An Approach to Ontology for Institutional Facts in the Semantic Web

Robert M. Colomb e C.N.G. Dampney

Technical Report 15/02 ISIB-CNR

Padova, Italy, November, 2002

National Research Council
Institute of Biomedical Engineering
ISIB-CNR
Corso Stati Uniti, 4
35127 Padova

Robert M. Colomb

School of Information Technology and Electrical Engineering
The University of Queensland
Queensland 4072 Australia
colomb@itee.uq.edu.au

Work performed partly while visiting LADSEB-CNR; Corso Stati Uniti, 4; Padova, Italia, and partly while visiting Department of Computing, Macquarie University, Sydney, Australia

C.N.G. Dampney

School of Information Technology, University of Newcastle, Callaghan, NSW 2308 Australia

Abstract

This paper shows how it is possible to represent the complex data structures needed to support electronic commerce applications in the semantic web using ontologies. The conventional mereological or subtype-oriented refinement of the ontology is supplemented by a method of coordinated refinement based on category theory. The combined methods make ontologies a much more powerful tool for organising the semantic web.

An Approach to Ontology for Institutional Facts in the Semantic Web

Robert M. Colomb

School of Information Technology and Electrical Engineering
The University of Queensland
Queensland 4072 Australia
colomb@itee.uq.edu.au

Work performed partly while visiting LADSEB-CNR; Corso Stati Uniti, 4; Padova, Italia, and partly while visiting Department of Computing, Macquarie University, Sydney, Australia

C.N.G. Dampney

School of Information Technology, University of Newcastle, Callaghan, NSW 2308 Australia

Abstract

This paper shows how it is possible to represent the complex data structures needed to support electronic commerce applications in the semantic web using ontologies. The conventional mereological or subtype-oriented refinement of the ontology is supplemented by a method of coordinated refinement based on category theory. The combined methods make ontologies a much more powerful tool for organising the semantic web.

INTRODUCTION - ontology engineering deficiencies for institutional facts, and their remediation

Much of the motivation for the Semantic Web is to permit easy interoperability of electronic commerce and related applications. In particular, the structures are intended to support the use of agents in these types of applications.

Electronic commerce involves the exchange of messages which execute the business transactions. There is a long history of standardisation of these kinds of messages in the Electronic Data Interchange (EDI) community, and these standards are being adapted for the Web, both by being simplified and by being represented in standard structured data transport languages such as XML. A typical very simple exchange between a customer C and a supplier S might involve

- C -> S Request for quotation (RFQ)
- S -> C Quote
- C -> S Purchase Order
- S -> C Delivery Advice
- C -> S Acceptance of Delivery
- S -> C Invoice
- C -> S Payment

Each of these messages is a speech act [1]. When the supplier sends a quote, the world has changed to the extent that he is now obligated to supply that product at that price to that customer if a purchase order is received within the period of validity of the quote. Similarly, the sending of a purchase order commits the customer to accept a delivery and an invoice, and if the goods are delivered in a satisfactory state, to make a payment.

Speech acts are not arbitrary messages. Their use in organizational contexts produces what Searle [8] calls *institutional facts*. Searle distinguishes institutional facts from physical objects, what he calls *brute facts*. An institutional fact is a brute fact which has a meaning in a particular institutional context. Following the quote example, a record of the quotation message having been sent is a brute fact. The brute fact is interpreted as a record of a quotation along the lines of the previous paragraph if it was sent in the appropriate context.

Our EDI exchange has a number of more or less complex contexts. The RFQ requires that the supplier and the product both be accessible to C. The quote requires that both the customer and the product be represented in S's database, and on the terms and conditions contained in the RFQ. A purchase order depends on the existence of a quote. A delivery advice depends on a purchase order, an acceptance of delivery on the delivery advice, an invoice on an acceptance of delivery, and a payment on an invoice.

Searle's characterization of institutional facts is *(brute fact) X counts as (institutional fact) Y in context C.* Institutional facts are potentially very complex. In particular, the context C can be richly and deeply structured. Our electronic commerce example illustrates this point – the context necessary for a message transmitting a sum of money to be accepted as payment for a shipment of a product in respect of an order based on a quotation in response to an RFQ is complex indeed.

Furthermore, almost all information systems in the electronic commerce sort of domain are devoted to generating and storing institutional facts.

Ontology is a major tool in organising the semantic web. Ontology has been defined by for example the IEEE SUO Working Group[1] as

> The goal of this Working Group is to develop a standard ontology that will promote data interoperability, information search and retrieval, automated inferencing, and natural language processing. An ontology is similar to a dictionary or glossary, but with greater detail and structure that enables computers to process its content. An ontology consists of a set of concepts, axioms, and relationships that describe a domain of interest.

Ontologies that have been constructed to date are generally networks of simple terms connected by subtype or part-of relationships. Although the information systems implementing the electronic commerce applications have complexly structured data representing institutional facts, it is difficult for the ontologies to represent the structures. In particular, with only simple terms it is difficult to represent a generic speech act such as for example *business transaction* which requires a generic *customer*, *supplier* and *product*; and for this generic speech act to be refined (the term comes from the formal methods in software engineering community) to more specific structures supporting specific business domains. The terms themselves can be refined in an ontology, but refinement of the relationships requires not only some kind of subtype or part structure for the generic relationship but also the coordinated refinement of the terms.

---

[1] http://ontology.teknowledge.com/

There is a branch of mathematics, called category theory (see eg [2]), which deals with graphs and has a number of very powerful tools for the analysis and synthesis of such structures. Category theory has been proposed as a meta-ontology for the structural aspects of semantic web applications by Johnson and Dampney [5]. It has also been proposed as a foundation ontology for the IEEE Standard Upper Ontology in the Information Flow Framework[2]

Using category theory, it has been possible to propose a solution to the long-standing problem of abstraction and refinement in entity-relationship models [4]. This solution makes it possible to say in a principled way that one more detailed complex structure refines a more abstract complex structure. The method advocated is focussed on the large scale structures being refined, and does not interfere very much with the more local refinements characteristic of ontology. We therefore propose that the two methods together be used to develop the structured ontology needed for institutional facts.

In the following, we first present some terminology and a number of plausible criteria for a principled structural refinement. We then present the category-theory based refinement method in the form of a set of guidelines, and show how the method interacts with a mereology-based ontological refinement. The electronic commerce business transaction is used as a source of examples throughout.

Terminology and criteria for structural refinement

Complex structures are frequently represented in information systems using the entity-relationship (ER) method. ER modelling has a well-developed and widely understood terminology. Category theory is a branch of mathematics which can be used for representing ER models, but of course category theory has its own well-developed system of terminology which is quite different from that supporting ER, and is very much less widely known in the semantic web community. Unfortunately for the comprehensibility of papers such as this, it is very difficult to express the results using only ER terminology, so some category theory terminology is necessary. In this section we first develop the terminology, then give a plausible set of requirements for structural refinement.

The fundamental elements of ER modelling are entities and relationships. An *entity* is a set of objects in the business domain (Universe of Discourse). A *relationship* connects entities, defining a relation among their instances. A number of integrity constraints are used in ER modelling, in particular the *cardinality* of a relationship. A relationship R between two entities A and B is said to be *many-to-one* if it defines a function from A to B. A relationship is said to be *mandatory* on the side of an entity if every instance of the entity is constrained to participate in the relationship. A many-to-one relationship between A and B determines a total function if it is mandatory on the A (domain) side, and a surjective function if it is mandatory on the B (codomain or range) side.

It is always possible to represent a conceptual model using binary many-to-one relationships which are mandatory on the many side. In set theory terminology, the model is represented as a collection of domains and total functions. There are many

[2] http://suo.ieee.org/IFF/versions/20020102/IFFFoundationOntology.htm

notations for ER models. In this paper we use one which emphasises the representation as a network of functional relationships, as in Figure 1. The names in boxes are names of entities. The arrows represent relationships which are many on the side without an arrowhead and one on the side with an arrowhead. The relationships are not named, unless names are needed for disambiguation or in discussion.

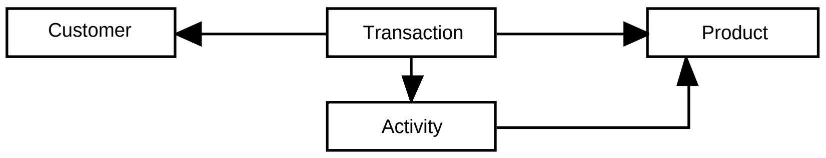

Figure 1: An ER model of a business transaction

Semantically, this model expresses at a high level a business transaction from the point of view of one of the parties. A *transaction* requires a *product* and a *customer*, and some *activity* on the part of the organisation to carry it out.

The notation of Figure 1 is that of a directed graph. Category theory (CT) is the theory of mathematical structures which can be represented by directed graphs. In CT the entities are called *objects* and the relationships *arrows*. The object at the many side is called the *source* and the object at the one side the *target* of the arrow. To be a category a collection of objects and arrows must have two additional properties – the arrows must compose associatively (functional relationships compose associatively), and associated with each object must be an identity arrow (the identity function is always possible). So Figure 1 can be viewed as a category (technically, the diagram of a category). Identity arrows are typically only represented on a diagram if they are needed for a discussion. Further, to conserve space on the page the boxes around the objects are generally omitted.

A fundamental notion in CT is that of a *functor*, a homomorphism between two categories. (A homomorphism is a functional relationship which preserves structure.) A functor takes objects into objects, arrows into arrows, and compositions of arrows into compositions of arrows. It is conventional in CT to represent functors as in Figure 2, with the domain of the functor shown above the codomain. Hence the objects in the domain carried into an object in the codomain are said to be *above* the codomain object, and similarly for the arrows.

$$\begin{tikzcd}
\text{Customer} & \text{Transaction} \arrow[l] \arrow[r] \arrow[d] & \text{Product} \\
& \text{Activity} \arrow[ru] \arrow[d, "P"'] & \\
& \text{business} \arrow[loop below, "1_{\text{business}}"] &
\end{tikzcd}$$

Figure 2: A functor

The functor P maps the category of Figure 1 into a minimal category consisting of only one entity (object) (called *business*) with only one relationship (arrow), the required identity relationship. (The convention is to name the identity relationship associated with an entity *1* subscripted by the name of the entity.) P takes each entity into *business* and each relationship into 1_{business}.

We turn now to the requirements for one ER model to plausibly be a refinement of another. Recall that the refinement under discussion is a coordinated ontological refinement of the sorts of institutional facts used in electronic commerce. We want the more abstract model to be something like a type for the more specific. As with ontology generally, this gives some of the tools for organising this aspect of the semantic web.

Since it is not common in ER modeling to discuss two related models, we need some new terminology. It is convenient to borrow some of the CT terminology. We want to say that one model R (for refinement) is a refinement of another S (for specification). Some authors advocate a method of refinement for development of ER models (eg. [3], [10], [11]), but their methods are weak in the sense that there are no properties possessed by the more abstract model which must also be possessed by the refinement. There is no well-defined way in which the detailed model can be said to satisfy a specification in the more abstract model.

We will speak of the refinement as being *above* the specification. Figure 2 therefore represents a refinement.

Requirement 1 - Integrity: Every entity in the refinement is above an entity in the specification, every relationship in the refinement is above a relationship in the specification, and every entity and relationship in the specification is refined.

This requirement says that nothing in the specification is lost, and that nothing is added that is not specified.

Requirement 2 – Composition: Every composition of relationships in the specification is refined by a composition of refinements of the relationships composed.

The relationship *Transaction -> Product* in Figure 1 is the composition of *Transaction -> Activity* and *Activity -> Product*. Every refinement of the latter two must compose into a refinement of the former. Otherwise, the functional relationships in the specification are not preserved in the refinement.

Requirement 3 – Completeness: Every entity refining a specification entity is the target of a relationship refining a relationship whose target is the specification entity. (There are alternative requirements canvassed in [4], but this version is adequate.)

Otherwise, there can be many things going on in the refinement which don't satisfy the requirements of the specification so long as there is one refinement which does. We want every refinement of *Product* to be associated with a refinement of *Activity*, and every refinement of *Activity* to be associated with a refinement of *Transaction*.

Requirement 4 – Pattern Preservation: Every entity and relationship in the refinement is part of a pattern with the same structure as the specification.

We want every refinement of *Transaction* to be associated with refinements of *Customer*, *Product* and *Activity*.

We argue that these are a close to being a minimal set of requirements to be able to talk meaningfully of refining a data model. Extended examples in [4] show that the requirements have force (it is possible to make mistakes), and at least in the examples presented can lead to a refinement which is a better design.

The refinement method

It is shown in [4] that category theory provides tools that can implement the requirements 1 – 4 for a model to be a refinement of a specification. We have already seen that both models can be viewed as categories.

Integrity is implemented by making the refinement a functor surjective on entities and relationships whose source is the refined model and whose target is the specification (the refinement is above the specification).

Composition is implemented by requiring that every relationship in the specification is refined by a relationship, possibly a composition, for each and every entity in the refinement and which is called the *cartesian relationship* with certain strong properties described below.

Completeness is implemented by requiring that every refinement of the target of a specification relationship be the target of a cartesian relationship. This is encapsulated in CT by a special type of functor called a *fibration*. The entities and relationships above a specification entity are called the *fibre indexed by the specification entity*. (Relationships whose source and target are both above the same specification entity are refinements of the identity relationship for that specification entity.)

Pattern preservation is implemented by requiring that the fibration have possibly several right inverses which are functors, and that every refined entity be related to a refined entity in one of the images of the specification. (All of the technical terms from CT are defined and illustrated in [2])

Note that functors compose, as do fibrations, so we can perform our refinement in steps, in the normal software engineering manner.

The refinement method can be presented as a set of five design guidelines:

***Guideline 1. Freedom within a fibre*:** The fibration does not itself place any restrictions on the refined model fragments within a fibre. This is because the constant functor (whose codomain is a single entity with its identity relationship, as in Figure 2) is always a fibration. This rule, or lack of rule, shows us that the guidelines being introduced are not local, so complementing the standard local refinement processes. The guidelines are in addition to the standard design rules, not simply reformulations of existing rules. Figure 2 is an example of this guideline. The refinement is entirely unconstrained.

Guideline 2. Connections between fibres must be unitary (requirements integrity and composition): A key element in the definition of a fibration is that there must be what is called a cartesian relationship above every relationship in the enterprise model for each entity above the target. A cartesian relationship has the property illustrated in Figure 3. This property is a very strong constraint, as we will see in the following.

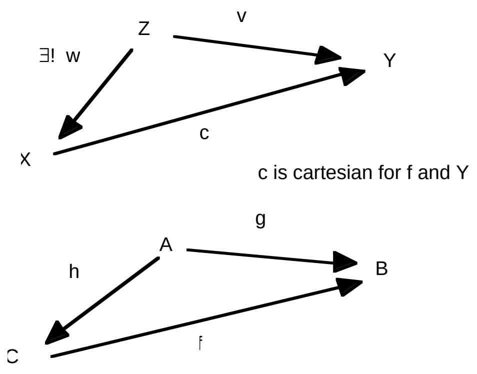

Figure 3: Cartesian relationship

The bottom diagram in Figure 3 is a fragment of the specification. The top diagram is a fragment of the refinement, with entity X in the fibre above C, Z above A and Y above B. The relationship of main interest is c:X -> Y above f:C -> B. The specification has a relationship g:A -> B which is a composition of f with h:A -> C. For c to be a cartesian relationship (for f and Y), associated with each v above g, there must be a unique w above h such that v is the composition of c with w. Note that there can be many different Ys above B.

This requirement has particular force above a single relationship of the specification, as every entity has an associated identity arrow, and all the arrows within a fibre map to the identity relationship associated with the entity the fibre is above. The situation is shown in Figure 4, where on the left the design satisfies the criterion, while on the right it does not.

What this means is all relationships originating in one fibre and terminating at a single entity in another fibre are examples of v in Figure 4. We will call an entity above the target fibre an *anchor* for relationships. We would expect many anchors, since the specification target is refined.

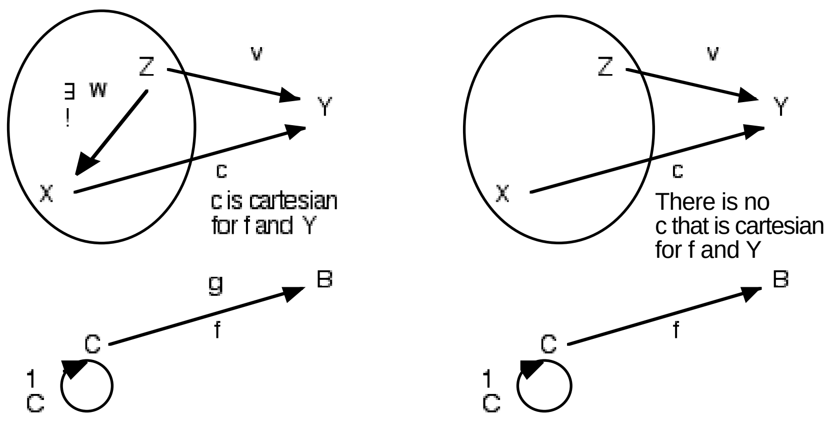

Figure 4: Cartesian relationship (left) and no cartesian relationship (right) for fibres above an identity

The constraint that the connections between fibres must be unitary is that one relationship must be cartesian for each anchor and all the others must be dependent on the cartesian relationship for that anchor in exactly one way. The source of the primary relationship is called the *centre* of the group of objects for that anchor.

Example T.1 We will develop a running example based on further refinement of Figure 2, the business transaction. We will distinguish specification entities from refined entities. Both will be named in *italics*, but the specification entities will be also in ***bold***.

Figure 5 shows a refinement of Figure 2, where we have introduced two parts to ***transaction,*** namely *order* and *delivery*. We have refined the relationships ***transaction*** -> ***customer***, ***transaction*** -> ***product*** and ***transaction*** -> ***activity***. If we want for example *delivery* to depend on *product*, the unitary connection guideline prevents us from making a direct relationship, instead we take *order* as the centre of relationships to the anchor *product*, and *delivery* must depend on *order*. This of course makes semantic sense, as the two parts of ***transaction*** must be related to each other, and the *order* part of the process occurs before the *delivery* part.

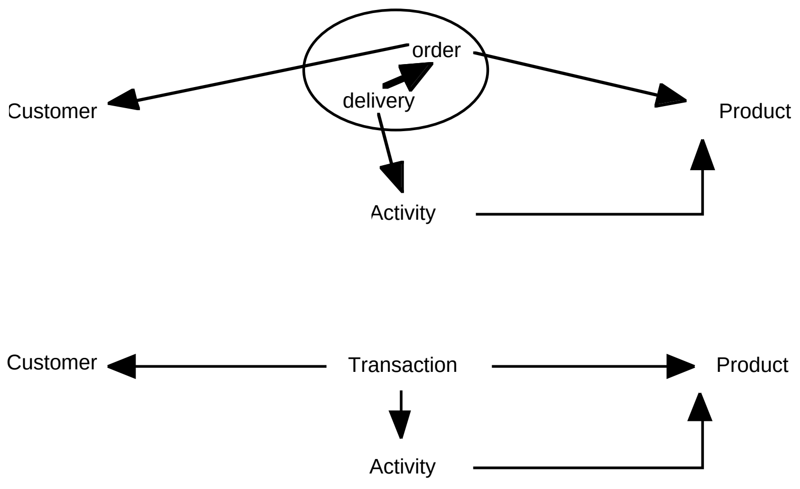

Figure 5: First refinement of ***transaction***

Guideline 3. Every entity above an entity in the enterprise model which is the target of a relationship must be the target of a relationship in the implementation model satisfying guideline 2 (requirement completeness). In other words, every entity in a fibre above the target of a specfication relationship must be an anchor for relationships above the specification relationship. This guideline comes from the definition of a fibration, which constrains the fibres which are above the targets of arrows, but not those which are above the sources of arrows.

Example T.2 If we now refine ***activity*** as in Figure 6, also into two parts, *pack* and *ship*. From guideline 2, one part must be the centre of a relationship to the anchor *product*, and since *pack* is prior, it makes semantic sense to choose it. *Ship* is dependent on *pack* for the same reason that *delivery* is dependent on *order*. Guideline 3 requires that both *pack* and *ship* be the anchors of relationships above ***delivery*** **->** **activity**. The indicated refinement satisfies this constraint, where the dependency *delivery* -> *pack* is derived.

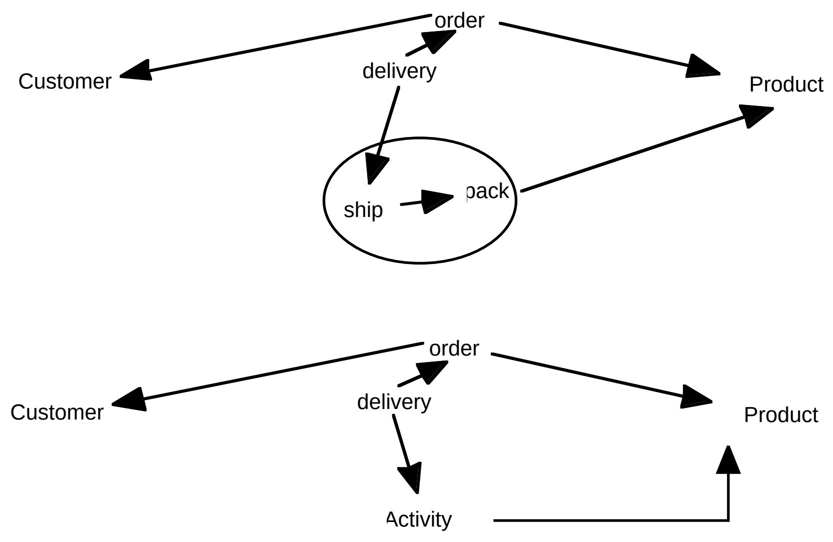

Figure 6 Refinement of ***activity***

Guideline 4. If there is a derived relationship in the specfication, relationships above it must be derived from the cartesian relationships of guideline 2. This guideline is derived from the general definition of a cartesian relationship given in Figure 3. The refinement in Figure 6 satisfies this guideline vacuously, since all the arrows shown plus the derived *delivery* -> *pack* are cartesian with respect to the refinement shown. If we view the composition of the refinements in Figures 5 and 6 as a refinement of the specification of Figure 2, the refinement also satisfies this guideline. The relationship *delivery* -> *product* above ***transaction*** -> ***product*** is not cartesian, but is derived from *delivery* -> *pack* and *pack* -> *product*, which are.

Guideline 5. If there are diverging relationships in the specification, then there must be relationships in the refinement above them whose centres are aligned (requirement pattern preservation). This guideline is a consequence of the requirement that everything in the refinement must be dependent on objects in an image of the specification.

Example T.3: The refinement of Figure 5 fails this guideline. *Delivery* has relationships with *activity*, *customer* and *product* (the latter two derived), but *order* is not the source of a relationship whose target is *activity*. The semantic force of guideline 5 in this situation is that as it stands, it is possible for an order to get lost, in the sense that no activity is ever associated with it. Think of a large organisation where the orders are taken by one department and filled by another. It would probably make the auditors happier if when an order is taken by marketing there is some acknowledgement by the shipping department recorded in the database.

The refinement of Figure 6 does not provide much help for this problem. We have refined ***transaction*** as a process with two parts, *order* and *deliver*, with the parts held together by the later having a functional relationship with the earlier. We have also refined ***activity*** as a process with two parts, *pack* and *ship*, held together similarly. To satisfy guideline 5, we must have a relationship between *order* and an entity above ***activity***. Neither of the existing entities will do, since the target of a relationship must exist prior to, or at least be created simultaneously with, the source. Issuance of a picking slip (recorded in *pack*) may occur well after the order is taken, and the issuance of a shipping order (recorded in *ship*) after that.

One way to solve this problem is to include an additional part in the process refining ***activity***, say *acknowledgement*, which is created in the same database transaction as the order. In practice, this may be simply a view on the order which is routinely checked by the shipping department. Another way is to make an inventory reservation in the same database transaction as the order, in the way airlines reserve seats when a trip is booked. A complete refinement of Figure 2 satisfying all the guidelines is given in Figure 7.

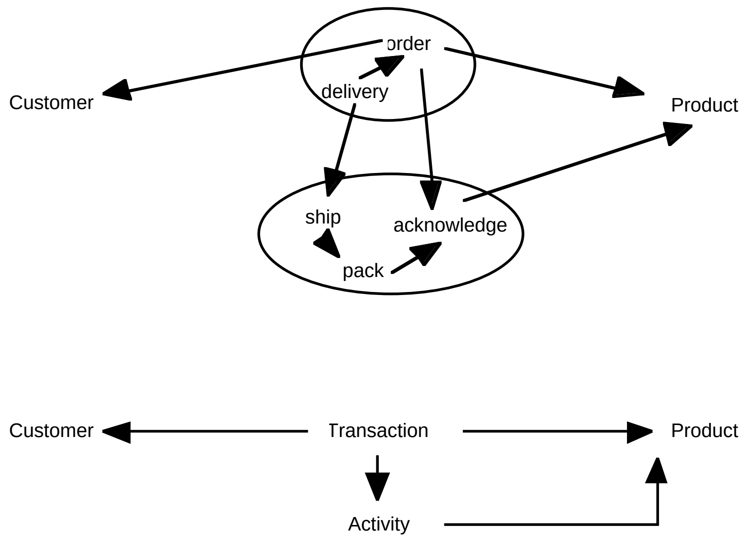

Figure 7: Refinement of Figure 2 satisfying all guidelines

Relationship with mereological refinement

Mereological refinement is a progressive articulation of a whole by defining its parts [9]. Its parts may exist all at once, or may be progressively created. The whole may be thought of as an object or a process. Parts may have parts (the whole-part relationship is transitive).

Fibrations-based structural refinement does not interfere with the mereological refinement of the individual specification entities. Simply, if there are relationships among specification entities, the structural refinement insures that the relationships are maintained as the specification entities are refined.

There are a number of issues to take into account.

- How strong the structural constraints should be – it makes sense to separate requirements 1-3 from requirement 4. We can call guidelines 1- 3 *weak structural refinement*, and the addition of guideline 4 *strong structural refinement*.
- What stages in the refinement process are considered integrity constraints and which are simply the sequence of making engineering decisions. In the example of T.1 – T.3 above, one might take the first refinement of Figure 2 to be integrity constraints and the others as engineering choices. In a larger system, there might be several layers of integrity constraints, the more refined of which may be local from the point of view of the more general specification.
- At what stage in the population of the mereological schemas we need the integrity constraints to be satisfied. In the business transaction example of Figure 7, they are always satisfied. In the refinement of Figure 6 (which fails to satisfy guideline 4), they are not. Between the placement of the order and the request to deliver there is no relationship between an instance of a refinement entity above ***transaction*** and an instance of a refinement entity above ***activity***. We will distinguish *continual* satisfaction of the constraint from *eventual* satisfaction.

These are business issues, all of which lead to technically satisfiable specifications.

In general, if we have a specification A -> B and refine both A and B by parts, we can think of the instances populating the mereological schemas as participating in chains of associations following the schema, which must be acyclic. Either the whole structure is inserted into the database in the same database transaction, or it is inserted in stages down the chains of functional dependencies (an instance of the target of a functional dependency must exist when an associated instance of the source comes into existence). We will call instances of entities which are not the target of a relationship *bottom* instances. Similarly, we will call instances of entities which are not the source of a relationship *top* instances.

The minimum constraint where structural refinement makes sense would be the eventual satisfaction of weak structural refinement. Here, the bottom instances of entities refining the target of a specification relationship must have an association with some instances of entities refining the source of the specification relationship. Eventual satisfaction of strong structural refinement requires that the top instances of entities refining the source of a specification relationship have an association with some instances of entities refining the target of the specification relationship.

Relationship with e-commerce

We began this story with a discussion of the sorts of messages used to implement institutional facts found in electronic commerce, with their complex contexts. We will finish the story by seeing how our machinery applies to these kinds of data objects.

Institutional facts are records of speech acts. Speech acts, at least the sort we are considering, involve inter-organisational or at least inter-agent communication. An organisation or an agent does not necessarily want to expose all of its internal structure to its potential communication partners. For this reason, it is usual to construct views which hide the structural details not necessary for external partners. Following [5], a view on a system E has a schema K and a homomorphism mapping K into the classifying category for E, which they call Q(E). The classifying category contains all the queries on E. For our purposes, a view looks like an information system. It is important that it can be constructed using the tools of category theory, but the details are not relevant here.

Figure 8 shows views for two parties who may participate in an e-commerce exchange such as we have discussed. Note that they share the specification for ***transaction*** and ***product***. They differ in that the purchaser has a specification for ***supplier*** while the supplier has a specification for ***customer***. The purchaser is of course only one possible customer for the supplier, and the supplier only one possible supplier for the customer. The refinement of ***product*** may be different for the two organisations, as it is very likely that a customer buys only a small fraction of any particular supplier's range, but buys from several suppliers.

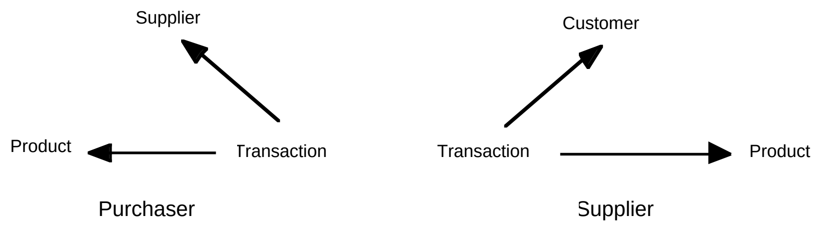

Figure 8 Purchaser's and supplier's views of each other.

The critical factor for the ability to interoperate is that the two organisations have visible in their views coordinated refinements for the data records associated with the ***transaction*** process, as we see in Figure 9. We have used the sequence *RFQ*, *Quote*, *Purchase Order*, *Delivery Advice*, *Delivery Acknowledgement*, *Invoice*, *Payment* discussed in the introduction. The process begins from the customer, so the *RFQ* has a link both to *supplier* on the customer's side and to *customer* on the supplier's side. *RFQ* also has a link to *product* on both sides. These functional relationships are necessary to establish the communication.

Both sides keep copies of all messages. This is common practice to support follow-up and to resolve misunderstandings. In addition, each side may establish relationships from their copy of a message to other structures outside the view. However, the minimum requirement is that each message has a relationship with the last. The linkages between the two organisations are always able to be derived.

Note that the specification in Figure 9 shows relationships between the organisations, using dashed lines. These relationships are necessary to conduct the business transaction. They are distinguished from the within-organisation relationships only for expository purposes. The refinement in the figure is a system which continually satisfies strong structural refinement. Note also that the relationship between *RFQ* and also *Payment* in the customer and supplier is bi-directional. This is formally necessary to satisfy requirement 3 (completeness), and semantically necessary to record (for *RFQ*) the confirmation of the initiation of a communication sequence, and (for *Payment*) the closure of the transaction for both parties.

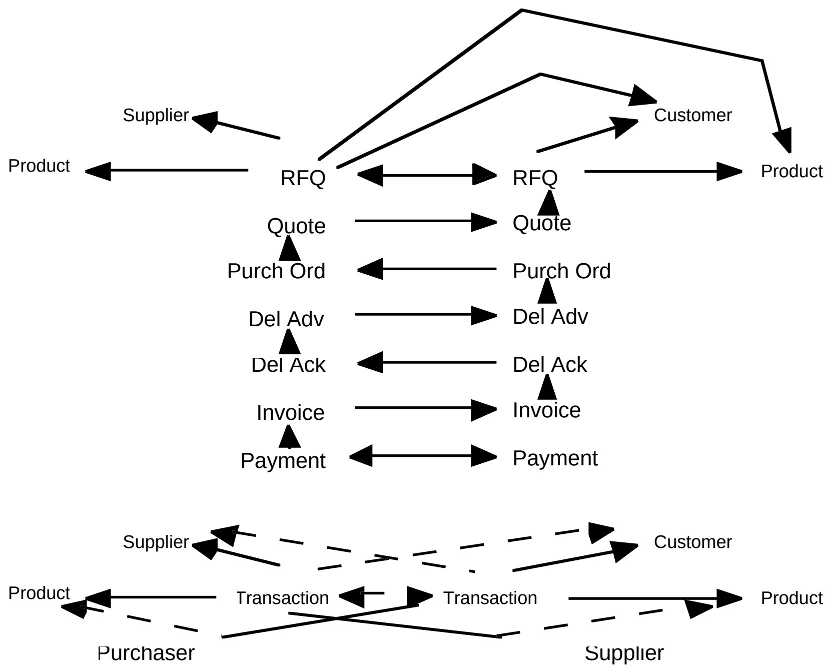

Figure 9: Coordinated refinement of ***transaction*** supporting an e-commerce application

Of course the refinement in Figure 9 is quite likely still not detailed enough to support specific interactions. The EDI standards have large and complex data structures, and exchanges supporting a particular class of business generally use only a small subset of the possible data items. Further refinement is therefore necessary, which is supported by our method.

Completing the link back to the semantic web, agents need not only to interact with each other, but also to find each other. Because interactions are not only complex but also have deep semantic content, agents do not generally seek each other on the open web using for example search engines. It is much more common to use e-commerce exchanges supporting particular industries or types of interactions. An e-commerce exchange can support much more complex data structures than a search engine.

Even using an exchange, the negotiations necessary to set up a transaction are complex. For example, the processes necessary for electronic tendering are analysed in [6]. The progressive refinements we have described in this paper can be used as an index to structure this complex data to support the protocol negotiations, in a way similar to that described in [7].

In conclusion, we have presented a method of coordinated refinement of complex data structures supporting institutional facts and the speech acts which create them, which interacts well with the more conventional local mereological and subtype refinement methods. The combined methods make ontologies a much more powerful tool for organising the semantic web.

Acknowledgements

This paper owes much to many people, but most immediately to discussions between Colomb and Chris Partridge and Nicola Guarino of LADSEB-CNR.

References

[1] Austin, J.L. How to Do Things with Words (Harvard U.P., Cambridge, Mass, 1962)

[2] Barr, M. and Wells, C. *Category Theory for Computer Scientists* (Prentice-Hall, New York, 1990).

[3] Batini, C., Ceri, S. and Navathe, S.B. *Conceptual Database Design*: *an Entity-Relationship Approach* (Benjamin/Cummings, New York, 1992).

[4] Colomb, R.M., Dampney, C.N.G. and Johnson, M. Category-Theoretic Fibration as an Abstraction Mechanism in Information Systems *Acta Informatica* 38 (2001) 1-44.

[5] Johnson, M. and Dampney, C.N.G. On Category Theory as (meta) Ontology for Information Systems Research in Welty, C. and Smith, B. (eds) *International Conference On Formal Ontology In Information Systems (FOIS-2001)* October 17-19, 2001, Ogunquit, Maine (ACM Press, New York, 2001).

[6] Kayed, A. and Colomb, R.M. Business-to-business Electronic Commerce: Electronic Tendering, in S.M. Rahman and R.J. Bignall (eds) *Internet Commerce and Software Agents: Cases, Technologies and Opportunities* (Idea Group Publishing, London, 2001) 231-250.

[7] Kayed, A. and Colomb, R.M. (2002) Using ontologies to index conceptual structures for tendering automation *Thirteenth Australasian Database Conference* (ADC2002, Jan. 28 - Feb. 1, 2002, Melbourne, Australia) 95-102.

[8] Searle, John R. *The Construction of Social Reality* (The Free Press, New York, 1995).

[9] Simons, Peter *Parts: a study in ontology* (Oxford University Press, Oxford, 1987).

[10] Simsion, G. *Data Modeling Essentials: Analysis, Design and Innovation* (Van Nostrand Reinhold, New York, 1993).

[11] Thalheim, B. *Entity-Relationship Modeling – foundations of database technology* (Springer, Berlin, 2000).