Communications of the Association for Information Systems

Volume 9 Article 10

September 2002

The Details of Conceptual Modelling Notations are Important - A Comparison of Relationship Normative Language

Steve Hitchman
The University of Melbourne, steve@infomod.fsbusiness.co.uk

Communications of the Association for Information Systems

THE DETAILS OF CONCEPTUAL MODELLING NOTATIONS ARE IMPORTANT – A COMPARISON OF RELATIONSHIP NORMATIVE LANGUAGE

STEVE HITCHMAN
Department Of Information Systems
The University Of Melbourne
steve@infomod.fsbusiness.co.uk

ABSTRACT

Discussions of conceptual modelling assume that notation details are of secondary importance, a matter of taste and past experience rather than of science. For example, it does not really matter if cardinality is shown with a 'crow's foot' or with the symbol '1..*'. This paper argues that such an assumption is wrong and that the notation is extremely important in the process of modelling relationships because of the normative language that the notation specifies. Normative language is shown to be a useful way of understanding and comparing relationship notation. Barker's practical relationship definition, using a formal notation, is shown to be sufficiently well formed to allow the modeller to make sense of the domain in their own linguistic context. Less formal notations are shown to disadvantage the less experienced modeller

KEYWORDS: entity-relationship modelling, ERM, OMT, UML

I. INTRODUCTION

Nordbotten & Crosby [1999] showed that graphical notation affects the way that students read very simple entity-relationship and class diagrams. However, no empirical research examines the effects of particular notations in the practice situation. Neither is there any discussion in the literature about why particular notations were developed and the implications of using them. Little empirical research in the practitioner domain exists, but this research suggests that many practitioners use relationships in an informal way [Hitchman 1995, Hitchman 2000]. The definition of a relationship between entity-types needs to be concerned with 'making sense of the real world' [Checkland & Holwell, 1998] and with *how* to model.

Thalheim [1999, pps.56, 67] is an example of a rigorous formalised definition where a (first-order) relationship is an "association between single entity-types or clusters of entity-types"; "an element of the Cartesian product R^c_1 x ... x R^c_n x Dom(B_1) x ... x Dom(B_k)". Looking at examples of relationships in the same Thalheim text there is no guidance on how to use the definition – how to model relationships in practice. Data modelling is an applied science and defining how to use relationships in practice is much more problematic than agreeing to define a relationship as a Cartesian product.

The details of the diagram notation used for relationships during modelling received little or no attention and is assumed to be of secondary importance, a matter of taste and past experience rather than of science. For example, it does not really matter if cardinality is shown with a crow's foot or with the symbol '1..*'. This paper compares a formal approach to relationship notation with less formal approaches with the aim of showing that such an assumption is wrong - the notation is very important in the modelling process because of the normative language that the notation specifies. A secondary aim of the paper is to show that a practically useful definition of relationships already exists but is overlooked by researchers because the definition is linked to a particular notation and because it is embedded in a 'practitioner' text – guidance for modellers based on experience.

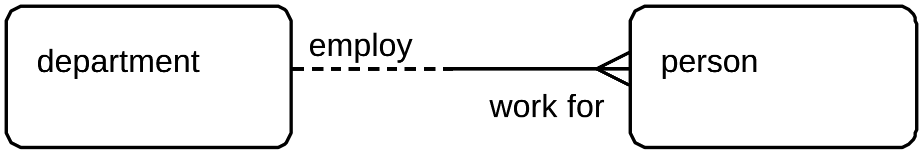
Section 2 describes the formal approach to relationship notation using the Barker definition. Section 3 discusses the importance of this notation through its effect on the normative modelling language. Section 4 compares the informal approaches to notation in several other standards to show that apparently disparate notations share a common informal approach. These alternative notations are shown to be both less formalised and less linguistically useful and therefore risk poor specification. Section 5 concludes by summarising the main findings.

II. THE FORMAL APPROACH TO RELATIONSHIP NOTATION

Many notations are proposed for data modelling. Several of these proposals adopt a 'formal' approach to relationship notation. In the context of practical modelling a notation is formal if it requires formality in conforming to an accepted custom – in the case of relationships the modeller is required to formally complete a specified set of notation dependent sentences for each relationship. This section describes the Barker [1989] relationship notation as the main proponent of the formal approach on the basis that:

- the standard was first published in 1989 in a text still in print (a major achievement for a computing text) and is still regarded by many as the best practitioner introduction to entity-relationship modelling; although separately specified in it's own text it is also part of an integrated analysis method;
- it is widely accessible;
- the text is referenced in undergraduate courses and in academic papers;
- it has been widely used since publication and is supported by a well known CASE tool;
- it was developed through, and is still supported by, a major software house with a worldwide presence (Oracle);
- it specifies business context relationships with an expectation of, but not a dependence on, implementation in a relational domain;
- the standard is prescriptive. It is mandatory to use the full formality (as users of the CASE tool will testify. However this does not mean it is used correctly, of course);
- it contains every element required to be both fully formal and elegantly simple. It represents the strongest level of formality in business specification.

Other notations, for example Bachman [Gane and Loosley 1990, Bachman 1969] could have been used because they have an equivalent formality, but can be considered to be less attractive, for example with the use of arrowheads in the case of Bachman. The Barker notation is shown in Figure 1, and is surprisingly simple. Indeed, one may wonder why later notations are so varied and lacking in formality given the low level of complexity involved.



A department may employ one or more people
A person must work for one and only one department

Figure 1. The Relationship Notation from Barker [1989]

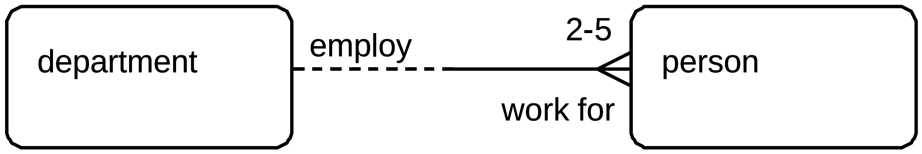
An interesting aspect of the Barker notation is that a normative language is also specified and is mandatory (see [Ortner & Schienmann, 1996] for an explanation of 'normative language' – basically the constrained language defined for the notation). The normative language takes the form of a combination of entity-type name, optionality, relationship name, cardinality, and entity-type name. The normative language structure for the previous diagram is shown in Table 1. The modeller can read the relationship in the order in which the symbols appear on the diagram. Conversely it is difficult to read the optionality and cardinality without including the relationship name.

Table 1. The Normative Relationship Language

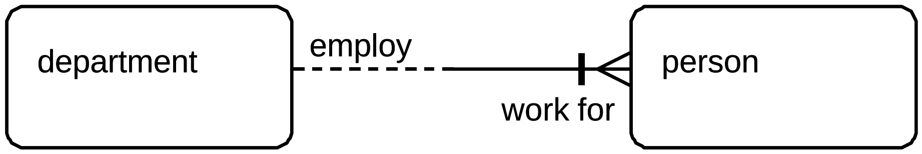
entity-type name	**optionality**	**relationship name**	**cardinality**	**entity-type name**
A department	may	employ	one or more	people
A person	must	work for	one and only one	department

Four additional constraints can be applied to relationships in the Barker standard. Barker specifies that a cardinality constraint, shown in 2a, can be applied to any many ended relationship, such as '2-5' (between 2 and 5), or <4 or =5. Barker also notes that such constraints are very rarely found in a business domain. These numeric constraints can be incorporated in the sentence pairs, replacing the 'one or more' phrase. This constraint is rarely used or supported by CASE tools and an example of the confusion about notations is found in Dorsey & Hudicka [1999, p.74-75], a generally well researched and well written book aimed at practitioners. The authors compare the Barker notation with UML (Unified Modelling Language) and incorrectly argue that the Barker standard does not support numerical constraints, whereas UML does. The authors incorrectly suggest that the Barker diagram requires twenty different relationships lines to support a '20' numeric constraint (Figure2a). The authors confuse the Barker standard with the lack of support for this feature in the Oracle CASE tool and then generalise the idea to all entity-relationship notations in order to support the contention that UML is a richer notation.

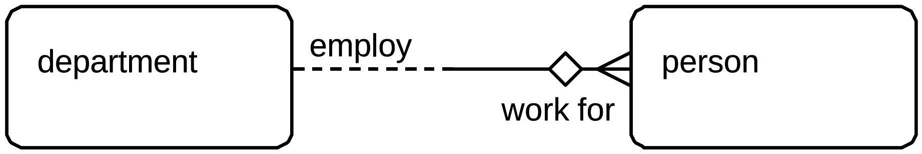
Figure 2b shows the widely used bar across the relationship line that means the unique identifier of the department is part of a concatenated unique identifier for the person. Although this is sometimes discussed as a 'dependency' the constraint is concerned with identities. The relationship already specifies that a person must work for a department and implies dependency - a person will not exist in this domain unless their department exists. On the other hand, a department has an existence independent of a person. It is easy to spot independent existence on a Barker diagram as an entity-type box will only be connected by dashed lines – the dashing giving the visual clue of optionality and therefore independence.



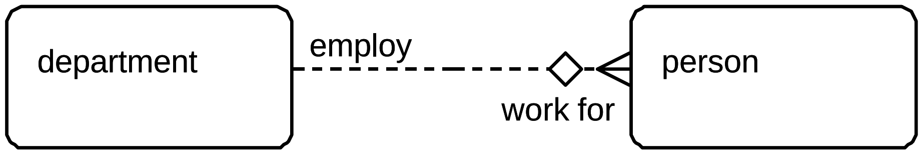
a. The cardinality constraint



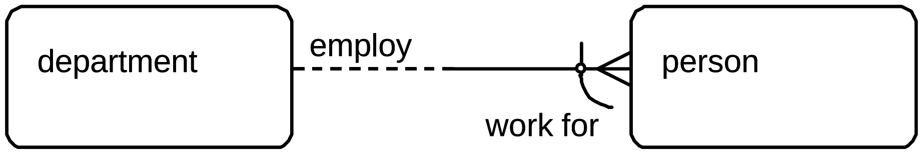
b. The ‘inherited unique identifier’ constraint



c. The ‘non-transferability’ constraint on a mandatory relationship



d. The ‘non-transferability’ constraint on an optional relationship



e. The ‘exclusive arc’ constraint

Figure 2 Additional Constraints

Figure 2c shows the ‘non-transferability’ constraint on a mandatory relationship. A person will only exist when they work for a department and will never change departments. Figure 2d shows an optional relationship where a person may exist for some time (independently) and then may eventually work for only one department. ‘Ever’ can be added to the relationship sentence such that ‘A person must work for one and only one department ever’ in the case of Figure 2c, and ‘A person may work for one and only one department ever' in the case of Figure 2d. It is worth noting that Simsion [2001] argues that it would be more useful to have the opposite 'transferable' notation.

The fourth constraint, shown in Figure 2e, is the 'exclusive arc' where a relationship is made either with this entity-type or with some other entity-type, where the arced relationships are exclusive. For example, a person must either work for one and only one department or must .... (or must ...)'. This constraint is almost always necessary in a domain and is widely used.

It is worth noting that n-ary relationships are not part of the Barker notation. Like the notation described in Chen's 1976 paper, n-ary and particularly ternary relationships have not been adopted by practitioners. Indeed, there is a strong oral tradition that practitioners were influenced more by the work of Bachman on database design. Evidence on the use of n-ary relationships in practice suggests that ternary notation is actually counter-productive [Hitchman 1999]. N-ary relationships, like many-to-many relationships, would in any case be converted to entity-types in the Barker method. Therefore this discussion excludes n-ary relationships.

It is also possible to include attribute detail on the diagram and these conventions are shown in Figure 3. The hash (#) flags the unique identifier, a star(*) mandatory and an open circle (o) optional attributes. When implemented as a table, person will have a posted foreign key from department and in many notations this notation is also included in the entity-relationship diagram. In the Barker standard, posted keys are not included as they would replicate the relationship specification. The CASE tool repository will also contain other attribute specific constraints, such as range checks, valid value lists, alternate unique identifiers and definitions that do not appear on the Barker diagram notation.

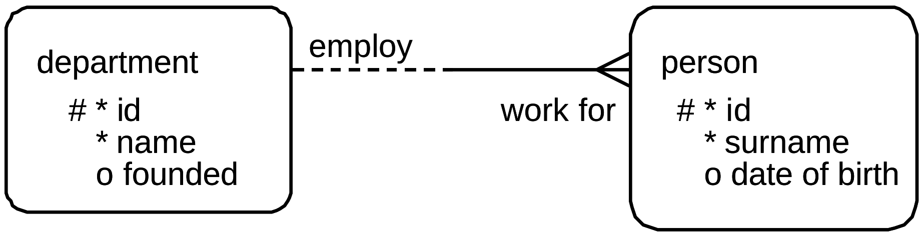


Figure 3. Attribute Conventions

III. THE IMPORTANCE OF NOTATION AND NORMATIVE LANGUAGE

The previous description includes most of the notation one needs to know to practise data modelling. We would also need to know about sub-type notation (which is 'box in box'), for example. The surprising thing about modelling is that the notation is simple but using it correctly and effectively is considered quite difficult to achieve. The aim of this section is to show that understanding a previously unexplored area of notation leads to a much clearer view of what is happening when we model. Going right back to the basics helps us understand what we are really doing.

Business relationships are only part of the entity-relationship model. It is assumed that a formal definition of each entity-type is mandatory and there may optionally be other contextual information such as:

- an informal list of examples;
- an informal list of synonyms and context

- an entity-type state diagram;

This contextual information may be essential to understand the relationship specification and is recorded in some form of repository. It is up to the experience of the modeller to deal with informal aspects of the model.

The scope of the language and notation is restricted as far as possible. For example, there is notation and normative language defined for exclusive arcs but not for inclusion. The rationale for this notation is that, in the business domain, we will very rarely, if at all, need to use an inclusive relationship concept to make sense of the domain in a useful way. If this situation does occur, we can document it in some informal way – for example with the entity-type definition. The notation would otherwise be cluttered with extra symbols that make it harder to use when trying to understand a domain sufficiently. This is the 'too infrequent too complex' decision.

The Barker notation is therefore simple and, importantly, enables the diagram to be directly read with the two way sentences (TWS). The sentence pairs and the diagram convey the *same* information with the exception that we use the plural entity-type name from the repository to complete the sentences with a 'one or more' cardinality. Given that the names of the entity-types and relationships are from the business domain, we therefore construct a set of 'English-like' sentences that any domain expert will recognize. This is not a technical diagram. Therefore, Barker doesn't just *define* relationships; the definition also shows how to *use* the formalism. The formalism becomes the 'normative language' of the specification; what is asked and what is then understood about the domain. Many writers think of the diagram as the end point but the notation and normative language influence how the modelling occurs to get to the end point.

A feature of the normative language is that each sentence can be judged to be either true or false. If every sentence is verified, then a simple way of verifying the complete model is available: the modellers can 'read through' the diagram. Therefore TWS allow very strong validation techniques, and provide a simple way of examining each association to confirm its cardinality and optionality. Fairly obviously, in the example we have used about people and departments, it is not the case that people work for one and only one department. People will move departments. Using TWS and making the statement about 'only one department' should highlight the mistake. More importantly it would be unlikely that the assertion would be made incorrectly in the first place. Of course, not all business relationships are this easy to understand.

By formalising the association the modeller is asked to investigate, in detail, important aspects of domain knowledge, that may otherwise go unquestioned. This approach results in a formalised way of knowing that the 'right' questions were asked. The implication is that, if this level of formalism is not apparent, then it is up to the modelling team to be sure that their understanding of domain associations is sufficient and accurate. This is not to say that we want to interpret the 'natural language' of the business. TWS are not an attempt at some form of 'natural language sentence' in the wider sense but represent the constrained way to talk about the business. Experience leads us to believe that Barker's is the language we need to use to understand the data. The modeller is making sense of the domain in their own linguistic context.

At this point in the argument we can see that the Barker notation makes it possible to turn our view of modelling on its head. Instead of being directed to the visual production of a diagram, we can instead view the diagram as a by product of what is really happening. Modelling relationships is mostly constructing TWS that make sense, based on talking through the data. TWS is why the normative language, and by implication the notation, is really important. The argument, then, is that it is more important to understand the normative language than to define concepts formally with predicate logic because we need to now how to use the notation. There is a strong suggestion here that linguistics is at the heart of notation and needs to be accounted for in any definitions; however this suggestion is beyond the scope of this paper.

IV. ALTERNATIVE NORMATIVE LANGUAGES FOR RELATIONSHIPS – THE LESS FORMAL APPROACH TO NOTATIONS

Another school of thought on the normative language of business relationships takes a less formal approach across many seemingly different notations and is now represented by the UML. The notations are informal since the modeller is not required to conform to a notation-dependent language custom and most importantly is not required to complete relationship sentences formally. This section compares the informal approach of several different notations. Again, although there are many notations to choose among, only a few are examined here. The objective is to understand the informal approach to relationships and the normative language effect

Very little is documented about why particular notations are used. Texts will generally state what a particular notation means without giving a rationale for particular symbols or saying why the symbol chosen is to be preferred to those already available. The reasons for choosing particular diagramming notations are generally shrouded in mystery. Informal evidence implies that the Martin notation was suggested because it was ideal for use with pen and paper. In this notation, an initial step in the diagram is to suggest, by drawing a line, that two entity-types are related. Other notations are added to this line - the open circle, bar and crow's feet - to indicate cardinality. Similarly the use of 'soft boxes' (rectangles with rounded corners) is a deliberate choice to make the diagram look 'easier on the eye'.

One other example of the mystery surrounding notations is the naming of the 'crow's foot' notation indicating 'one or more'. This notation can lead to discussions such as 'have we missed the crow's foot off this relationship', which is, apparently, bizarre. The reason that this notation should be popularly named after a visual similarity with a child's drawing or cartoon of a bird's foot may well reflect the fact that no 'natural' name exists for this device. The crow's foot notation gives a visual clue of fan out, or attachment to many entities, and may be simply the result of reversing an arrowhead. Arrowheads were commonly used in flow diagrams to link boxes, but their use is frowned on in ERM diagrams as no flow is implied by a relationship.

Figure 4 shows four relationship notations that look completely different. It is difficult to see that Barker and Bachman work the same way but use different symbols. In Bachman notation, the filled dot corresponds to the solid line, the empty dot to the dashed line and the arrowhead represents 'one or more' like the crow's foot. Both notations attempt to give a visual clue to optionality. Bachman also stipulates TWS and it is this common normative language that makes the apparently different notations so similar; the symbols both correspond and they are in the same positions on the diagram.

Martin and UML represent the alternative school of notation and share the same difference with the Barker standard. Martin and UML symbols correspond

- the crow's feet with the '*',
- the zeros,
- and the bars and 1's.

However, Martin and UML notations are 'in different places' to those of Barker and Bachman. Instead of splitting the optionality and cardinality information, the less formal notation combines it into one end of the relationship – correspondingly symbols are in different places. This combination turns out to have a subtle but far reaching effect on the normative language. Table 2 shows a comparison of association sentences, resulting from the different approaches of the Barker and the Martin [1990] notation.

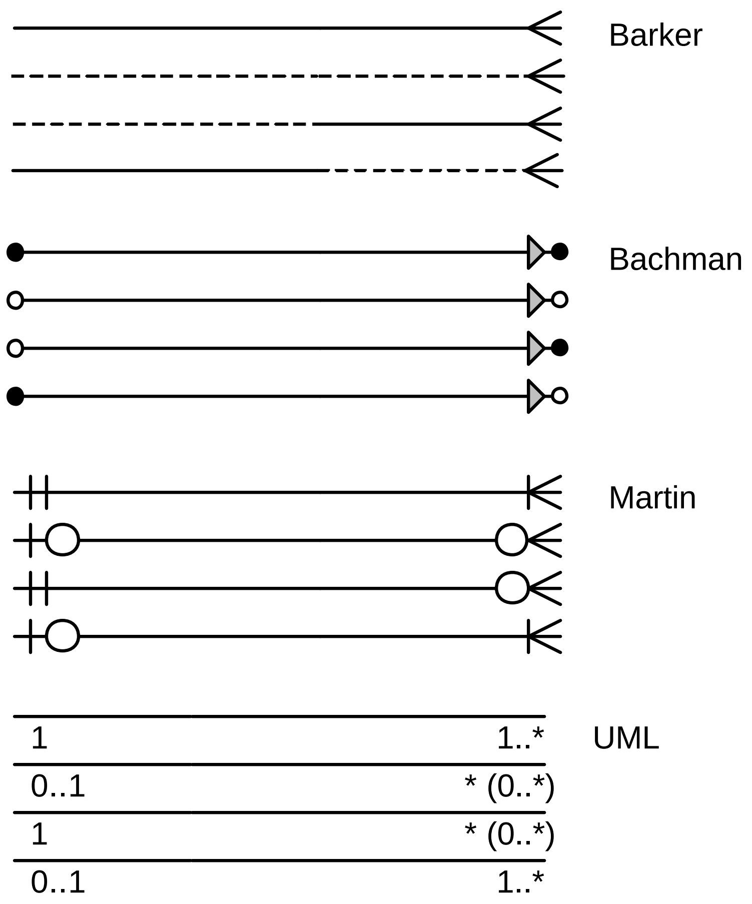


Figure 4. Alternative Relationship Notations

Table 2. Association Sentences in Alternative ERM Standards

Barker	**Martin**
A department may employ one or more people	A department employs zero, one or more people
A person must work for one and only one department	A person works for one department

THE MARTIN NOTATION

The Martin standard was chosen for discussion here since it is also widely used by practitioners, is supported by CASE tools, and illustrates the key difference in approach between various ERM standards. In this alternative approach, the optionality and cardinality information are compressed into simply 'cardinality' information that is placed at the 'far' end of the relationship line. The concept of 'may ... one or more' becomes a compressed '... zero one or more'.

This doesn't seem to amount to much of a difference – and the advice seems similar to Barker's:

> "Lines between boxes are bi-directional. The line could be read in either direction. ... The information ought to read like an English sentence" [Martin 1990, p.310];
>
> "It is desirable that the label on the link compose a sentence. ... This sentence building should be enforced as a discipline of entity analysis." [Martin & McClure 1985,p.303].

These statements clearly reflects thinking in the Barker approach although the 'zero one or more' sentence produced is less elegant in business terms. The difference is that this is now a sentence. Martin [1990] advises that labelling in one direction will generally make the other label name obvious. The clumping of the cardinality information, as it were away from the relationship name information, leads to the notation and normative language being seen as separate issues:

> "It is usually necessary to label in only one direction" Martin [1990,p.310];
> "Some data analysts like to label every link. This takes time and the additional work is often not worthwhile. The meanings of most links are obvious. When a label could have alternate meanings, it should be labelled." Martin & McClure [1985,p.304].

So we have the comparison with the Barker formalism shown in Figure 5. By naming in only one direction, there is no formalised check on the cardinality of both relationship ends. Figure 5 presents an implied relationship, say 'is included in', which may be generated each time a relationship is read (and may be inconsistent from reading to reading). TWS equivalents to the information in the diagram are no longer available. Perhaps more important, it is now no longer obvious that TWS exist - the relationship name is untangled from the cardinality.

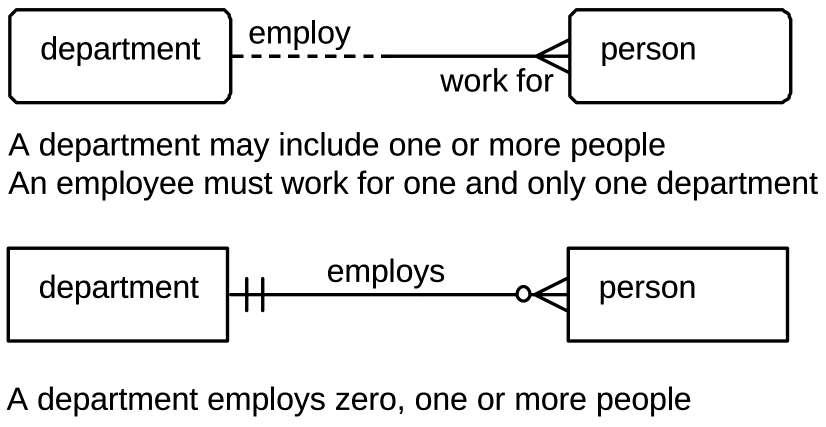

department
employ
work for
person
A department may include one or more people
An employee must work for one and only one department
department
employs
person
A department employs zero, one or more people

Figure 5. The Use of Association Naming in The Barker and Martin Formalism

Formality in the Martin notation is sometimes made stronger by the use of the 'double bar' to indicate 'one'. This double bar specifies 'one and only one', as opposed to either 'zero or one' or 'zero or one or more'. In many other standards the 'one' has no symbol, i.e. the lack of a symbol implies 'one'. In the Martin standard, a lack of symbols is considered to be potentially dangerous; perhaps the analyst has simply forgotten to check. Therefore, the explicit use of a 'bar' in the Martin ERM is designed to ensure that the 'one' issue was considered. Here, then, the diagram performs some of the function of the TWS check in Barker. The double bar, however, is not used in many implementations of the Martin standard, and the single bar is more commonly found in texts and CASE tools

Confusingly, the same notation can be used with a different normative language. Finkelstein [1989, p.52] phrases sentences using (the equivalent of) 'A department may have one or many people, or none' and uses TWS. When an empty circle, a single bar and a crow's foot are used together, Finkelstein [1989, p.53] uses (the equivalent of) 'a department may initially have no people, but must at some time have at least one, or many people'. So Martin and Finkelstein notation look the same but are read differently. It is necessary to know both the notation and how the normative language works to understand what the model means. Therefore, we cannot be expected to understand a diagram visually without having been told the normative language.

THE OMT NOTATION AND ROLE NAMING

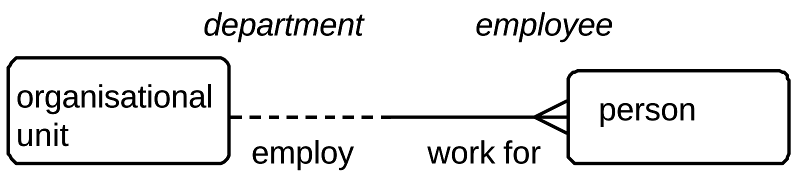
The Object Modelling Technique (OMT) notation, the predecessor of UML, uses very oblique visual notation clues – a filled circle ('●') signifies 'zero or more', whereas the empty circle ('O') signifies 'zero or one'. The difference between a filled circle and a '1+' is the (zero) optionality – a very complex visual cue. In OMT the clumping of cardinality and optionality is named as 'multiplicity' and given a status outside the relationship name. Although "Associations are inherently bi-directional… The name of a binary association usually reads in a particular direction ... In reality both directions are equally meaningful ..." Rumbaugh et al. [1991,pp.27-28], there is no formalised handling of association naming and sentences. The association name is optional if it is considered to be obvious. The normative language could be similar to Martin, with the use of ... zero or one (see for example [Rumbaugh, 1991, p.45]). In some examples the 'one' is not used, and the sentence is '... many', rather than '... one or many' (for example,[Rumbaugh et al. 1991, p.30]). The use of '... may have one', '... may have many' can also be found instead of the 'zero or many'. In OMT the notation is very distant from TWS and it is therefore difficult to see how to 'talk about' and use the notation.

A further complication in use arises with the option of using role names instead of relationship names. "Use of role names provides a way of traversing associations from an object at one end, without explicitly mentioning the association …A person assumes the role of employee with respect to a company; a company assumes the role of employer with respect to a person." ... Use of role names is optional, but it is often easier and less confusing to assign role names instead of, or in addition to, association names" [Rumbaugh et al. 1991,p.34]. The foregoing is the further extreme in the less formal approach - role relationship naming is now completely divorced from the multiplicity information.

Jacobson [1995,p.118-119] gives several reasons for preferring role names to association names. Firstly, asking questions concerning roles 'seems the natural thing to do' although the claim of 'the natural thing to do' is found in many prescriptive texts to support diverse ways of doing similar things. Secondly, 'relations normally exert influence in one direction'. Thirdly, using role names helps to find a structure in the object model. However, from the formal point of view much seems to have been lost for indeterminate gain. The comparative naming of the entity-type and the role is obviously problematic – we needed to change 'department' to 'organisational unit'. It is quite possible, however, to extend the Barker normative language to account for role names. A possible extension is shown in Figure 6.

THE UML NOTATION

The Universal Modelling Language (UML, see for example [Booch & Rumbaugh, 1997]) made some detailed changes from OMT. Not surprisingly the cardinality indicator of circles was replaced. The relationship name is given an arrow indicating the direction for reading the relationship name and a normative language is suggested which would be "A person acting as an employee works for a department as an employer". However, the approach is inconsistent as role names can be omitted where the relationship membership names seem better or where it is difficult to find a role name. Similarly, the relationship name can be omitted. The multiplicity and naming are still separated and although the notation is essentially the same as Martin (Figure 4) with different symbols, there is no hint that TWS are involved.



A person taking the role of employee
must work for one and only one
organisational unit taking the role of department
An organisation unit taking the role of department
may employ one or more
people taking the role of employees

Figure 6. Normative Language for Roles

Some learning from ERM occurred in the development of UML. Exclusive relationships are not part of OMT [Rumbaugh et al., 1991], but have been incorporated in the later development of the UML Notation and according to Booch & Rumbaugh [1997,p.7] "Occasionally one class can participate in two associations, but each object can only participate in one association at a time. This association can be shown by placing an "or"". So this aspect of the Barker standard has taken some five years to become included in UML developments.

It is possible to say, then, that although named the Universal Modelling *Language*, it is precisely the relationship language that is not defined formally.

THE ORM NOTATION

The situation in another paradigm, the Object Role Model (ORM, [Halpin 1995]), is similar to Martin, although using an example based normative language. For example, Halpin advises that "... If desired the inverse predicate may be included..." [Halpin 1995, p.62]. Scanning through an ORM text there are few examples of TWS. Nijssen and Halpin [1989, p.52] provide a rationale for this absence that is similar to that given by Martin and McClure [1985]. The key addition is the example population that can be shown on the diagram that also implies the constraints. Therefore, the user may confirm the model partly with the example population. This notation would result in fact statements similar to 'Department (1) includes person (22)'. This difference is interesting from the 'more abstract' TWS, but carries with it the problem of large diagram sizes and complex notation in return for formalising some of the informal aspects of Barker.

INFORMAL NOTATIONS ARE DIFFERENT

The variation in notation is large, but ideas about relationship naming in the less formal school are similar, with the addition of role naming in the object-oriented paradigm. Both formal and informal approaches have diverse notations that make notation seem to be of secondary importance. However, this viewpoint disguises the fundamental difference that a notation designed for TWS makes. The Barker notation makes a good benchmark because it formalises the normative language. Comparing notations with the Barker benchmark shows a stark difference in formality that intuitively suggests that the formal approach is less risky. This notational difference was previously unexplored in the literature.

We cannot conclude, nor are we interested in suggesting, that experienced modellers in the formal school would necessarily produce better models (whatever 'better' may mean). Although this hypothesis might be an interesting area for empirical research it could be argued that a well trained experienced modeller would only require an informal use of TWS, some of the time. What is at issue here is knowing about TWS and having that tool available when experience dictates its need. Knowing and understanding about TWS could be thought of as a key element in being able to run a modelling session since understanding TWS is understanding the language of modelling. Understanding TWS is not the same as formally, and unthinkingly, following the rules. When Martin & McClure [1985,p.304] advise us that thoroughly naming relationships is not necessary, they speak as highly experienced modellers. Therefore, like learning in general, it should pay to follow the rules carefully when learning modelling in order to be proficient. Learning through using a notation that does not make TWS explicit will risk poor specification and ineffective modelling meetings. At the more extreme end of the alternative school, the UML trained practitioner will not be able to be aware of TWS at all.

V. CONCLUSIONS

Contrary to previous assumptions, notation detail was shown to be really important in understanding business relationships because of the effect of normative language. The suggestion is that normative language is a useful way of understanding notations and relationship modelling in a linguistic context. The research literature previously overlooked the Barker relationship definition which was shown to provide a useful way to understand how to use business relationships in practice – whereas current definitions of prepositional logic are difficult to apply because they lack the essential element of *use.* Comparisons with the Barker benchmark notation reveal different levels of normative language formality. The less formal approaches arguably put the less experienced modeller at a disadvantage, risking ineffective modelling. Verifying this assertion should be a useful area for future research.

Editor's Note: This article was received on April 2, 2002. It was with the author four weeks for four revisions. It was published on September 17, 2002

REFERENCES

Barker, R. (1989) *Case*Method: Entity Relationship Modelling* , Boston MA: Addison Wesley

Bachman, C. W. (1969) Data Structure Diagrams Database: *A Quarterly Newsletter Of SIGBDP*, Summer, **1**,2\

Booch, G. and Rumbaugh, J. (1997) *The Unified Notation* , Cupertino, CA: Rational Software Corporation

Chen, PP-S. (1976) The Entity Relationship Model: Towards a unified view of data, ACM Transactions on Database Systems 1(1) pp. 9-36

Checkland, P and Holwell, S. (1998*) Information, Systems, and Information Systems: Making Sense Of The Field*, Chichester: Wiley

Dorsey, P & Hudicka, J (1999) *Design Using UML Object Modeling*, Berkeley, CA: Oracle Press-McGraw-Hill

Finkelstein, C. (1989) *An Introduction To Information Engineering: From Strategic Planning To Information Systems*. Boston MA: Addison-Wesley

Gane, C. and Loosley, C. (1990) Information Systems Modeling Part 3 *InfoDB* Winter 1990/91 pp. 28-36

Halpin, T., A. (1995) *Conceptual Schema and Relational Database Design*. 2nd Edition. Upper Saddle River, NJ, Prentice Hall
Hitchman, S. (1995) Practitioner perceptions on the use of some semantic concepts in the entity-relationship model. *European Journal of Information Systems,* **4**, 31-40

Hitchman, S. (1999) Ternary Relationships – to three or not to three, is there a question ? *European Journal Of Information Systems* Vol 8 December pp.224-231

Hitchman, S. (2000) Object-Oriented Modelling In Practice: Class Model Perceptions In The ERM Context, *Proceedings of the 19th International Conference on Conceptual Modelling*, ER2000, Edited by Liddle, S.W, Mayr, C & Thalheim, B. Salt Lake City, USA 9-12 October 2000 (published as Heidelberg: Springer Verlag Lecture Notes In Computer Science volume 1920, ISBN 30540-41072-4) pp.397-408

Jacobson, I., Ericsson, M., Jacobson, A. (1995) *The Object Advantage: Business Process Reeingineering With Object Technology ,* Boston MA*:* Addison-Wesley

Martin, J. & McClure, C. (1985) *Structured Techniques For Computing* ,Upper Saddle River, NJ:Prentice Hall

Martin, J. (1990) *Information Engineering - A Trilogy* Upper Saddle River, NJ: Prentice Hall (Details of the notation are in Book 2:Planning and Analysis, pp. 162-163, and in Book3: Design and Construction p.58)

Nijssen, G.,M. & Halpin,T.,A. (1989) *Conceptual Schema & Relational Database Design: A Fact Oriented Approach* , Upper Saddle River, NJ: Prentice Hall

Nordbotten, J. C. & Crosby, M.E. (1999) The effect of graphic style on data model interpretation. *Information Systems Journal* **9** pp.139-155

Ortner, E. and Schienmann, B. (1996) Normative Language Approach: A Framework for Understanding. *Proceedings of 15th International Conference on Conceptual Modelling, ER'96*, Cottbus, Germany, October. Heidelberg: Springer Verlag as Lecture Notes in Computer Science 1157. pp.261-276

Rumbaugh, J., Blaha, M., Permalani, W., Eddy, F. and Lorensen, W. (1991) *Object Oriented Modelling and Design* , Upper Saddle River, NJ: Prentice Hall

Simsion, G. (2001) *Data Modeling Essentials* (2nd Edition), Scottsdale, AR: Coriolis

Thalheim, B. (1999) *Entity-Relationship Modeling* , Heidelberg : Springer Verlag

ABOUT THE AUTHOR

Steve Hitchman has been lecturing and consulting in data modelling issues for over ten years. This paper was written during a sabbatical at Melbourne University. For the previous three years he worked as a corporate modeller and data architect within several UK banks and insurance companies.

Communications of the Association for Information Systems

ISSN: 1529-3181