nested in an object. This is a less convenient term for contextual purposes, as it suggests the subordination of relations to objects. In contextual modeling, there is no such subordination. Any type simultaneously has objective features (that is, it has a set of instances) as well as relational features (that is, it is existence dependent on the types in its context).

OWL. The DL (Description Logic) variant of OWL [27] does not allow quantifying over predicates, because of its first-order restriction. Since relations translate to predicates, this implies that relations cannot have their own set of instances. The first-order restriction would enable only one level of context in CIMs, which is too restrictive for Contextual CIMing. In contrast, OWL Full supports higher order constructs that are needed for Contextual CIMing, but at the (probable) cost of loss of convenient computational properties.

MERODE. MERODE [25] is a CIMing approach from the object-oriented modeling world. The first step of MERODE's enterprise modeling phase involves: (i) an existence dependency graph (EDG) expressing existence dependency relations among so-called object types, and (ii) the identification of business events and the construction of an object-event table (OET).

Additional structure may be added to the EDG through the distinction between object types and attribute types, and through generalization and specialization (subtyping). MERODE provides guidelines for deciding whether to model a given phenomenon as an object type or an attribute type and whether as a subtype relationship or a so-called role type. It appears that, if these guidelines would be used from the perspective of keeping the model as open as possible, all modeling decisions are taken in favor of object types, rather than attribute types, and in favor of role types, rather than subtyping. So, if one would parameterize MERODE with the requirement to keep the enterprise model as context-aware as possible, the EDG would approximate contextual models. It is approximation, not equality, because MERODE would never decide to turn all object types into role types.

In MERODE, the difference between event types and object types is defined with reference to the real world. Contextual CIMing denies that mere inspection of the real world can settle any dispute about whether some phenomenon should be represented by an object type or an event type. From the perspective of Contextual CIMing, MERODE's object types and event types are specializations of phenomenon types. The participation of object types in event types is specified in the OET. This participation relationship can be seen as a specialization of existence dependency.

Most differences between MERODE and Contextual CIMing can be explained by the object-oriented oriented philosophy of MERODE vis-à-vis the association-oriented background of Contextual CIMing. Despite these differences, the approaches may be used in combination.

Relational database modeling. There is a straightforward structure-preserving transformation of conceptual CIMs into relational database schemes [5]. In this transformation, basically, every type t is given its own table, with its own unique primary key and an attribute — that is, table column — for reference to the key of the table of each type in $c(t)$. This transformation is also semantics-preserving, simply because the relational database itself constitutes the context within which all of its data should be interpreted. The database itself can be seen as the implementation of the contextual horizon in the contextual CIM. So, a switch from a structuralist view to an atomist view is not needed. Much like a map does not equal the area it is meant to describe, the database does not equal the real world it is meant to describe.

E-business standardization. One of the ebXML specifications [26] includes a notion of context in order to allow for some degree of variation. It consists of a set of eight context categories — such as business process, product classification, and industry classification — with which context-specific selection and composition of data components, called core components, can be made. This is very different from Contextual CIMing, because it presumes that core components have context-independent meaning, context is separated from the information model, the context categories themselves are assumed to have context-independent meaning.

XBRL has another way of dealing with context. The base specification [35] allows for a set of three separate context categories: reporting period, reporting entity and reporting scenario. For this, the same can be said as for the ebXML notion of context. In a separate specification, the context notion is generalized with so-called *hypercubes* [8]. At first sight, hypercubes resembles the context function in Contextual CIMing, but to comply to the higher-order character of Contextual CIMing one would need nested or stacked hypercubes. In XBRL, these are not provided, though. This makes XBRL contextually flat, much like the first-order approaches discussed above.

In contrast to e-business standardization approaches, Contextual CIMing starts with the recognition of *variation*, and then looks for common semantic ground, hence for candidates for standardization. E-business standardization approaches tend to go the other way and look at variation as an (unwelcomed) exception on variation.

Structuralist (re)use of other approaches. Most uses of existing CIMing approaches are used — explicitly or implicitly — in an atomist way, whereas Contextual CIMing builds on structuralist views. This might imply an unbridgeable gap between the two groups. Yet, there is a bridge. Such a bridge uses the fact that many CIMing languages contain a sublanguage that allows for structuralist use. This then only helps if the language does not inextricably come with a first-order formalization that excludes the language, and any of its sublanguages, from being used in a structuralist sense. This paper found such sublanguages:

- UML class diagrams, in which all types are association classes;
- ERM diagrams, in which all types are associations;
- ORM (assuming that standard first-order formalization is let loose) diagrams, in which all types are objectified relations;
- MERODE, restricted to EDG's in which all types are separate object types.

7 Business Rationale

In this paper, the business value of, and business need for, interoperability within and between organizations (enterprise interoperability) is taken for granted. The semantic aspect of the problem however often does not get the business attention it should. Semantics is regularly seen as a data issue, in contrast with process interoperability, which is seen as a business issue. This is a flaw, because not only data, but also processes and services need semantics in order to be meaningful and valuable. It is no coincidence that the words *meaningful* and *valuable* have closely related meanings. The semantics of enterprise information, rules, processes, and services is about the (operational) core of the organization. Semantics are indispensable for business IT alignment.

The business value of using Contextual CIMing is about whether and how the fruits of enterprise interoperability are harvested at all. Enterprise interoperability can only be *effective* if all involved are able to meaningfully and valuably engage in the information exchange at hand. This requires the information exchange solution to respect all relevant contexts and, at the same time, seize their common ground as well as relate to particularities. Only when enterprise interoperability solutions are effective, they can bring about *efficiency*, by yielding reuse of enterprise data, processes, services, events, and the like.

In a number of case studies, we found support for these claims. In one case study [33], two government agencies experienced difficulties in reusing each others data about employers, their employees and their wages. The ambition to reuse these data stemmed from an efficiency goal: to reduce the administrative burden for the companies involved. However, at first, neglect of the semantic differences concerning, for instance, the term *employer*, threatened the effectiveness of reuse, since the data was not sufficiently meaningful to all involved. Standardization may seem to be an efficiency measure, but without appreciating necessary variation, efficiency quickly evaporates by lacking effectiveness. In the case study, we reconciled the two party's models about employers, following a reconciliation process as sketched in Section 3. According to the domain experts involved, this reconciliation process led to a model in which both parties simultaneously recognized their own context, as well as their shared universe of discourse.

In a second case study, we again set out to reconcile two information models under the requirement to leave both unchanged, but nevertheless find maximum options for mutual reuse. A semantic bridge (or hinge) between the two was found, thanks to the loose coupling among terms in Contextual CIMing: it keeps semantics local to terms, rather than having properties proliferate through the model.

In both cases, it was welcomed that Contextual CIMing brought in a new and valuable perspective in information modeling, while retaining the connection with many (not all) popular modeling languages in use.

Yet, even though these case studies were successful, there have been too few of them, so far, to substantiate our claim on what may be the most important business value of Contextual CIMing, in relation to other approaches: the added value of Contextual CIMing increases if the semantic scale grows. The more domains, contexts, or perspectives at hand, the less one can neglect variability, and the less one can continue without context-awareness. The same points can be made with regard to semantic changes through time (that is consecutive contexts, rather than simultaneous ones). Only by increasing the scale of our case studies (in time or space), we will be able to prove that Contextual CIMing also yields:

- *agility,* as it postpones premature semantic structures that may hinder change;
- *durability*, as it enables the discovery of shared contexts, which may be called semantic infrastructure, and are more stable than specific contexts;
- *scalability,* as it takes a rigorously networked approach instead of a hierarchical approach.

Again, there is a parallel with SOA in this respect. It is with the ambition to enlarge the scale of our attempts, that we recently started a consortium project with market players and user organizations (www.essence-project.nl).

8 Conclusions

This paper introduced an approach to conceptual information modeling that makes it applicable to large-scale semantic interoperability across domains. The approach is at the same time unconventional as well as related to existing conceptual modeling practice. The unconventional aspect is probably rooted in its structuralist nature, where in information technology surroundings, atomist views generally prevail.

Contextual CIMing postpones distinctions, and hence CIM structure, that other approaches fix a priori in their metamodel. Such distinctions include: static versus dynamic phenomena, attributes versus relations, subtypes versus role types, objects versus relations, information versus meta-information, information versus context. Each of these distinctions hampers large-scale interoperability, because different contexts will apply them in different ways. The real world, nor natural language, can be the ultimate judge when such differences occur. Still, the point is not to deny such distinctions, but to apply them at the appropriate moment.

In parallel, new research lines are investigated. One of them concerns the use of Contextual CIMing for reconciling metamodels of existing CIMing approaches, from the relational, the object-oriented, the business-process, and the service modeling field. Another is the relation between Contextual CIMing and (role- and rule-based) authentication approaches. Yet another concerns the formal specification of a structuralist semantics for the approach.

References

1. Abels, S., Haak, L., Hahn, A.: Identification of common methods used for ontology integration tasks. In: Proceedings of the First International Workshop on Interoperability of Heterogeneous Information Systems (Bremen, Germany), pp. 75–78 (2005)
2. Bashir, K.: Inheritance in O/R Mapping (2010), `http://www.alachisoft.com/articles/inheritance_mapping.html`
3. Chen, P.P.S.: The Entity-Relationship Model: Toward a Unified View of Data. ACM TODS 1(1), 9–36 (1976)
4. Chen, P.P.S.: English sentence structure and entity-relationship diagrams. Information Sciences 29(2-3), 127–149 (1983)
5. Codd, E.F.: A relational model for large shared data banks. CACM 13(6), 377–387 (1970)
6. Cover, T.M., Thomas, J.A.: Elements of information theory. John Wiley & Sons, Hoboken (2006)
7. Daft, R.L.: Organization Theory and Design, 10th edn., Cengage Learning, Florence (2009)
8. Debreceny, R., Felden, C., Ochocki, B., Piechocki, M., Piechocki, M.: XBRL for Interactive Data: Engineering the Information Value Chain. Springer, Heidelberg (2009)
9. Del Grosso, E., Missikoff, M., Smith, F., Taglino, F.: Semantic Services for Business Documents Reconciliation. In: Proceedings of Workshop ISDSI 2009 (2009)
10. Erl, T.: SOA: principles of service design. Prentice Hall, Upper Saddle River (2007)
11. Fowler, M.: Analysis patterns: reusable object models. Addison-Wesley, Reading (1997)
12. Halpin, T.: Object-Role Modeling (ORM/NIAM). In: Berners, P., Mertins, K., Schmidt, G. (eds.) Handbook on Architectures of Information Systems, pp. 88–103. Springer, Heidelberg (2006)

13. Halpin, T., Morgan, A.J., Morgan, T.: Information Modelling and Relational databases, 2nd edn. Morgan Kaufmann, Burlington (2008)
14. Harris, R., Taylor, T.J., Love, N., Versteegh, C.H.M.: Landmarks in linguistic thought, 2nd edn. Routledge, New York (1997)
15. Hay, D.C.: Data Model Patterns: Conventions of Thought. Dorset House, New York (1996)
16. IDA: European Interoperability Framework for Pan-European eGovernment Services. Version 1.0. European Commission, Brussels (2004)
17. Insfrán, E., Pastor, O., Wieringa, R.J.: Requirements Engineering-Based Conceptual Modelling. Requirements Engineering 7(2), 61–72 (2002)
18. Kent, W.: A Simple Guide to Five Normal Forms in Relational Database Theory. CACM 26(2), 120–125 (1983)
19. Object Management Group: OMG Unified Modeling Language (OMG UML), Superstructure. (version 2.3) (2003a), `http://www.omg.org/spec/UML/2.3/Superstructure`
20. Object Management Group: MDA Guide (version 1.0.1) (2003b), `http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf`
21. Ogden, C.K., Richards, I.A.: The Meaning of Meaning: A Study of the Influence of Language Upon Thought and of the Science of Symbolism. Routledge & Kegan Paul, London (1923)
22. Pokraev, S.V.: Model-Driven Semantic Integration of Service-Oriented Applications. Ph.D. Thesis. Novay PhD Research Series No. 25. Novay, Enschede (2009)
23. Sharp, A., McDermott, P.: Workflow modeling: tools for process improvement and applications development. Artech House, Norwood (2009)
24. Simsion, G.C., Witt, G.C.: Data Modeling Essentials, 3rd edn. Morgan Kaufmann, San Francisco (2005)
25. Snoeck, M., Dedene, G., Verhelst, M., Depuydt, A.M.: Object-Oriented Enterprise Modelling with MERODE. Leuven University Press, Leuven (1999)
26. UN/CEFACT: Core Components Technical Specification. Part 8 of the ebXML Framework (version 2.01) (2003), `http://www.unece.org/cefact/ebxml/CCTS_V2-01_Final.pdf`
27. W3C: OWL Web Ontology Language Guide. W3C Recommendation (February 10, 2004), `http://www.w3.org/TR/owl-guide/`
28. Wand, Y., Storey, V.C., Weber, R.: An Ontological Analysis of the Relationship Construct in Conceptual Modeling. ACM TODS 24(4), 494–528 (1999)
29. Wieringa, R.J.: Design methods for reactive systems. Morgan Kaufmann, San Francisco (2003)
30. Wikipedia: Associative Entities (2010), `http://en.wikipedia.org/wiki/Associative_Entities`
31. Wisse, P.E.: Metapattern: Context and Time in Information Models. Addison-Wesley Professional, Boston (2000)
32. Wisse, P.E.: Ontology for interdependency: steps to an ecology of information management. PrimaVera Working Paper 2007-05. University of Amsterdam, Amsterdam (2007)
33. Wisse, P.E., Oude Luttighuis, P.H.W.M., Ter Doest, H., Abrahamse, M.: Praktijkmodellering van het begrip werkgever. Forum Standaardisatie (2009) (in Dutch)
34. Wisse, P.E.: Op weg naar een stelselmatige aanpak van authenticatie en autorisatie: een gedachtenexperiment. Personal communication (2010) (in Dutch)
35. XBRL International: Extensible Business Reporting Language (XBRL) 2.1. Recommendation 2003-12-31 + Corrected Errata–2008-07-02 (2003), `http://www.xbrl.org/Specification/XBRLRECOMMENDATION-2003-12-31+Corrected-Errata-2008-07-02.rtf`