Information Management Metamodel (IMM) Specification

OMG Draft Specification: revised Submission to the Request for Proposals
ab/05-12-02

VOLUME IV

APPENDICES

Date: March 25, 2009

Draft : Version 8.0

APPENDIX A: GLOSSARY

This glossary attempts to define all terms highlighted in the Platform-independent entity/relationship model (Volume II) specification text. For the most part, sources are texts prominent in the specified field. The full description of the text is given in the "References" Section.

There are two exceptions. "OMG1" refers to terms defined in the original IMM Request for Proposal. "DH" is used for those terms defined by David Hay in the course of writing the specified section.

Highlighted words in the definitions refer to other words in the glossary.

Class names in the XML Schema Metamodel are defined in Volume III, Section 2..

Term	Definition	Section	Source
AB	(See architecture board.)		
abstract class	A class that cannot be directly instantiated. (Each instance must be an instance of a sub-type class. –DH)	7.4	[Booch, *et. al.* 1999, p. 457]
aggregation	A special form of association that specified a whole-part relationship between the aggregate (the whole) and a component (the part).	7.4	[Booch, *et. al.* 1999, p. 458]
Architect's View	(Row Three of the ***Architecture Framework***) A view of the underlying structures of Row Two, rendered in a more disciplined fashion, completing the conceptual model of the business. This is still without reference to any particular technology. (Represented by ***platform-independent*** models.)	1	[Hay 2006, p. 344]
architecture board (AB)	The OMG plenary that is responsible for ensuring the technical merit and MDA-compliance of RFPs and their submissions.	1	OMG1
architecture framework	A scheme for representing a body of knowledge systematically. In this context, it consists of six rows describing the six perspectives of actors in the development of information systems, and six columns representing the six types of information involved: what, how, where, who, when, and why. Each cell represents a way of looking at one of these types of information from a particular perspective.	1	[Hay 2006, p 344]
associative relationship	a ***binary relationship*** between entity classes that does not describe one's being a super-type of the other.	7.4	DH

attribute	A *characteristic of* a ***entity class*** that is *constrained by* a ***domain***. (An ***attribute*** is a kind of ***property***)***.***	7.4	[Hay 2006, p. 69]
binary relationship	A ***relationship*** between exactly two ***entity classes***	7.4	[Hay, 2006, p. 346]
board of directors (BOD)	The OMG body that is responsible for adopting technology.	1	OMG1

Term	Definition	Section	Source
BOD	(See board of directors.)	1	OMG1
Builder's View	(Row Five of the ***Architecture Framework***) A view of the details of a particular language, database storage specification, network, and so forth.	1	[Hay 2006, p. 346]
Business Owner's View	(Row Two of the ***Architecture Framework***) Defines—in business terms—the nature of the business, including its structure, processes, organization, and so forth. These are usually multiple business owners' views of a given enterprise, and these may overlap or even contradict each other. (Cf. ***Semantics of Business Vocabulary and Business Rules.***)	1	[Hay 2006, p. 347]
cardinality	The number of elements in a set. (cf. **multiplicity**).	7.4	[Rumbaugh, *et. al.* 1999, p.182]
cardinality constraint	The *maximum cardinalities* a role or attribute may have. (cf. ***mulitiplicity***.)	7.4	DH
class	See ***object type.***	7.4	[Martin/ Odell 1995, p.33]
class of platform (technology) independent model	(See ***platform-independent model***.)		
class of platform (technology) specific model	(See ***platform-specific model***		
common object request broker architecture (CORBA)	An OMG specification for data repository integration.	1	OMG1
common warehouse metamodel (CWM)	An OMG specification for metadata.	1	OMG1

composition	A form of ***aggregation*** with strong ownership and coincident lifetime of the parts by the whole; parts with non-fixed multiplicity may be created after the composite itself, but once created, they live and die with it; such parts can also be explicitly removed before the death of the composite.	7.4	[Booch, *et. al.* 1999, p. 459-60]
continuous domain	A domain whose values are determined by a datatype or an algorithm of some kind.	7.4	DH
CWM	(See common warehouse metamodel.)		
data definition language (DDL)	The part of SQL (typically) used for declaring information structures (Tables, Columns, Schemas) as opposed to manipulating them.	1	OMG1
data type	A descriptor of a set of values that lack identity (independent existence ad the possibility of side effects). Data types include primitive predefined types and user-definable types. Primitive types are numbers, strings, and time. User definable types are ***enumerations***.	7.4, 7.5	[Rumbaugh, *et. al.,* 1999, pp. 247-8]

Term	Definition	Section	Source
data warehouse	a *subject oriented, integrated non-volatile and* time variant collection of data in support of management's decisions.		[Inmon 1996]
Designer's View	(Row Four of the ***Architecture Framework***) A view of how technology may be used to address the information-processing needs identified in the previous rows. Here, object-oriented databases are chosen over relational ones (or vice versa), kinds of programming languages are selected (third or fourth generation, object-oriented, and so on), program structures are defines, user interfaces are specified and so forth. (This view is represented by ***platform-specific models***.)	1	[Hay 2006, p. 355]
discrete domain	A domain consisting of a discrete list of valid values. (As opposed to a ***continuous domain*** which is an algorithm of some kind.	7.4	DH
domain	[The definition of] a constraint on one or more attributes.	7.4	[Hay 2006, p. 356]
ECORE	Native metamodel format for the Eclipse Modeling Framework (EMF)http://www.eclipse.org/emf. It is generally interchangeable with the Essential MOF (EMOF) compliance level of MOF2	1	OMG1
elementary entity class	The definition of a [basic] thing of significance to the organization about which it wishes to hold information. This is as distinct from ***virtual entity class*** that is derived from other ***entity classes.***	7.4	[Hay 2006, p. 356]
entity	A thing or object of significance, whether real or imagined, about	7.4	[Barker 1989, p. 3-1]

	which information needs to be own or held.		
entity class	The definition of something of significance to an organization. An entity type has as predicates ***attributes*** and ***relationships*** with other entity types. .	7.4	[Hay 2003, p. 409]
entity type	(See ***entity class***)	7.4	
foreign key	A combination of ***columns*** in a ***table,*** where the value of each is equal to [that of] a ***primary key*** column in another table. The set of ***columns*** that constitute the ***foreign key*** must correspond to the set of ***columns*** that constitute the ***primary key*** of the other table.	7.5	[Hay 2008, p. 360]
Functioning System	(Row Six of the ***Architecture Framework***) The view of existing programs, databases, procedures, and responsibilities.	1	[Hay 2008, p. 361]
generalization relationship	a ***binary relationships*** that describes one entity class's being a ***super-type*** of another. the second is a ***sub-type*** of the first.	1	DH
IDEF1X	Commonly used notation for logical models of relational databases, standardized by NIST.	1	[IDEF1X FIPS184] and [IDEF1X Hayes]

Term	Definition	Section	Source
IDL	(See interface definition language.)		
IE	(See information engineering.)		
information engineering (IE)	Widely used traditional software development method, focused on data analysis. Includes a commonly used data modeling notation, most famous for using 'crows feet' to represent multiplicity.	1	OMG1
interface definition language (IDL)	An OMG and ISO standard language for specifying interfaces and associated data structures		OMG1
letter of intent (LOI)	A letter submitted to the OMG BoD's Business Committee signed by an officer of an organization signifying its intent to respond to the RFP and confirming the organization's willingness to comply with OMG's terms and conditions, and commercial availability requirements	1	OMG1
LOI	(See letter of intent.)		
mapping	Specification of a mechanism for transforming the elements of a model conforming to a particular metamodel into elements of another model that conforms to another (possibly the same) metamodel.	1	OMG1
meta object facility (MOF)	An OMG standard, closely related to UML, that enables metadata management and language definition.	1	OMG1

metadata	The data that describe the structure and workings of an organization's use of information, and which describe the systems it uses to manage that information		[Hay, 2006]
metamodel	A representation of a body of ***metadata***	1	[Hay, 2006]
MOF	(See meta object facility.)		
multiplicity	The range of possible cardinalities a set may hold. Specifically, the range of cardinalities of a role or attribute. Commonly, the *maximum cardinalities* a role or attribute may have. (cf. ***cardinality constraint***)	7.4	[Rumbaugh, *et. al.* 1999, p.182]
multi-variate relationship	A ***relationship*** between three or more ***entity classes.***	7.4	[Hay 2006,p. 369]
n-ary relationship	(See ***multi-variate relationship*** .)		
navigation direction	A characteristic of a relationship that describes which role is expected to be most commonly used.	7.4	DH

Term	Definition	Section	Source
normative	Provisions that one must conform to in order to claim compliance with the standard. (as opposed to non-normative or informative which is explanatory material that is included in order to assist in understanding the standard and does not contain any provisions that must be conformed to in order to claim compliance).	1	OMG1
normative reference	References that contain provisions that one must conform to in order to claim compliance with the standard that contains said normative reference.	1	OMG1
object	Anything to which a concept applies. It is an instance of a concept.	7.4	[Martin/ Odell 1995, p. 26]
object type	A concept. (i.e., it is a notion or and idea we apply to the objects in our awareness.)	7.4	[Martin/ Odell 1995, p.34]
optionality constraint	The minimum cardinality a role or attribute may take. Is a value required for an instance of an entity class?	7.4	DH
ordered relationship	[One in which] the set of objects at one end of an association are in an explicit order.	7.4	[Booch, *et. al.* 1999, p. 459-60]
package	A general purpose mechanism for organizing elements into groups. (Cf. ***subject area.***)	7.4	[Booche, *et. al.* 1999, p. 464]
planner's view	(Row 1 of the Architecture Framework) A view that defines	1	[Hay 2006, p.

	the enterprise's direction and business purpose. This is necessary to establish the context for any system development effort. It includes definitions of the boundaries of systems or other development projects. (Cf. ***architecture framework.***)		375]
platform	A set of subsystems/technologies that provide a coherent set of functionality through interfaces and specified usage patterns that any subsystem that depends on the platform can use without concern for the details of how the functionality provided by the platform is implemented.	1	OMG1
platform-independent models	Entity/relationship models that represent the semantics and structure of the business, without respect to the technologies that might be used to store and process its components.	1	[McGilvray 2008, p. 48]

Term	Definition	Section	Source
platform-specific models	Models of data structure that reflect the technology that will be used to manage it. Currently this includes UML design models, relational database designs, and XML Schemas, among others.	1	DH
primary key	In relational theory, the set of ***columns*** whose values can be used to uniquely identify each ***row*** (*tuple* in Dr. Codd's original terminology) in a ***table*** (*relation* to Dr. Codd).	7.5	[Hay 2008, p. 376]
property	A named value denoting a characteristic of an element. [Booch, *et.al.* 1999, p. 465] Specifically, an attribute value or a role value for an entity class. [DH]	7.4	
relationship	A structural association between two ***entity types***. It consists of two or more ***roles.***	7.4	[Hay 2003, p. 430]
reverse polish notation	A method of algebraic expression whereby the operators follow their operands; for instance, to add three and four one would write "3 4 +" rather than "3 + 4". If there are multiple operations, the operator is given immediately after its second operand; so the expression written "3 − 4 + 5" in conventional infix notation would be written "3 4 − 5 +" in RPN: first subtract 4 from 3, then add 5 to that.	7.4	Wikipedia, Reverse Polish Notation August 22, 2008
role	The specific association of one ***entity type*** with another.	7.4	DH
semantics of business vocabulary and business rules (SBVR)	An Object Management Group specification to define the vocabulary and rules for documenting the semantics of business vocabulary, business facts, and business rules. This addresses the ***business owners' views*** of the ***architecture***		

	***framework*.**		
subject area	An area of interest to a particular individual or group. Used to organize entity classes. An entity class can appear in more than one subject area. (Cf. ***package*.**)	7.4	DH
sub-type	an ***entity class's*** relationship to another where an instance of the ***entity class*** is an instance of another ***entity class*** (called the ***super-type***).	7.4	[Hay 2006, p. 382]
super-type	an ***entity class's*** relationship to one or more other ***entity classes*** [where] an occurrence of the first is also an occurrence of exactly one of the others.	7.4	[Hay 2006, p 383]
table	A two-dimensional array fo data, consisting of one or more rows representing instances of the thing the table describes ad one or more columns, each containing a [value for a] kind of data describing that thing.	7.5	[Hay 2008, p. 384]
unique identifier	The fact that the values of a specified set of ***attributes*** and relationship ***roles*** are sufficient to uniquely identify each instance of an ***entity class.***	7.4	[Hay 2006, p. 386]
virtual entity class	An ***entity class*** that is [defined in terms of] other ***entity classes, attributes,*** and ***relationships.***	7.4	[Hay 2006, p. 356]
visibility	How a name can be seen and used by others.	7.4	[Booch, *et. al.* 1999, p. 468]

APPENDIX B: REFERENCES

Adams, D. 1980 *The Restaurant at the End of the Universe* (New York: Pocket Books). P. 38.

Ambler, S. 2008. "A UML Profile for Data Modeling". (Agile Data). *http://www.agiledata.org/essays/umlDataModelingProfile.htm.l*

_____ "UML Profile for Data Modeling". (Universitiet Antwerpen)

Barker, R. 1989. *CASE Method: Entity class Relationship Modelling*. (Wokingham, England: Addison-Wesley).

Booch, G., Rumbaugh, J., Jacobson, I. 1999. *The Unified Modeling Language User Guide*. (Reading, MA: Addison-Wesley.

Bruce, T.A. 1992. *Designing QualityDatabases with IDEF1X Information Models*. (New York: Dorset House).

Carlson, D. 2001. "Modeling XML Vocabularies with UML: Part III". (O'Reily XML.com). http://www.xml.com/pub/a/2001/10/10/uml.html

Cattell, R.G.G., Barry, D.K., *et al.* 2000. *Object Data Standard* (Boston: Morgan Kaufmann)

Chen, Peter Chen, P. 1976. "The Entity class-Relationship Approach to Logical Data Base Design". The Q.E.D. Monograph Series: Data Management. Wellesley, MA: Q.E.D. Information Sciences, Inc. p. 17.

This is based on his article, "The Entity class-Relationship Model: Towards a Unified View of Data", ACM Transactions on Database Systems, Vol. 1, No 1, (March 1976), pages 9-36.

COBOL. 1989-2002. ISO 1989:2002 - Programming Language COBOL

Cowen, J. Tobin, R. 2004. "W3C Recommendation 4". *XML Information Set (Second Edition)*. (W3C)

ECLIPSE. 2008 "Model Development Tools (MDT) - XSD". *http://www.eclipse.org/modeling/mdt/?project=xsd#xsd*

_____. 2006 "HyperModel 2.0.0**".** (XMLModeling.com). http://update.xmlmodeling.com/updates/index.html

FFE Software, Inc. 2005. "SQL Tutorial – Version 2.1". *http://www.firstsql.com/tutor.htm*

Finkelstein, C. 1992. *Information Engineering: Strategic Systems Development*. (Sydney: Addison-Wesley).

Fowler, M. 2000. *UML Distilled Second Edition.* (Reading, Ma: Addison-Wesley).

Gornik, D. 2005. "UML Data Modeling Profile". (Somers, NY: Rational). *http://www.jeckle.de/files/RationalUML-RDB-Profile.pdf*

Halpin, T. 2008. *Information Modeling and Relational Databases*. (Boston: Morgan Kaufmann).

Hay, D. 1995. *Data Model Patterns: Conventions of Thought.* (New York: Dorset House).

_____ 2003B. "A Comparison of Data Modeling Techniques (IDEF1X, Information Engineering)". (Essential Strategies, Inc.) http://www.essentialstrategies.com/publications/modeling/compare.htm.

_____ 2003A. *Requirements Analysis: From Business Rules to Architecture*. (Upper Saddle River, NJ: Prentice Hall PTR).

_____ 2006. *Data Model Patterns: A Metadata Map* (Boston: Morgan Kaufmann).

Hoberman, S. 2008. "Leveraging the Industry Logical Data Model as Your Enterprise Data Model", TeraData white paper.[undated] *(www.teradata.com/t/pdf.aspx?a=83673&b=162511). page 7.*

McFadyen, R. 2008. "Introduction to Entity Relationship Modeling". http://io.uwinnipeg.ca/~rmcfadye/2914/ERD/introEntityRelationshipModeling .htm

McGilvray, D. 2008. *Information and Data Quality Improvement: Ten Steps to Quality Data and Trusted Information*. (Boston: Morgan Kaufmann).

Object Management Group (OMG) 2008-a. *Model Driven Architecture*. *http://www.omg.org/mda/specs.htm*. January 22, 2008.

_____ 2008-b. *Semantics of Business Vocabulary and Business Rules (SBVR).* Version 1.0. http://www.omg.org/docs/formal/08-01-02.pdf. January, 2008

_____ 2003. *The Common Warehouse Metamodel*. Version 1.1. http://www.omg.org/docs/formal/03-03-02.pdf. March, 2003.

Rumbaugh, J., Jacobson, I., Booch, G. 1999. The Unified Modeling Language Reference Manual. (Reading, MA: Addison-Wesley).

Silingas, D., Kaukenas, S. 2008. "Applying UML For Relational Data Modeling". http://www.magicdraw.com/files/articles/-Sep04%20Applying%20UML%20for%20Relational%20Data%20Modeling.htm?-NMSESSID=6fa86909edc2eebf845d068413d73d6e.

Simsion, G.C., Witt, G. 2005. *Data Modeling Essentials, Third Edition*. (Boston: Morgan Kaufmann).

U.S. Department of Commerce. 1993. "Integration Definition For Information Modeling (IDEF1X)". Federal Information Processing Standards Publication 184. *http://www.itl.nist.gov/fipspubs/idef1x.doc*

Whitemarsh Information Systems Corporation, 2008. "SQL Standards" *http://www.wiscorp.com/SQLStandards.html*

Wikipedia. 2008. “Reverse Polish Notation”.
http://en.wikipedia.org/wiki/Reverse_Polish_notation

APPENDIX C: THE SEMANTICS OF BUSINESS VOCABULARY AND RULES SPECIFICATION AND THE CONCEPTUAL ENTITY/RELATIONSHIP MODEL

Appendix C is under review by the IMM and SBVR teams to better state the relationship between IMM and SBVR including transformation of a SBVR based vocabulary (part of EU Rent example) to its ER model equivalent. This section will be updated to address the above for the final IMM submission projected for June 2009 OMG meeting.

This section is intended to address the intersection between the models produced under the Semantics of Business Vocabulary and Business Rules (SBVR) specification and those that would be produced as platform-type-independent entity/relationship models. The former is intended to represent the business owners' perspectives on data (Row Two, Column One in John Zachman's "Framework for Enterprise Architecture" [Hay, 2003, pp. 1-6]), while the latter represents the architect's perspective on the same column. Since the perspectives are different, a transformation process will be required when dealing with any particular area of interest.

It is not the intention here necessarily to present an automatable process. Indeed, it is clear that considerable judgment and intuition are required to carry out the steps described. But these are the steps, and it is for people cleverer than your authors to say how any automation might be done.

The examples that follow are taken from the SBVR specification case study for a hypothetical car rental company, called "EU-rent". See "Annex E" of the SBVR specification for the full description of the case study.

Among other things, this model is the source of the examples used in the preceding section.

1. DEFINITIONS

1.1 SBVR

In the Semantics of Business Vocabulary and Business Rules (SBVR) specification, a "business vocabulary" is defined as the set of "all the specialized terms, names, and fact type forms of concepts that a given organization or community uses in their talking and writing in the course of doing business." [OMG 2008, p. 220]

It has always been the business analyst's job to try to capture that vocabulary and rationalize it so that it can become the basis for systems development. SBVR formalizes that process of capturing the vocabulary.

The problem with the vocabulary as expressed by human workers and captured by analysts is that it is often not consistently expressed by the various players, and it often neglects to express concepts that are important.

SBVR addresses the former by providing the basis for defining tools to capture and organize vocabulary and business rules. It provides the ability to define concepts and categorize them at least hierarchically, and to define relationships among them. Specifically, it provides "A basis for identification and/or definition of individual entity classes, events and states, the relationships among them, and their relationship to time, for text document and data mining" [OMG 2008, p. 221]

It does not explicitly provide a systematic means for identifying omissions.

1.2 TERMS AND CONCEPTS

SBVR is based on the following concepts:

Expression – things used to communicate (e.g., sounds, text, diagrams, gestures), but apart from their meaning — one expression can have many meanings.

Representation – the connection between expression and a meaning. Each representation ties one expression to one meaning.

Meaning – what is meant by a word (a ***concept***) or by a statement (a ***proposition***) – how we think about things.

Extension – the things to which meanings refer, which can be anything (even expressions, representations, and meanings [themselves] when they are the subjects of our discourse). [OMG 2008, p. 17]

A business vocabulary, then is a set of ***representations*** that are each *the use of* an ***expression*** *to describe* a ***meaning***.

The SBVR specification sub-divides instances of ***meaning*** into ***concepts*** and PROPOSITIONS. It then specifically defines a "noun concept" as either an "individual concept", a "general concept" (otherwise known as an "object type"), or a "role". In our transformation, described below, for the most part, "general concepts" will be represented by entity classes, and "roles" will be represented by relationship ends.

1.3 ALETHIC AND DEONTIC BUSINESS RULES

The SBVR specification goes beyond simple vocabulary to describe business rules in the form of ***propositions*** (the second kind of "meaning"). Propositions are of two types: ***Alethic propositions*** assert that something is necessary (that it "must be") or possible (that it "may be"). ***Deontic propositions*** assert that something is obligated (by some authority), or that it is prohibited or permitted. [OMG 2008, p. 97-98]

Thus, in addition to a business vocabulary (as just described), the second output of SBVR is a set of business rules expressed as propositions. In both cases, these are expressed in terms understood by the business, but with some degree of additional organization.

According to the SBVR specification, "alethic modality" is "historically, any of the five central ways or modes in which a given proposition might be true or false: ***necessity*** (and non-necessity), ***possibility*** (and impossibility), and ***contingency***" [OMG 2008, p. 110].

Propositions exhibiting alethic modality (that is to say, alethic business rules) can usually be represented in entity/relationship models.

Also according to the specification, "deontic modality" is "any of the five central ways or modes in which one might think of the social desirability of a certain other person(s)'s making true some proposition, that is, the social desirability that the act(s) be performed, by a certain other person(s), that would make the proposition true; viz., ***obligation*** (and its negation, ***non-obligation***), ***permission*** (and its negation, ***non-permission*** (***forbidden/prohibition***)), and ***optionality"*** [OMG 2008, p. 111].

Propositions exhibiting deontic modality cannot readily be translated into entity/relationship model elements. These propositions are probably expressed in terms of the entity classes in the model, but they must be described separately from the entity/relationship model.

1.4 ENTITY/RELATIONSHIP MODEL

This essay is concerned with transforming the results of an SBVR effort into an entity/relationship model that describes the business graphically. This model is called variously a "conceptual model" [Simsion & Witt 2005, pp. 16-17], or a "logical model" [Hoberman, p. 7]. In the context of the OMG's "Model Driven Architecture", this is called a "platform-type-independent" model. [OMG 2008-a, p. 149]

2. THE TRANSFORMATION FROM SBVR TO A CONCEPTUAL ENTITY/RELATIONSHIP MODEL

The transformation described here focuses on addressing nouns and alethic propositions. According to the SBVR specification, "a term [in a vocabulary] is a symbol that is for a concept and that is a word or phrase. A term is typically a common noun or noun phrase, unless it is a name." [OMG 2008-b, pp. 19-20] The assignment then is to examine the terms (the common nouns at least) to determine which elements they describe in an entity/relationship model. This involves the following steps (in no particular order):

Convert nouns to instances of entity classes or sub-types of generalized entity classes.

Convert Alethic Rules to entity/relationship roles.

Convert nouns to entity classes and roles.

Convert nouns to attributes (and relationships).

Identify omissions.

Note that this essay does not take up the issue of whether and to what extent these transformations may be automated. This is a follow-on discussion beyond the scope of what is being presented here.

2.1 CONVERT NOUNS TO INSTANCES OF ENTITY CLASSES AND SUB-TYPES OF GENERALIZED ENTITY CLASSES.

The SBVR specification defines a "noun concept" as either an "individual concept", a "general concept" (otherwise known as an "object type"), or a "role" [OMB 2008-b, pp. 19-20]. The first step in the process is to eliminate nouns that are simply individual concepts. These will become instances of entity classes. Some are obvious, such as Rhode Island, John Smith, or "my car".

Others are less so. An oil production company, for example, might have references to "oil well", "dehydration plant", and "pumping facility". From one perspective, these might be individual concepts. But they may also be what the SBVR specification calls "general concepts", requiring attributes and multiple instances of each. In fact, many of the characteristics of these things are similar. It may be useful to create an abstraction (called, for example, "FACILITY") to incorporate these three. Each of the original things is then an entity class that is a *sub-type* of FACILITY. Having done that, and gotten agreement with the business community that it is appropriate to group them this way, you can ask the question, "by the way, are there any other FACILITIES that you haven't mentioned?" As often as not, there are.

This is part of an overall strategy of looking for patterns in the terminology of the business people. To be sure, the language captured has to be visible to them as sub-types, or relationship names. But the underlying structure of any organization is invariably much simpler than the company's language would suggest. It is the analyst's job to identify that underlying simplicity and capitalize on it whenever possible.

This both makes the conversation with the business people easier, and leads to systems that are much simpler and more robust.

2.2 CONVERT ALETHIC RULES TO ROLES

The alethic business rules, however, can be represented in the model. For example, here's a sample business rule for EU-Rent:

> Car manufacturer *supplies* car model [OMG 2008-b, p. 290]

In the absence of other information, this appears to mean (in entity/relationship terms):

> Each CAR MANUFACTURER may be *the supplier of* one or more **car model**s.

Accompanying that rule is the inverse rule:

> **Necessity:** Each car model *is supplied by* exactly one car manufacturer

This second form is flagged as a necessity, suggesting that what it is actually saying is:.

> Each **car model** must be *supplied by* exactly one CAR MANUFACTURER.

The results are shown in Figure 1.

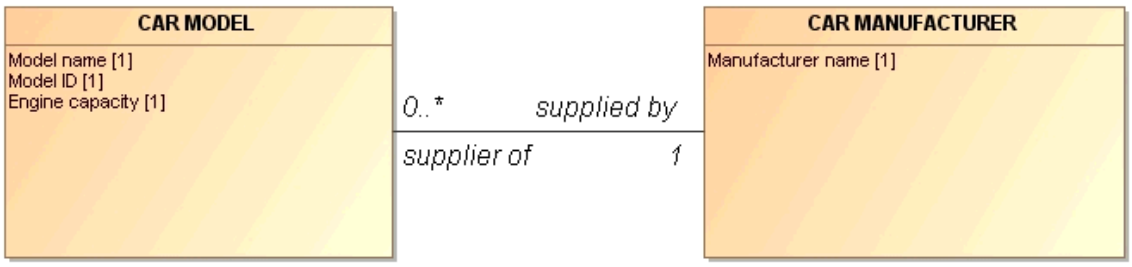

Figure 1: Car Manufacturers

In terms of the SBVR specification, if the role going in the second direction is a *necessity* (must be), then in the absence of that designation, the role going in the first direction is a *possibility.*

In another example, on page 269 of the EU-Rent specification is the alethic proposition:

> It is necessary that each rental *has* exactly one requested car group [OMG 2008-b, p. 269].

An equivalent statement can be constructed in the entity/relationship format, but this one requires some reformulation. Specifically, the noun phrase "requested car group" is actually—in SBVR terms—the role "requested" combined with a general concept "car group". "Has" is a very weak relationship name, which does not convey any meaning. A stronger statement more directly asserts that:

> Each **rental** must be *a request for* exactly one CAR GROUP.

This is shown in Figure 2. Among other things, if it is not true, a business person will find it difficult to accept the statement. The first statement above could leave h' not quite registering the implications of what is being said. .

Note that the inverse is not made explicit in the SBVR specification, but the entity/relationship requires it to be so:

> Each CAR GROUP may be *requested in* one or more **rental**S.

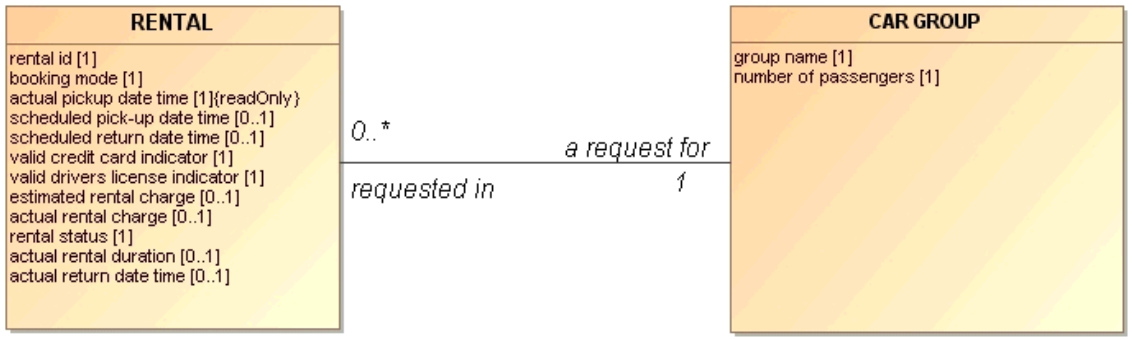

Figure 2: Two Alethic Business Rules

Note that “necessity” suggests the form “must be”, but this is not always the case. For example, the above assertions were made on page 269 of the SBVR specification. Further down, on page 299, we find the following:

Car model *is requested for* rental

> **Synonymous Form:** rental requests car model
>
> ***Necessity:*** Each rental *requests* at most one car model.
>
> ***Possibility:*** The car model *specified for* an advance rental *is changed before* the actual pick-up date/time *of* the advance rental.
>
> ***Necessity:*** The car model *specified for* an advance rental *is* not *changed after* the actual pick-up date/time *of* the advance rental [OMG 2008-b, p. 299]

In this case, the statement above is misleading: “Necessity” suggests “must be”, but “at most one” means that it might be 0.

The diagram in Figure 3 as an elaboration the one in Figure 2:

> While each **rental** must be *a request for* exactly one CAR GROUP, each **advance rental** may be *a request for* exactly one (that is, no more than one) **car model**.
>
> Each **car model** may be *requested by* one or more **advance rental**s.
>
> This is consistent with a statement in the introduction to the case study, *to wit:* “A rental booking specifies the car group required.. Optionally, the reservation … may request a specific car model within the required group.” [OMG 2008-b, p. 267]

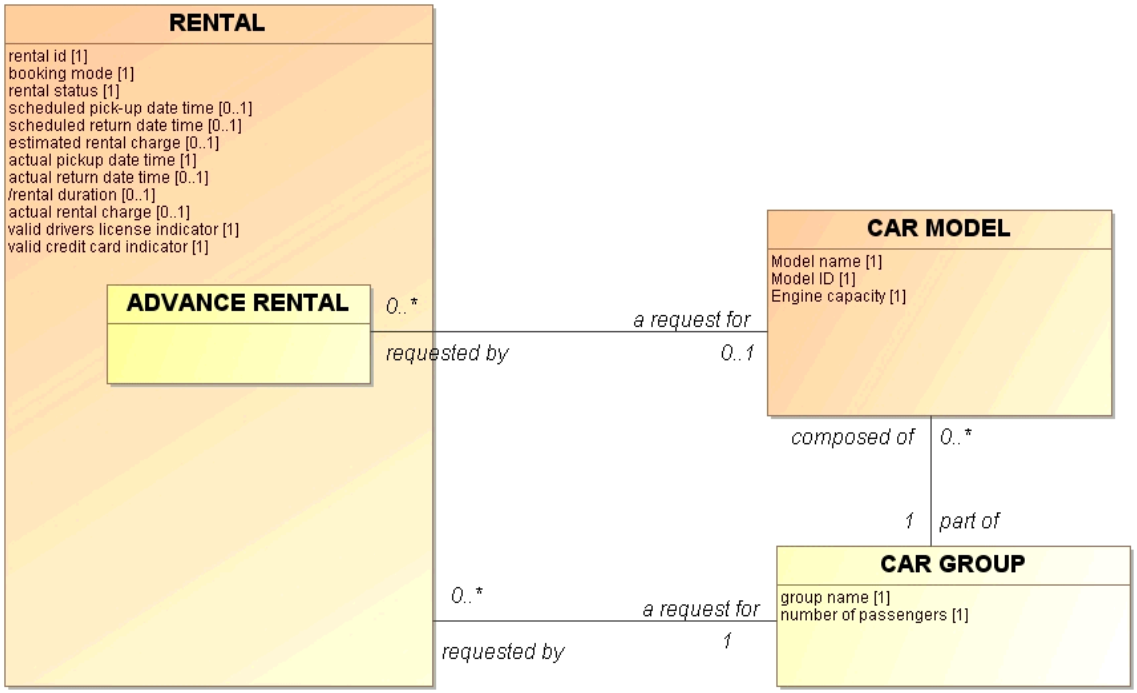

Figure 3: Different Alethic Business Rules

In general, ***deontic*** business rules (describing obligation or permission) cannot be described in an entity/relationship model. These must be documented behind the scenes. That deontic rules cannot be expressed directly in an entity/relationship model is probably due to the relational theory heritage of data modeling, where a database can only describe structure. (See Simsion/Witt quote on page **Error! Bookmark not defined.**, above.)**Error! Reference source not found.** Any constraints that are not structural will require program code. For example, for a car rental agency, a deontic rule might be:

> It is obligatory that the rental duration *of* each rental *is at most* **90 rental days** [OMG 2008-b, p. 269]

The data model must of course describe that each **rental** has the attribute "Rental duration", but it cannot assert anything about the permitted values of that attribute.

Some *sets* of deontic business rules can be described in the model, however, in that the parameters that control them may be represented by entity classes. In this case, the value of the constraint would be data managed by the business, even as the engine that makes use of it is a program.

For example, MAXIMUM DURATION could be defined as "a parameter constraining the length of time **rental**S that are *an example of* a particular **rental** TYPE may be kept.

This is shown in Figure 4. Attributes of MAXIMUM DURATION include not only its "duration value" (and unit of measure), but also "effective date" and "until date". That is why, over time, each **rental** TYPE may be *subject to* one or more MAXIMUM DURATIONS.

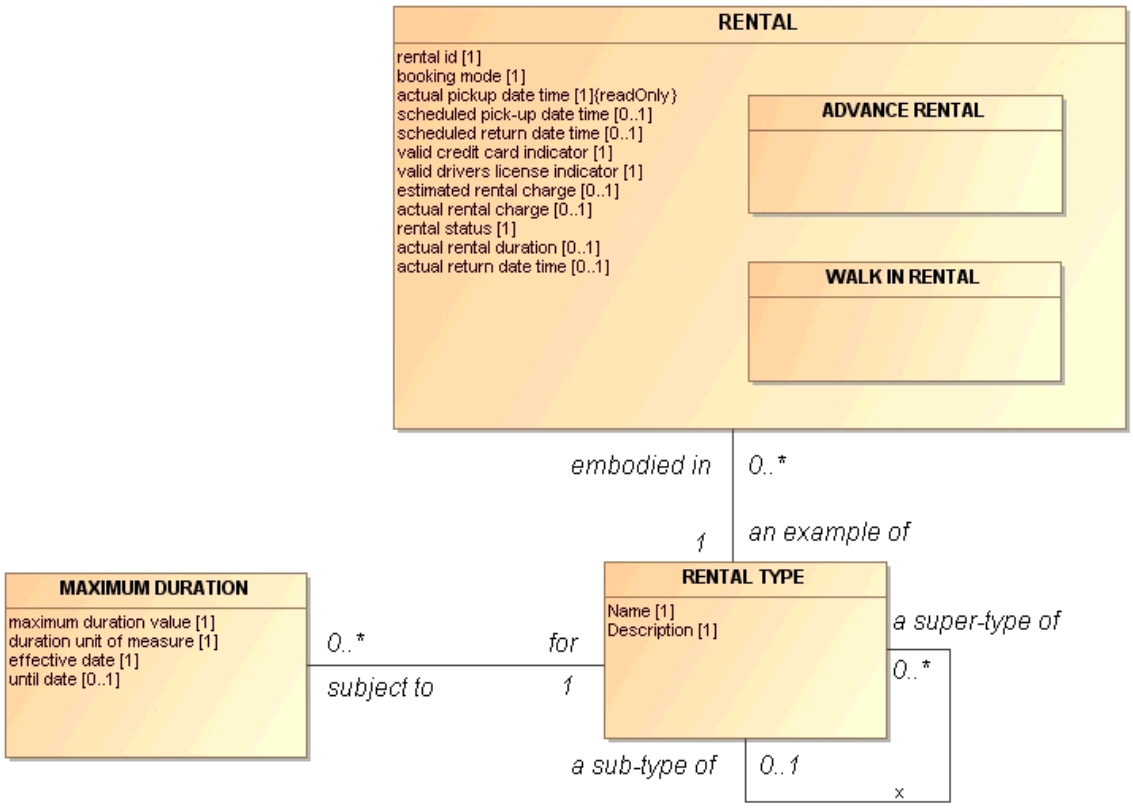

Figure 4: Maximum Duration

As an alternative, "Maximum Duration" could be added as an attribute of **rental** TYPE.

The deontic rule, then, could be expressed as:

> It is obligatory that the rental duration *of* each **rental** *is at most* the **"duration value"** of the MAXIMUM DURATION that is *for* the **rental** TYPE embodied in the **rental** and whose "effective date" is before the "scheduled pickup date" of the **rental**, and whose "until date" is on or after after the "scheduled pickup date" of the **rental** or is <null>.

Even with this approach, however, the rules themselves must be described in natural language outside the model. The values manipulated by the rules are instances contained in the data model.

2.3 CONVERT NOUNS TO ENTITY CLASSES AND ROLES

Of general noun concepts remaining after individual noun concepts are removed, many will appear on the face of it to represent entity classes. PRODUCT and PROJECT, for example are likely to represent entity classes. Beware, however. It is common for people to use nouns that encode roles being played into their meaning. These include such things as "customer", "vendor", "employee", and so forth. A "customer" however, is not a thing in its own right. You were not born a customer. You are an instance of entity

class **person**. Your "customerness" is a *role* that you sometimes play with respect to a particular business contract or transaction. Similarly, you are not inherently an employee. You play the role of employee when you contract with your employer to provide work in return for compensation (usually money). In many companies, some PEOPLE who are employees are also customers. In a database, it would not be appropriate to have records for the same person in both an employee table and a customer table.

Thus, the true definition of "customer" is "a **person** (or an **organization**) that is *a customer in* an **order**". "Customerness" is conveyed in such a model by the *relationship* between **person** or **organization** and **order**. It is not an entity class in its own right. Similarly, a "vendor" is "a **person** (or an **organization**) that is *a vendor in* an **order**. Note that the same person or organization may be a vendor to the company in one order and a customer to the company in a different order.

That is, the noun concept "customer" is an instance of a role. It is instantiated by the proposition, "each **person** may be *a customer in* one or more **order**S."

As an example, in EU-Rent, the noun concept "return branch" is defined as follows:

return branch

Concept Type: role

Definition: branch stipulated in the rental contract for return of the rented car
Note: If the renter does not return the car to the location of this branch, a penalty charge will be levied.

Necessity: Each rental *has* exactly one return branch *at* a given date/time.

Possibility: The return branch *of* a rental *is changed before* the actual return date

"Return branch" would appear to be an entity class, but the definition is actually saying that it is a "role". Thus, the entity class is in fact BRANCH. Figure shows a model of the "Necessity" assertion:

"Each **rental** must be *stipulated to be returned to* exactly one BRANCH."

Note that this specification does not include the ***possibility*** that is the inverse relationship:

Each BRANCH may be *a return point for* one or more **rental**S.

This is in fact shown in the entity/relationship model in Figure 5.

Some notations have the ability to add a symbol for "transferability" to show that the BRANCH to be returned to could be changed over time. UML, however, does not. So the "possibility" rule stated above must be documented separately.

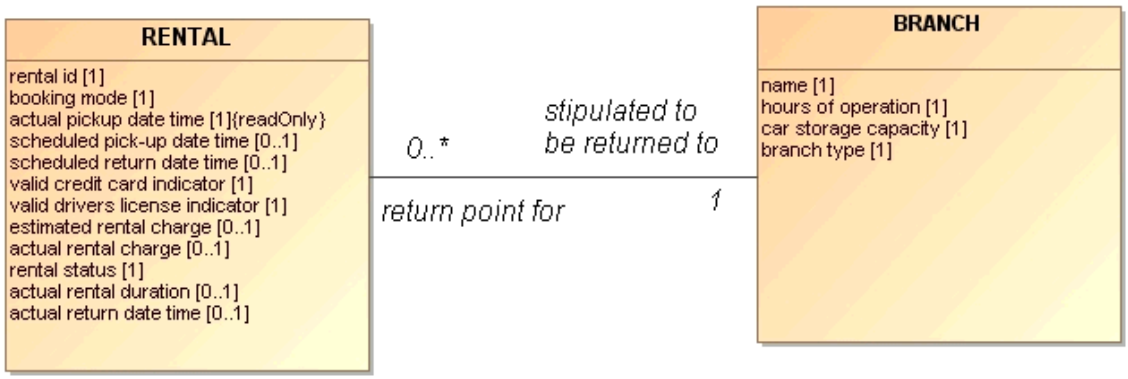

Figure 5: Return Branch

Separately the specification addresses the actual movement of cars to and from branches (shown in Figure 7, below), so if the BRANCH is only *stipulated to be* the drop-off point, the model as shown might be adequate. Note that only the latest return branch can be shown at any point in time.

If, on the other hand, maintaining a history of the drop-off stipulations is important, an intersect entity class can be added to the model to represent this. This is shown in Figure 6

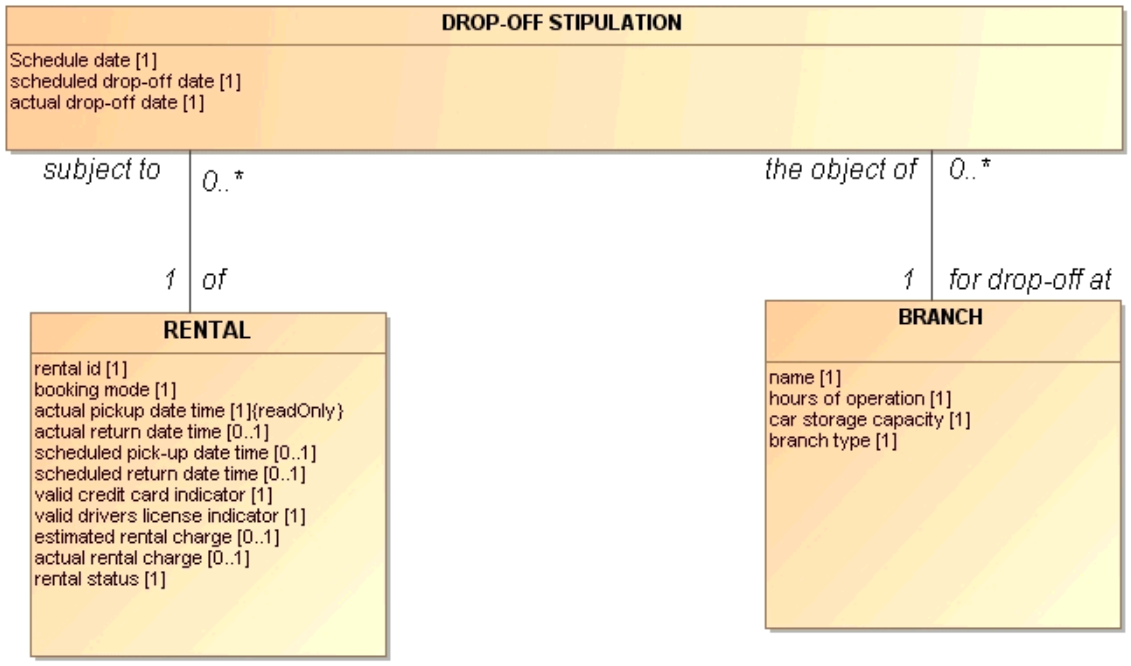

Figure 6: Drop-off Assignment

That is, each DROP-OFF STIPULATION is the fact that a particular **rental** stipulates that its car is to be dropped off at a particular BRANCH, *at a particular day and time*. In other words, each DROP-OFF STIPULATION must be *of* one **rental** and *for drop-off at* one

BRANCH. Attribute values for an instance of DROP OFF STIPULATION describe both when the stipulation was made ("schedule date"), and when the car must in fact be returned ("scheduled drop-off date"). The latter was initially an attribute of **rental**, but moving it here allows maintenance of a history of changes to the scheduled date. Is this more than is required by the business? Could be. It is important to be able to go back to the people involved to find out.

That is, during the data modeling process, it will be necessary to maintain contact with the business community to ask questions and feed back the results.

2.4 CONVERT NOUNS TO ATTRIBUTES (AND RELATIONSHIPS)

Once the entity classes have been defined, we can use the SBVR output to identify attributes (characteristics) for each. For example, in the car rental company:

rental *has* pick-up branch

> **Necessity**: A rental *has* a pick-up branch if and only if the pick-up branch *is* the sending branch *of* the car movement that *is included in* the rental.

rental *has* return branch

> **Necessity**: A rental *has* a return branch if and only if the return branch *is* the receiving branch *of* the car movement that *is included in* the rental

rental *has* actual pick-up date/time

> **Necessity**: A rental *has* an actual pick-up date/time if and only if the actual pick-up date/time *is* the start date/time *of* the rental period that *is included in* the rental.

rental *has* actual return date/time

> **Necessity**: A rental *has* an actual return date/time if and only if the actual return date/time *is* the end date/time *of* the rental period that *is included in* the rental.

rental *has* rental duration

> **Necessity**: A rental *has* a rental duration if and only if the rental duration *is* the duration *of* the period that *is* the rental period that *is included in* the rental.

A rental "pick-up branch", "return branch", "pickup date/time", "actual return date/time", and "rental duration" could all be specified in the entity/relationship model as attributes of **rental**.

As discussed above, however, we have already recognized that "(stipulated) return branch" is not an attribute, but a role played by a BRANCH for a **rental**. Similarly, in this case "pick-up branch" and (actual) "return branch" are not attributes, but roles played by BRANCH in a **car movement**. This is shown in the entity/relationship model in Figure 7.

Specifically, “Car movements happen in two ways:

A direct transfer. This is an action internal to EU-Rent: a EU-Rent employee drives the car.

A one-way rental between different local areas in the same country: a rental customer drives the car. [Note: if the one-way rental is between different countries, car ownership is not transferred. At a time decided by the branch manager, the car is returned to the country in which it is registered.]” [OMG 2008-b, p. 268]

The entity class **car movement**, then, as shown in Figure 7, has two sub-types, **rental movement** and **transfer**. Instances of each of these must be *from* one BRANCH and *to* another BRANCH. Each **rental** MOVMENT¸ however, must be *included in* one **rental**.

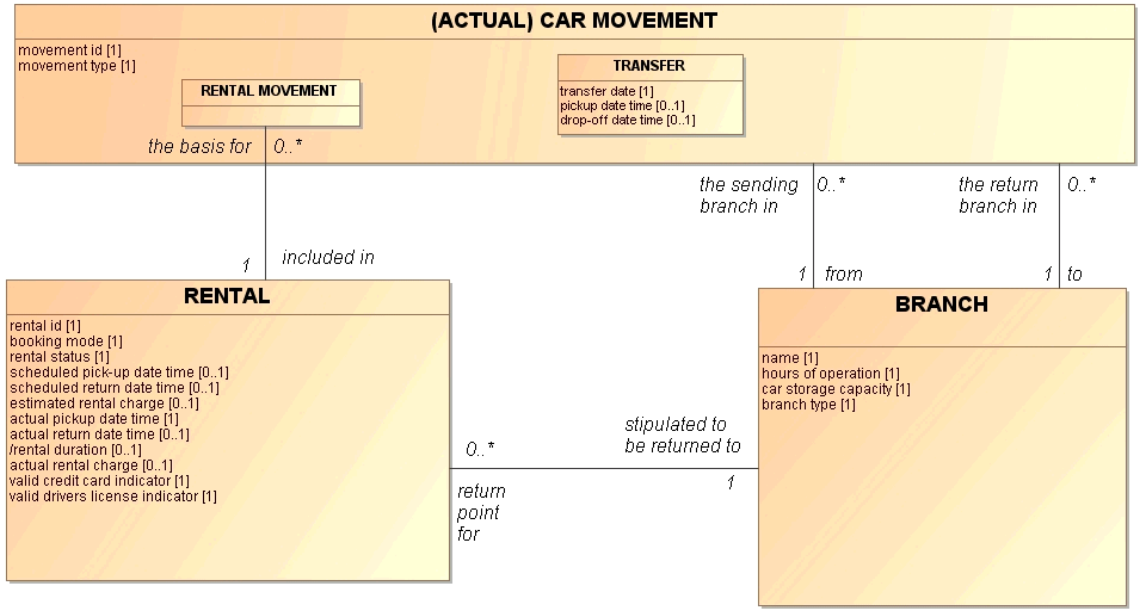

Figure 7: Sending and Return Branches

“Actual pick-up date/time” and “Actual return date/time” could be legitimate attributes of **rental**, since they represent characteristics of **rental** that do not have attributes themselves. The description, however, asserts that they are attributes of something called a **rental** PERIOD. This would suggest that an entity class could be defined for **rental** PERIOD along with the role:

Each **rental** must be *during* one and only one **rental** PERIOD.

But it is not appropriate to implement **rental** PERIOD as an entity class. The business information contained in that is a reproduction of the attributes already appropriately in **rental**. And there is no permanent thing of significance independent of those **rental attributes**. It is not meaningful, for example, to assert that:

Each **rental** PERIOD may be *referenced in* one or more **rental**S.

But that makes no sense. It is highly unlikely that two rentals will be for exactly the same time period. For this reason, **rental** PERIOD is not an entity class in our example.

Rental duration, while it is indeed an attribute of **rental**, is a special case: It is a ***derived attribute***, whose value is calculated by subtracting the actual pick-up date/time from the actual return date/time. This is shown on the model by the attribute name's being preceded by "/".

2.5 IDENTIFY OMMISSIONS

Figure 8 shows a fragment of a model developed by the Business Rules Team for the SBVR specification [OMG 2008-b, p. 293]

It is very similar to the architect's entity/relationship model that we are trying to derive from the SBVR specification. It has the specific names presented by business people: "rental" "request for pick-up", etc.

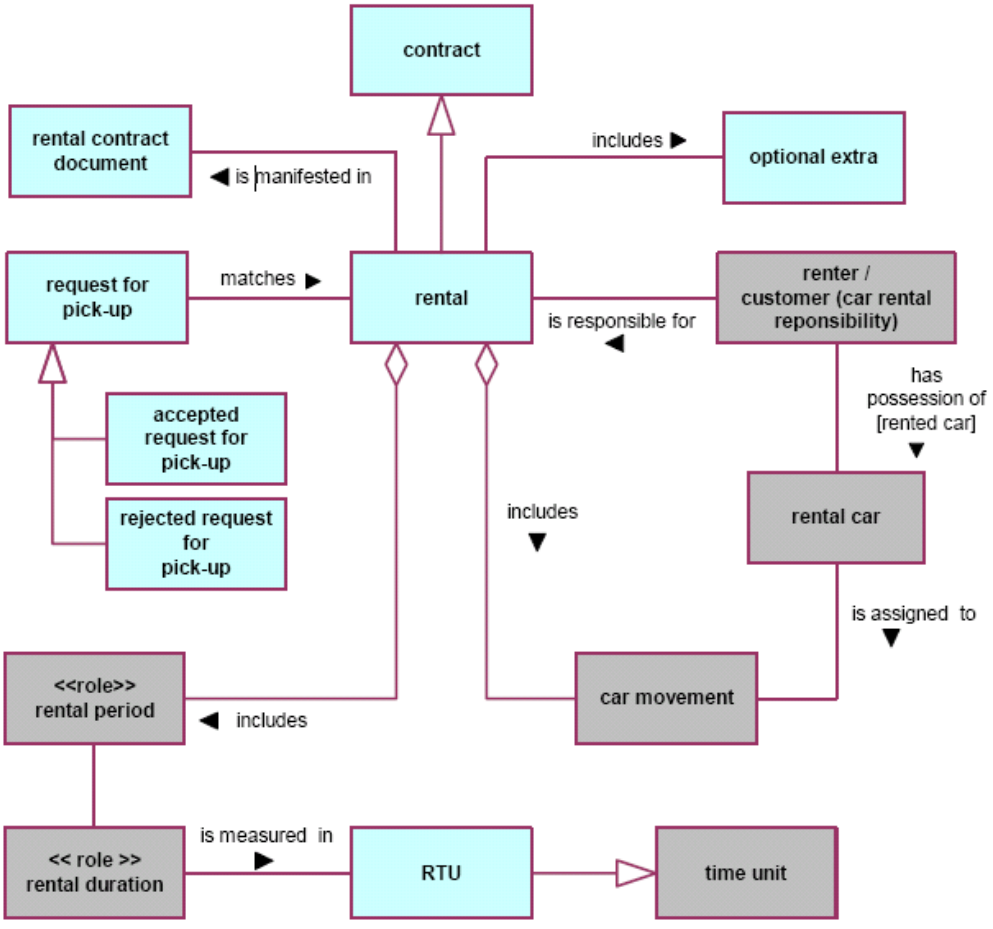

Figure 8: Part of the EU-Rent SBVR model [OMG 2008-b, p. 293]

Figure 9 shows the standard entity/relationship model for a contract. Clearly this is not as interesting to the business community as is Figure 8. But it does generate some questions. Noting that a **rental** is in fact a kind of contract, who are the "buyer" and the "seller"? The buyer (if the CONTRACT is a **rental**) is clearly the RENTER/CUSTOMER. But, following our discussion above about roles inside nouns, is not "renter" really the relationship name between **person** and **rental**? Indeed, even in the class box in the VR specification model is labeled "car rental responsibility", which suggests that it is a **person** who might be *responsible for* the **rental**?

Assuming for a moment, however, that RENTER / CUSTOMER is sufficient to describe the buying side of the contract, there is no information about the seller. In other pages of this model, there is the assertion that a **rental** uses a [drop-off] BRANCH and separately that a **rental** CAR is *stored at* a BRANCH, but nowhere is there the assertion that a **rental** must be *issued by* a BRANCH. Of course this is probably one of those things that "goes without saying". But what if subsequently, the home office issues a special rental for the President of France?

Yes, this may be highly unlikely. But it is important to have a complete description of the contract. Our industry is famous for being burned by specifications that "go without saying", so no one does—and someone should have.

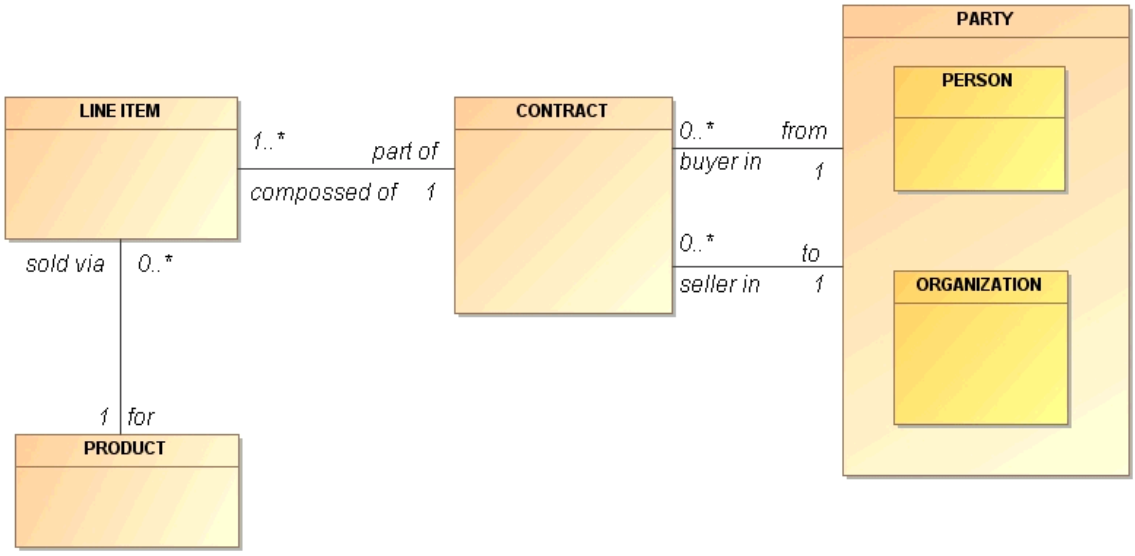

Figure 9: Standard Contract Model

3. THE COMPLETE SAMPLE MODEL

Figure 10 shows the model fragments discussed above all brought together. This drawing, then, will serve as the example to be carried through the various models present in this document. For this reason, two elements have been added to it:

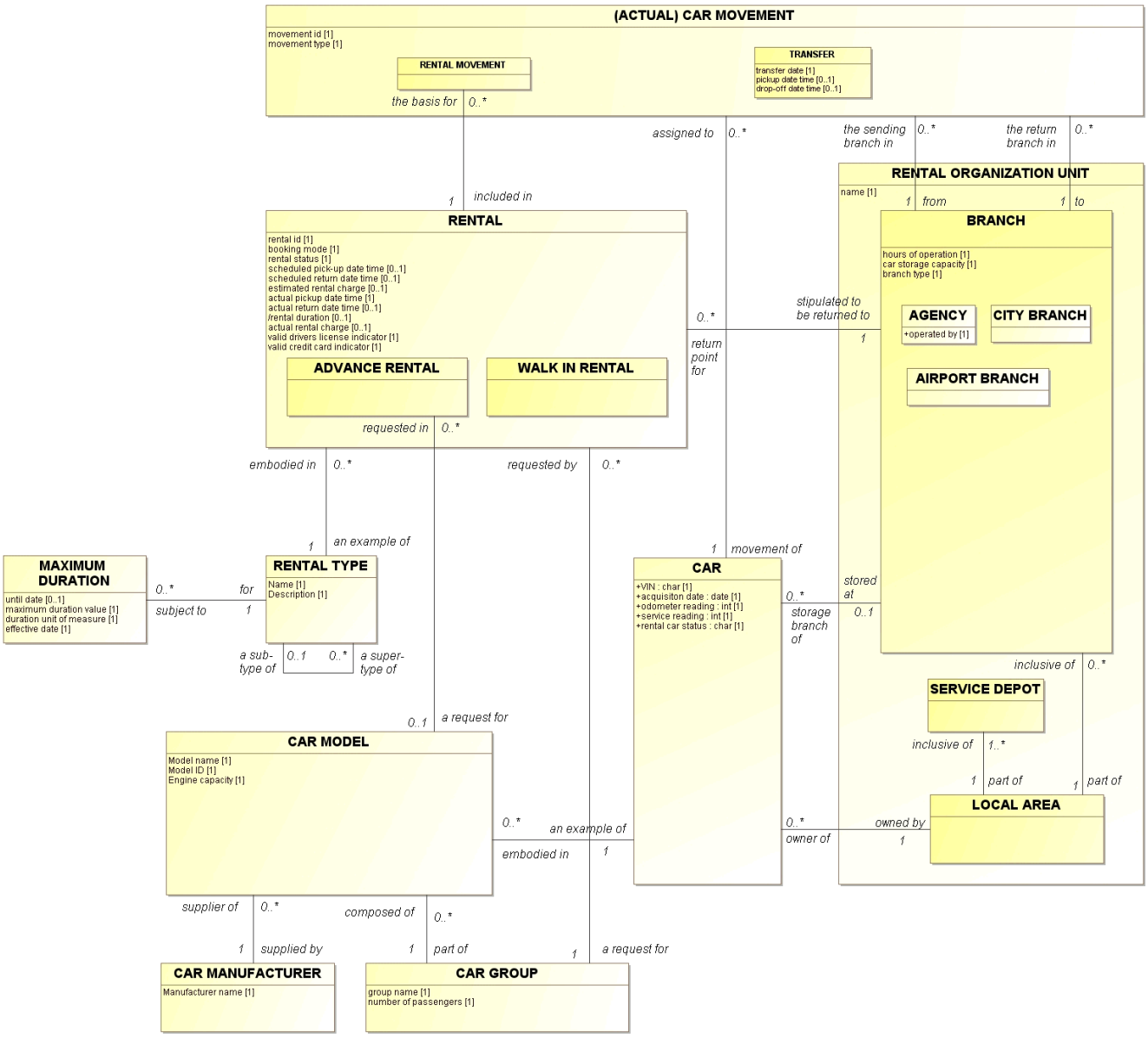

Figure 10: The Complete Sample Model

First, it adds CAR, a physical vehicle available for rent. In the SBVR specification, a "Rental" is defined as:

A: contract with a renter specifying use of some car of a car group for a rental period and a car movement [OMG 2008-b, p. 294]

This suggests that there should be a direct relationship between **rental** and CAR. To wit:

Each **rental** must be *for the use of* exactly one CAR.

The SBVR specification goes on to assert, however, that

A rented car *is assigned to* a rental if and only if the rented car *is* the rental car that *is assigned to* the car movement that *is included in* the rental. [OMG 2008-b, p. 296]

This means that the direct relationship between **rental** and CAR would be redundant, since it is achieved via the navigation through **car movement**. It also allows for the case where there may be multiple **car movement**s for one **rental**, with multiple CARS involved. For example a rented car might break down during the rental and be replaced. (Note that "rental car" is another example of the entity class CAR being folded in with the role, its relationship to **rental**.)

A second addition to this drawing from the SBVR specification is the super-type **rental organization** UNIT, along with various sub-types. These will be used to illustrate the various approaches to converting super-type structures into relational databases.

Note that each CAR may be *stored at* no more than one BRANCH (at a time?) [OMG 2008-b, p. 316]

In addition, each CAR must be *owned by* exactly one LOCAL AREA [OMG 2008-b, p. 316]

("LOCAL AREA" is here presumed to be a legal entity class that can own cars.)

4. CONCLUSION

The SBVR specification captures the language of a business in its vocabulary of terms and its propositions that assert business rules. Entity/relationship modeling, properly done, can represent the vocabulary and the alethic propositions (those describing what is necessary or possible) in a graphic form. In so doing, it adds a level of discipline necessary to provide a basis for developing computerized databases and application systems. This section attempted to describe a method for doing so.

Deontic business rules—those concerned with obligation and company policy—cannot be presented in this graphic form. They must still be captured, however, and passed along to those responsible for developing systems.