ORM 2

Terry Halpin

Neumont University
Salt Lake City, Utah, USA.
e-mail: terry.halpin@neumont.edu

Abstract: Object-role Modeling (ORM) is a fact-oriented modeling approach for specifying, transforming, and querying information at a conceptual level. Unlike Entity-Relationship modeling and Unified Modeling Language class diagrams, fact-oriented modeling is attribute-free, treating all elementary facts as relationships. For information modeling, fact-oriented graphical notations are typically far more expressive than other notations. Introduced 30 years ago, ORM has evolved into closely related dialects, and is supported by industrial and academic tools. Industrial experience has identified ways to improve current ORM languages (graphical and textual) and associated tools. A project is now under way to provide tool support for a second generation ORM (called ORM 2), that has significant advances over current ORM technology. This paper provides an overview of, and motivation for, the enhancements introduced by ORM 2, and discusses an open-source ORM 2 tool under development.

1 Introduction

Object-Role Modeling (ORM) is a fact-oriented approach for modeling, transforming, and querying business domain information in terms of the underlying facts of interest, where all facts and rules may be verbalized in language readily understandable by non-technical users of those business domains. Unlike Entity-Relationship (ER) modeling [6] and Unified Modeling Language (UML) class diagrams [31, 32, 33], ORM treats all facts as relationships (unary, binary, ternary etc.). How facts are grouped into structures (e.g. attribute-based entity types, classes, relation schemes, XML schemas) is considered an implementation issue irrelevant to capturing business semantics. Avoiding attributes in the base model enhances semantic stability and populatability, and facilitates natural verbalization. For information modeling, fact-oriented graphical notations are typically far more expressive than other graphical notations. Fact-oriented textual languages are based on formal subsets of native languages, so are easier to understand by business people than technical languages like the Object Constraint Language (OCL) [35]. Fact-oriented modeling includes procedures for mapping to attribute-based structures, such as those of ER or UML. For a basic introduction to ORM, see [14], and for a thorough treatment see [15].

Though less well known than ER and object-oriented approaches, fact-oriented modeling has been used successfully in industry for over 30 years, and is taught in universities around the world. The fact-oriented approach comprises a family of closely related "dialects", some using the generic term "Object-Role Modeling" (ORM) [15], and some using different names such as Natural language Information

Analysis Method (NIAM) [12, 36], and Fully-Communication Oriented Information Modeling (FCO-IM) [1, 2]. ORM languages include RIDL [30], LISA-D [26, 27] and FORML [15]. Though using a different notation, the Object-oriented Systems Model (OSM) [11] is a close relative to ORM, with its attribute-free approach.

Commercial tools supporting the fact-oriented approach include the ORM solution within Microsoft's Visio for Enterprise Architects [23], and the FCO-IM tool Case-Talk [5]. Free ORM tools include VisioModeler [34] and Infagon [28], as well as various academic prototypes. Dogma Modeler [10], an ORM-based tool for specifying ontologies, is currently being significantly extended.

Industrial ORM experience has identified ways to improve current ORM languages (graphical and textual) and tools. Our project aims to specify and provide tool support for a second generation ORM (called *ORM 2*), that has significant advances over current ORM technology in both functionality and usability. This paper overviews and motivates several enhancements introduced by ORM 2, and discusses our tool under development to support it. The initial development team comprised of faculty and students at Neumont University is being expanded to include external collaborators from industry and academia. The current implementation is coded in C# as a free, open-source plug-in to Microsoft Visual Studio .NET, using the new Microsoft Designer Framework Software Development Kit for building domain specific languages.

The rest of this paper is structured as follows. Section 2 focuses on improvements made to the ORM graphical notation. Section 3 discusses enhancements to the ORM textual notation. Section 4 discusses tooling aspects. Section 5 summarizes the main results, suggests topics for further research, and lists references. An online appendix includes sample schemas in the new notation, and a screen shot from the new tool.

2 The ORM 2 Graphical Notation

This section includes sample diagrams to contrast the new notation with the current notation used by the ORM source model solution in Microsoft Visio for Enterprise Architects [23]. The main objectives for the ORM 2 graphical notation are:

- More compact display of ORM models without compromising clarity
- Improved internationalization (e.g. avoid English language symbols)
- Notation changes acceptable to a short-list of key ORM users
- Simplified drawing rules to facilitate creation of a graphical editor
- Full support of textual annotations (e.g. footnoting of textual rules)
- Extended use of views for selectively displaying/suppressing detail
- Support for new features (e.g. role path delineation, closure aspects, modalities)

Although far more expressive graphically than UML or industrial ER for static data models, ORM schema diagrams typically consume more space because of their attribute-free nature (which also leads to greater semantic stability). The larger diagram size problem may be ameliorated by providing attribute-views on demand, and/or by redesigning the ORM graphic notation to be more compact. The first solution includes displaying "minor fact types" as attributes on an ORM diagram, and automatically generating attribute-based schemas for implementation (e.g. relational schemas, OO class schemas, XML schemas). As we have not yet completed our implementation of attribute view toggles, we instead focus on the new ORM graphical notation we have implemented. This notation is more compact, typically yielding diagrams about 65% the size of an equivalent diagram in the old notation. All English-specific symbols in the old notation have been replaced by language-neutral symbols to improve localization in different language communities. A survey was issued to eighteen ORM experts. Each change introduced by the new notation was acceptable to a majority.

Most of the ORM 2 figures in this paper were drawn using an ORM2 Visio stencil that our team created. Currently, the stencil may be used for drawing purposes only. In contrast, the ORM 2 modeling tool is intended to support automatic transformation between graphical and textual representations, as well as transformation to/from other schemas (e.g. relational, class, and XML schemas), and code generation (e.g. to DDL or program code). The tool is also intended to front-end other modeling tools (e.g. one might enter and transform an ORM schema to a relational schema for export to another database design tool to generate the DDL code). As schemas developed in the tool are fully exposed as XML, there is significant scope for inter-operability.

2.1 Object Type Shapes

In the old ORM notation, object type (entity type and value type) shapes are depicted by named ellipses. Ellipses are faster for users to draw manually, but they consume more space than rectangles (hard or soft), especially when names are long. For ORM 2, the default shape for object types is a *soft rectangle* (rectangle with rounded corners). Besides providing a more compact container for the enclosed text, this is consistent with the current notation for nested object types. The shape auto-sizes to provide appropriate white space around the text. Users may spread text over multiple lines (as in the Visio ORM source solution). Text is displayed in a user-definable default style, individual text elements may be user-selected for alternate styles, and text may be left/center/right justified.

To make this notation change more acceptable, we allow an ellipse or a hard rectangle as an alternative shape for object types, as set by a configuration option. Fig. 1 shows some examples. If the ellipse option is chosen, the shape is still more compact than the old notation. Object type shape examples in the rest of this document use the default shape (soft rectangle). Of the 18 experts in the survey, 12 preferred the soft rectangle, 5 preferred the ellipse shape, and one preferred the hard rectangle.

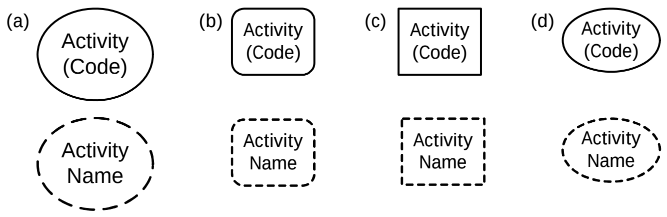

Fig. 1. Object type shapes in (a) old and new notations: (b) default: (c) and (d) alternatives

2.2 Shapes and Readings for Predicates and Roles

To save space, the *size of the role boxes is significantly reduced* (see Fig. 2). A line connecting a role box to an object type shape goes from the mid-point of an outer edge of the role box to the *center* of the object type shape (unlike the old solution, which uses connection points). *Predicate readings* may be user-positioned beside the predicate shape. Although it is no longer possible to place a predicate reading inside a role box, as in Fig. 2(a), the role boxes may now be used to include role sequence details (to disambiguate role paths). By default, all text is in 7 point Tahoma (Visio uses 8 point Arial). Of the 18 experts surveyed, 14 approved the changes.

The reduced role box size allows multiple single-role set-comparison constraints, (Fig. 2(d)). Unlike the old notation in Fig. 2(c), to add an extra set-comparison constraint between the role-pairs the new notation requires moving a constraint to make room for the third constraint, but since such cases are rare this minor inconvenience is acceptable. The size of the constraint bubbles is slightly reduced in the new notation.

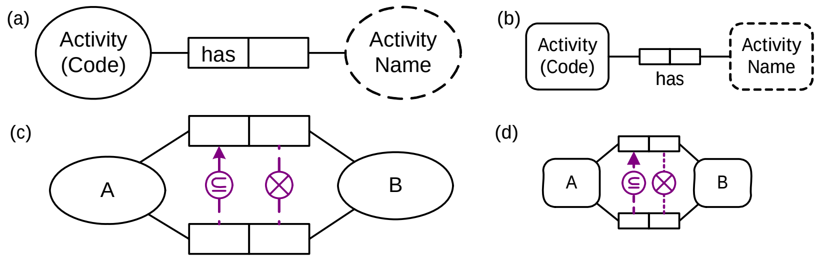

Fig. 2. The role box and text size in ORM (left side) is reduced in ORM 2 (right side)

A forward predicate reading is read left-to-right or top-to-bottom, and an inverse predicate reading (pre-pended by "« ", or possibly an arrow-head "◀") is read right-to-left or bottom-to-top. Two binary predicate readings may be displayed together, separated by a slash, or separately on either side of the predicate shape with the inverse reading pre-pended by "« ". The display of any predicate reading may be toggled on/off. Multi-line reading displays are allowed. Fig. 3(b) shows some possibilities.

For a fact type with *n* roles (*n* > 0), ORM 2 allows predicate readings for all possible (*n*!) permutations of role orderings. For each such role ordering, one or more *alias readings* may be supplied (e.g. "is employed by" as an alias of "works for"). Query navigation in relational style from any role of an *n*-ary fact type is enabled by just *n* predicate readings (one starting at each role), but industrial modelers requested this additional flexibility. For non-binary fact types, at most one predicate reading is displayed on the diagram. ORM 2 allows a name (as distinct from a reading) for the fact type (e.g. "Birth" for Person was born on Date), though this is not normally displayed on the diagram. One use of *fact type names* is to generate a suitable target name for fact types that map to a single table or class. Multi-line fact type names are allowed.

The *display of role names* in square brackets (Fig. 3(c)) may be user-positioned and toggled on/off. Multi-line role names are allowed. The display toggle may be set globally or on an individual role basis. Although each fact type has at least one predicate reading, the display of predicate readings may be suppressed (e.g. to focus on role names). By default, role names are displayed in a different color (e.g. indigo).

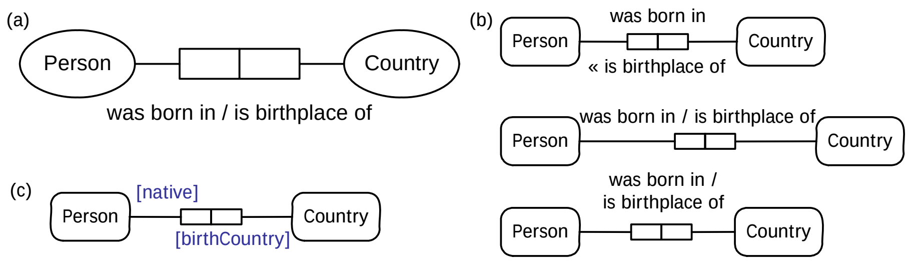

Fig. 3. Predicate and role readings display in (a) ORM and (b), (c) ORM 2

Apart from size reduction, *objectification* in ORM 2 allows *nesting of unary predicates*, as well as *predicates with non-spanning uniqueness constraints*, in accordance with new formal semantics and guidelines for objectification for ORM 2 [16, 22].

2.3 Uniqueness Constraints

Internal uniqueness constraints apply to single predicates, and appear in ORM as arrow-tipped lines. ORM 2 instead uses *simple lines*, which are faster to draw manually, and intuitively correspond to the common practice of underlining keys (Fig. 4(b)). The line is shorter than the role box length to avoid ambiguity. The old ORM notation marks a primary uniqueness constraint as “P”. In ORM 2, a *preferred uniqueness constraint* is indicated by a *double line* (as in the practice of doubly underlining primary keys when alternate keys exist). This also avoids the English bias of “P”. In ORM 2 the notion of preferred uniqueness is conceptual (a business decision to prefer a particular identification scheme). By default, all ORM 2 constraints are colored violet.

In the case of an internal uniqueness constraint that spans non-contiguous roles, a *dashed line* bridges the gap across the inner role(s) that are excluded from the constraint. Such a case may arise only if the association is ternary or higher. For example, the upper uniqueness constraint on the ternary in Fig. 4 spans the first and last roles. Of the 18 experts surveyed, 17 preferred the new internal constraint notation.

An *external uniqueness constraint* spans roles from different predicates. The old ORM notation depicts this by a circled “U” (for unique), or “P” (for “primary”) if used for primary reference (Fig. 4(a)). This notation is biased towards English, and differs totally from the internal uniqueness notation. For localization and consistency, the new notation (Fig. 4(b)) uses a *circled underline*, or a *circled double underline* if the constraint provides the preferred identification scheme (consistent with the new internal uniqueness constraint notation and the horizontal notation for relational schemas [15]). Of the 18 experts surveyed, 14 preferred this new constraint notation.

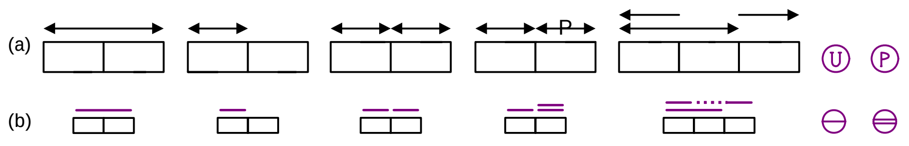

Fig. 4. Uniqueness constraints in (a) ORM and (b) ORM 2

2.4 Mandatory Role Constraints

In the old ORM notation, *simple mandatory role constraints* are indicated by a solid dot either (a) at the intersection of an entity type shape and the line connecting it to a role, or (b) at the role end. Option (b) avoids ambiguity when an object type plays many mandatory roles whose connections to the object type are too close to distinguish the role to which the dot applies. *Disjunctive mandatory (inclusive-or) constraints* are depicted by placing the solid dot in a circle connected by dotted lines to the roles it applies to. ORM 2 retains this notation, except that the solid dot is consistently colored *violet* and a global *configuration option* determines the default placement of simple mandatory dots at the role or object type end. Users may override this global setting on an individual role basis. Fig. 5 shows some simple examples.

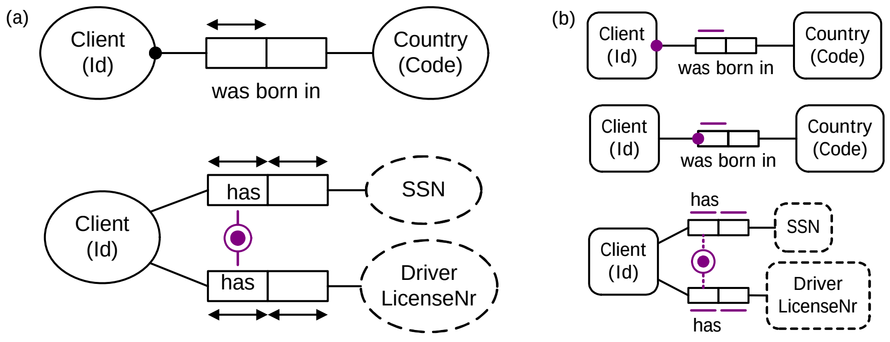

Fig. 5. Mandatory constraints in (a) ORM and (b) ORM 2

2.5 Set-comparison, Exclusive-Or, Frequency and Value Constraints

Subset, exclusion, and equality constraints continue to be denoted by a circled ⊆, ×, = respectively, connected to the relevant roles by dashed lines, except that the ORM 2 shapes are a bit smaller, with refined symbols. In addition, ORM 2 supports the *n-ary version of the equality constraint*. As usual, *exclusive-or* constraints are denoted by combining the circled × with the circled dot, and users may display the two component constraints overlaid or separately (as in Visio). Fig. 6(b) shows the basic shapes.

Frequency constraints are displayed as in Visio, except that single symbols (≤, ≥) replace double symbols (<=, >=), for example 2, ≥3, 2..5. *Value constraints* are denoted as in Visio, except that many values may be displayed on a single line (e.g. {'M', 'F'}, {'a',..'z'}), and *open ranges* are supported (e.g. "> 0" for PositiveReal).

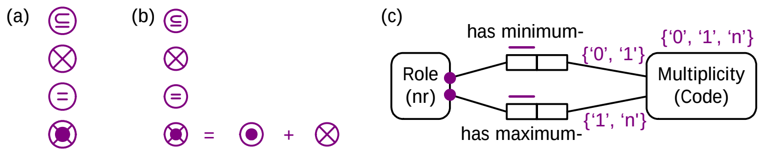

Fig. 6. Set-comparison, Xor and value constraints

ORM 2 allows *value constraints on roles*, not just object types. For example, the Information Engineering metamodel fragment in Fig. 6(c) includes value constraints on the minMultiplicity and maxMultiplicity roles. Of the 18 experts surveyed, all favored support for open ranges, and 17 favored role-based value constraints.

2.6 Ring Constraints and Subtyping

The current Visio ORM tool uses English abbreviations for various *ring constraints*: ir = irreflexive, as = symmetric, ans = antisymmetric, it = intransitive, ac = acyclic, sym =symmetric. Ring constraints are displayed as a list of one or more of these options, appended to a ring symbol "O", and connected to the two relevant roles (if placed very close, the connection display is suppressed). For example, the reporting relationship is declared to be acyclic and intransitive as shown in Fig. 7(a).

This old notation has disadvantages: the abbreviations are English-specific (e.g. Japanese readers might not find "ir" to be an intuitive choice for irreflexivity); and the ring constraint display tends to cross over other lines (as in the example). To remove the English bias, and suggest the constraint semantics, ORM 2 uses *intuitive icons*. To reduce edge crossings, ORM 2 *omits role links* if the predicate has just two roles played by the same object type (or compatible object types). For example, in ORM 2 the reporting relationship is declared acyclic and intransitive as shown in Fig. 7(b).

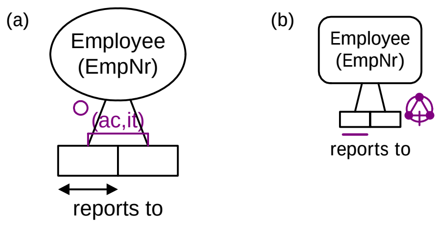

Fig. 7. Acyclic and Intransitive ring constraints depicted in (a) ORM and (b) ORM 2

The ORM 2 icons for ring constraints are loosely based on our icons for teaching ring constraints [15, sec. 7.3], where small circles depict objects, arrows depict relationships, and a bar indicates forbidden (Fig. 8). The different node fills in the antisymmetric icon indicate that the inverse relationship is forbidden only if the objects differ (in the other icons, the objects may be the same). For diagramming purposes, these teaching icons take up too make room, especially when combinations of ring constraints apply.

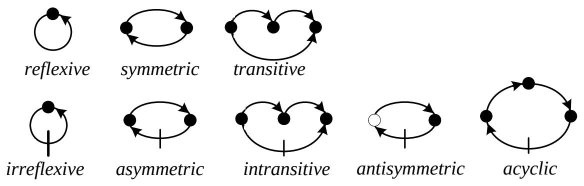

Fig. 8. The original icons used for teaching ring constraints

So simplifying adaptations were made to ensure the final icons are distinguishable while maintaining a compact footprint. The ORM 2 icons (Fig. 9) print clearly at 600 dpi, and are readable on screens at typical resolutions used for industrial modeling. They may be distinguished on low resolution screens by increasing the zoom level. ORM 2 has an icon for each of the ten simple or combined ring constraint choices supported by the current ORM source model solution. In contrast to the teaching icons, arrow-heads are removed (they are assumed), and relevant pairs of constraints are collapsed to a single icon. While the simplified icons are less intuitive than the teaching icons, once their origin is explained it should be easy to remember their meaning. Of the 18 experts surveyed, 14 agreed to the new ring constraint icons.

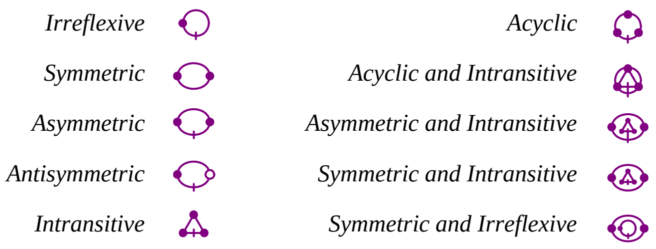
Irreflexive
Symmetric
Asymmetric
Antisymmetric
Intransitive
Acyclic
Acyclic and Intransitive
Asymmetric and Intransitive
Symmetric and Intransitive
Symmetric and Irreflexive

Fig. 9. ORM 2 icons for ring constraints

The current arrow notation for *subtyping* will remain, perhaps supplemented by Euler diagrams as an alternative display option for simple cases. ORM 2 adds support for explicit display of *subtype exclusion* (⊗) *and exhaustion* (⦿) *constraints*, overlaying them when combined, as shown in Fig. 10. As such constraints are typically implied by other constraints in conjunction with subtype definitions, their display may be toggled on/off. Of the 18 experts surveyed, 17 approved this extension.

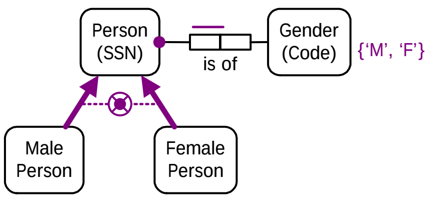

Fig. 10. Explicit display of exclusion and exhaustion constraints for a subtyping scheme

3 The ORM 2 Textual Notation

ORM 2 will support a high level, formal, *textual language* for the declaration of ORM schemas (including fact types, constraints and derivation rules), ORM queries, and possibly fact addition and deletion. The tool will generate code to implement the semantics conveyed by the textual language. The textual language will be cover all the semantics conveyed by the graphical language, as well as additional semantics (e.g. constraints that cannot be captured graphically). All graphical constraints will have automated *constraint verbalizations*, using improvements discussed elsewhere [19]. Unlike the Visio ORM solution, the ORM 2 textual language may be used to input models, instead of being merely an output language for verbalizing diagrams.

Textual constraints may be noted on the diagram by *footnote numbers*, with the textual reading of the constraints provided in *footnotes* that can be both printed and accessed interactively by clicking the footnote number. Fig. 11 provides an example. Of the 18 experts surveyed, 17 approved the use of footnotes for textual constraints.

Derivation rules may be specified for derived object types (subtype definitions) and derived fact types. ORM 2 allows *fully-derived subtypes* (full subtype definition provided), *partly-derived subtypes* (partial subtype definition provided) and *asserted subtypes* (no subtype definition provided). *Subtype definition*s will be supported as derivation rules in a high level formal language rather than mere comments, and may be displayed in text boxes as footnotes on the diagram. Iff-rules are used for full derivation, and if-rules for partial derivation. Here are sample rules in both ORM 2's textual language and predicate logic for fully and partly derived subtypes respectively:

> Each Australian is a Person who was born in Country 'AU'.
> $\forall x$ [Australian $x \equiv \exists y$:Country $\exists z$:CountryCode (x was born in y & y has z & z = 'AU')]
>
> Person_1 is a Grandparent if Person_1 is a parent of some Person_2 who is a parent of some Person_3.
> $\forall x$:Person [Grandparent x $\subset \exists y$:Person $\exists z$:Person (x is a parent of y & y is a parent of z)]

The final grammar for the textual language is not yet determined, but should support declaration of ORM models and queries in *relational style*, *attribute style* and *mixed style*. Relational style uses predicate readings (e.g. the subtype definitions above), while attribute style uses role names. Attribute style is especially useful for derivation rules and textual constraints of a mathematical nature (e.g. see Fig. 11).

As an example of a derivation rule for a derived fact type, we may define the uncle association in relational style thus: Person_1 is an uncle of Person_2 iff Person_1 is a brother of a Person_3 who is a parent of Person_2. Assuming the role names "brother" and "parent" are declared, we may also specify the rule in attribute style thus: For each Person: uncle = brother of parent. Further examples may be found elsewhere [19, 21].

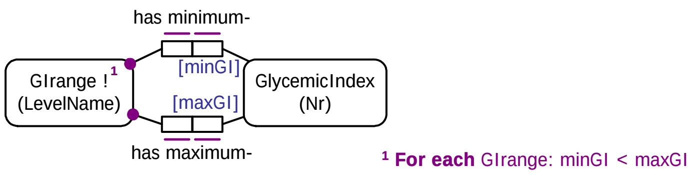

Fig. 11. Textual constraints appear as footnotes in ORM 2

4 Tooling Aspects

This section briefly considers further aspects of the ORM 2 tool. A screen shot from the tool illustrating use of Intellisense in the fact editor is accessible online at www.orm.net/orm2/dietschema.doc. There is no space here to detail the tool's features, but the tool should significantly extend the functionality of Microsoft's current

ORM tool, both by supporting ORM 2 and by adding features. For example, reference modes are treated as views of reference fact types, and the tool allows automatic toggling of the display between the implicit (reference mode) and explicit (reference fact type) representations. This toggle capability is being extended to support attribute-based displays as views of the underlying fact-based schemas. Full integration with Visual Studio provides a powerful and familiar environment for developers, and the architecture and open source nature facilitates adaptation and extensions by others.

The ability to hide/show types of constraints by placing them on different layers should be at least as versatile at that in the Visio solution, which offers 5 layers. The ORM 2 tool will probably extend this to allow suppressing display of all constraints (including internal uniqueness and mandatory constraints).

One abstraction mechanism provided by the Visio ORM solution is the ability to spread a model over many pages, where each page focuses on a sub-area of the business domain. Model elements may be redisplayed multiple times on different pages, as well as on the same page (to reduce edge-crossings). The Show-Relationships feature is indispensable for revealing connections (and for other reasons). At a minimum, this functionality should be supported. Ideally, users should be able to decompose models into multiple levels of refinement, and abstract upwards, with the tool automating the process via the major object type algorithm [3, 15] or a similar technique.

5 Conclusion

This paper discussed a project under way to specify and provide open source tool support for a second generation ORM (called ORM 2), that provides significant advances over current ORM technology. Proposed changes to the graphical notation were described, and their motivation explained. Results from a survey of ORM experts with regard to these changes were noted. Various enhancements to the ORM textual notation were examined, and some improvements from a tooling perspective were identified. To better appreciate the difference made by the new notation, a three page Diet model in both the Visio ORM source model notation and the ORM 2 notation is available as an online appendix at http://www.orm.net/orm2/dietschema.doc.

Parties who own a copy of Visio (Standard edition or higher) and who wish to explore the new graphical notation using models of their own may download a zip file containing the Visio ORM 2 stencil and template plus a sample model file from the following site: www.orm.net/ORM2_Beta.zip. This ORM 2 stencil is for drawing only—it does not generate code. Parties who wish to collaborate in the actual development of the open source ORM 2 tool should e-mail the author of this paper.

The tools project team is currently researching extensions to ORM in several areas, including richer support for join constraints (e.g. distinguishing inner-outer join semantics, displaying complex join constraints, and role path disambiguation [20]), extended open/closed world semantics, and deontic/alethic constraint distinctions [19].

Acknowledgements

This paper benefited from discussion with the tools project team at Neumont University, and from the ORM 2 survey responses by Don Baisley, Dick Barden, Scott

Becker, Linda Bird, Anthony Bloesch, Necito dela Cruz, Dave Cuyler, Lance Delano, Jan Dietz, Ken Evans, Gordon Everest, Brian Farish, Pat Hallock, Bill MacLean, John Miller, Borje Olsson, Erik Proper and Pieter Verheyden.

References

1. Bakema, G., Zwart, J. & van der Lek, H. 1994, 'Fully Communication Oriented NIAM', *NIAM-ISDM 1994 Conf. Working Papers*, eds G. M. Nijssen & J. Sharp, Albuquerque, NM, pp. L1-35.
2. Bakema, G., Zwart, J. & van der Lek, H. 2000, *Fully Communication Oriented Information Modellin*g, Ten Hagen Stam, The Netherlands.
3. Bird, L., Goodchild, A. & Halpin, T.A. 2000, 'Object Role Modeling and XML Schema', *Conceptual Modeling – ER2000*, Proc. 19th Int. ER Conference, Salt Lake City, Springer LNCS 1920, pp. 309-322.
4. Bloesch, A. & Halpin, T. 1997, 'Conceptual queries using ConQuer-II', *Proc. ER'97: 16th Int. Conf. on conceptual modeling*, Springer LNCS, no. 1331, pp. 113-26.
5. CaseTalk website: http://www.casetalk.com/php/.
6. Chen, P. P. 1976, 'The entity-relationship model—towards a unified view of data'. *ACM Transactions on Database Systems*, 1(1), pp. 9–36.
7. Cuyler, D. & Halpin, T. 2005, 'Two Meta-Models for Object-Role Modeling', *Information Modeling Methods and Methodologies, eds* J. Krogstie, T. Halpin, & K. Siau, Idea Publishing Group, Hershey PA, USA (pp. 17-42).
8. Demey J., Jarrar M. & Meersman R. 2002, 'A Markup Language for ORM Business Rules', in Schroeder M. & Wagner G. (eds.), *Proc. International Workshop on Rule Markup Languages for Business Rules on the Semantic Web* (RuleML-ISWC02 workshop), pp. 107-128, online at http://www.starlab.vub.ac.be/publications/STARpublications.htm.
9. De Troyer, O. & Meersman, R. 1995, 'A Logic Framework for a Semantics of Object Oriented Data Modeling', *OOER'95, Proc. 14th International ER Conference*, Gold Coast, Australia, Springer LNCS 1021, pp. 238-249.
10. DogmaModeler: www.starlab.vub.ac.be/research/dogma/dogmamodeler/dm.htm.
11. Embley, D.W. 1998, *Object Database Management*, Addison-Wesley, Reading MA, USA.
12. Falkenberg, E. D. 1976, 'Concepts for modelling information', in Nijssen G. M. (ed) *Proc 1976 IFIP Working Conf on Modelling in Data Base Management Systems*, Freudenstadt, Germany, North-Holland Publishing, pp. 95-109.
13. Halpin, T. 1989, 'A Logical Analysis of Information Systems: static aspects of the data-oriented perspective', doctoral dissertation, University of Queensland.
14. Halpin, T. 1998, 'ORM/NIAM Object-Role Modeling', *Handbook on Information Systems Architectures*, eds P. Bernus, K. Mertins & G. Schmidt, Springer-Verlag, Berlin, pp. 81-101.
15. Halpin, T. 2001, *Information Modeling and Relational Databases*, Morgan Kaufmann, San Francisco.
16. Halpin, T. 2003, 'Uniqueness Constraints on Objectified Associations', *Journal of Conceptual Modeling*, Oct. 2003. URL: www.orm.net/pdf/JCM2003Oct.pdf.

17. Halpin, T. 2004, ‘Comparing Metamodels for ER, ORM and UML Data Models’, *Advanced Topics in Database Research, vol. 3*, ed. K. Siau, Idea Publishing Group, Hershey PA, USA, Ch. II (pp. 23-44).
18. Halpin, T. 2004, ‘Information Modeling and Higher-Order Types’, *Proc. CAiSE’04 Workshops*, vol. 1, (eds Grundspenkis, J. & Kirkova, M.), Riga Tech. University, pp. 233-48. Online at http://www.orm.net/pdf/EMMSAD2004.pdf.
19. Halpin, T. 2004, ‘Business Rule Verbalization’, *Information Systems Technology and its Applications*, Proc. ISTA-2004, (eds Doroshenko, A., Halpin, T. Liddle, S. & Mayr, H), Salt Lake City, Lec. Notes in Informatics, vol. P-48, pp. 39-52.
20. Halpin, T. 2005, ‘Constraints on Conceptual Join Paths’, *Information Modeling Methods and Methodologies, eds J.* Krogstie, T. Halpin, T.A. & K. Siau, Idea Publishing Group, Hershey PA, USA, pp. 258-77.
21. Halpin, T. 2005, ‘Verbalizing Business Rules: Part 11’, *Business Rules Journal*, Vol. 6, No. 6. URL: http://www.BRCommunity.com/a2005/b238.html.
22. Halpin, T. 2005, ‘Objectification’, *Proc. CAiSE’05 Workshops, vol. 1*, eds J. Calestro & E. Teniente, FEUP Porto (June), pp. 519-31.
23. Halpin, T., Evans, K, Hallock, P. & MacLean, W. 2003, *Database Modeling with Microsoft® Visio for Enterprise Architects*, Morgan Kaufmann, San Francisco.
24. Halpin, T. & Proper, H. 1995, ‘Database schema transformation and optimization’, *Proc. OOER’95: Object-Oriented and Entity-Relationship Modeling*, Springer LNCS, vol. 1021, pp. 191-203.
25. Halpin, T. & Wagner, G. 2003, 'Modeling Reactive Behavior in ORM'. *Conceptual Modeling – ER2003*, Proc. 22nd ER Conference, Chicago, October 2003, Springer LNCS.
26. ter Hofstede, A. H. M., Proper, H. A. & Weide, th. P. van der 1993, ‘Formal definition of a conceptual language for the description and manipulation of information models’, *Information Systems*, vol. 18, no. 7, pp. 489-523.
27. ter Hofstede A. H. M, Weide th. P. van der 1993, ‘Expressiveness in conceptual data modeling’, *Data and Knowledge Engineering*, 10(1), pp. 65-100.
28. Infagon website: http://www.mattic.com/Infagon.html.
29. Lyons, J. 1995, *Linguistic Semantics: An Introduction*, Cambridge University Press: Cambridge, UK.
30. Meersman, R. M. 1982, *The RIDL conceptual language*. Research report, Int. Centre for Information Analysis Services, Control Data Belgium, Brussels.
31. Object Management Group 2003, *UML 2.0 Infrastructure Specification*. Online: www.omg.org/uml.
32. Object Management Group 2003, *UML 2.0 Superstructure Specification*. Online: www.omg.org/uml.
33. Rumbaugh J., Jacobson, I. & Booch, G. 1999, *The Unified Language Reference Manual*, Addison-Wesley, Reading, MA.
34. VisioModeler download site: http://www.microsoft.com/downloads/ results.aspx?displaylang=en& freeText=VisioModeler.
35. Warmer, J. & Kleppe, A. 1999, *The Object Constraint Language: precise modeling with UML*, Addison-Wesley.
36. Wintraecken J. 1990, *The NIAM Information Analysis Method: Theory and Practice*, Kluwer, Deventer, The Netherlands.