EROSS-2020 Reference Scheme Modelling questions and Answers

Terry Halpin, 2020 Sep 13

Anonymous: NIAM was first proposed in a period where many semantic data modeling languages were being proposed (e.g., ER, Abrial's Semantic Data Model). What were the foundations NIAM/ORM based on? Does it connect philosophically to this view of the world being composed of Facts?

Terry: In the 1970s, especially in Europe, substantial research was carried out to provide high level semantics for modelling information systems. Jean Abrial, Mike Senko and others modelled binary relationships. Eckhard Falkenberg generalized their work to *n*-ary relationships and decided that attributes should not be used at the conceptual level because they involved "fuzzy" distinctions and they also complicated schema evolution. Falkenberg's PhD thesis proposed the fundamental ORM framework, which he called the "object-role model". This framework allowed *n*-ary and nested relationships, but depicted roles with arrowed lines.

Sjir Nijssen adapted this framework by introducing the circle-box notation for objects and roles and adding a linguistic orientation and design procedure to provide a modelling method called ENALIM (Evolving NAtural Language Information Model). A major reason for the role-box notation was to facilitate validation using sample populations. Nijssen led a group of researchers at Control Data in Belgium who developed the method further, including Franz van Assche who classified object types into lexical object types (LOTs) and non-lexical object types (NOLOTs). Today, LOTs are often called "value types" and NOLOTs are called "entity types". Bill Kent provided several semantic insights and clarified many conceptual issues.

Robert Meersman added subtypes and made major contributions to the RIDL query language with Falkenberg and Nijssen. The method was renamed "aN Information Analysis Method" (NIAM). In later years the acronym "NIAM" was given different expansions, and is now known as the "Natural language Information Analysis Method".

In the 1980s, Falkenberg and Nijssen worked jointly on the design procedure and moved to the University of Queensland, where the method was enhanced further. It was there in 1989 that Terry Halpin provided the first full formalization of the method, including schema equivalence proofs, and made several refinements and extensions.

In the early 1990s, Halpin developed an extended version of NIAM called ORM (Object-Role Modeling), initially supported in the InfoDesigner modelling tool which later evolved into InfoModeler, then VisioModeler, then Visio for Enterprise Architects. Anthony Bloesch and Terry Halpin designed an associated query language called ConQuer, supported in the ActiveQuery tool. Microsoft acquired Visio in 2000 and modified its ORM solution for use in Visual Studio.

ORM is one of many related methods known as *fact-based modelling* (FBM), where all facts are represented by relationships (unary or longer) with no use of attributes. Although most ORM proponents favour *n*-ary relationships, some prefer binary-relationship modelling. The Predicator Set Model (PSM) was developed mainly by Arthur ter Hofstede, Erik Proper, and Theo van der Weide, and includes complex object constructors. Olga De Troyer and Robert Meersman developed another version with constructors called Natural Object-Relationship Model (NORM). Harm van der Lek and others developed Fully Communication Oriented Information Modeling (FCO-IM), and Shir Nijssen's company PNA now supports CogNIAM. Various other FBM dialects exist and have associated tool support. For further historical details see pp. 107-108 of Halpin & Morgan (2008).

Anonymous: How does ORM relate to alternative evolutions of NIAM such as CogNIAM?

Terry: Though similar in most respects, ORM and CogNIAM differ in their range of constructs, some semantics (e.g. the definition of "role"), and in tool support. See above for some other FBM dialects.

Anonymous: Could you please elaborate on domains in industry in which Conceptual Modeling is perceived to play a fundamental role? space? Finance? Does the emergence of numerical methods (e.g., machine learning) render Conceptual modeling somehow obsolete?

Terry: Conceptual modelling is relevant for modelling *any* universe of discourse, and has been used successfully in hundreds of different business domains. Machine learning and AI do not render conceptual modelling obsolete, as one should first understand the domain in terms of concepts that are naturally used by the domain experts.

Giancarlo: Terry, your work and background spans through philosophy, logics, linguistics (besides computers science, of course). Could you please tell us a bit about your personal trajectory? How did you first get interested in Conceptual Modeling? we have many students here, so could you also tell us a bit about your experience of working with Conceptual Modeling in industry? Thanks!

Terry: My masters degree focused on logic and philosophy, which provided an ideal basis for other studies. My introduction to conceptual modelling was NIAM, as taught by Falkenberg and Nijssen, and my PhD on the formalization of NIAM drew heavily from my logic background (e.g. schema equivalence proofs). As well as teaching as an academic in computer science, I worked on conceptual modelling in industry for various companies (e.g. InfoModelers, Visio, Microsoft, LogicBlox), as well as consulting for various companies (e.g. ESA). In working with domain experts to create conceptual models, I learnt early on the value of using controlled natural language to verbalize the semantics, and populating fact types with small examples and counterexamples to check whether a constraint is actually intended.

Giancarlo: In your example of "everyone is born in a country but we don't have to know which one" you are in a subtle way hinting for the difference between representing facts in the world and representing epistemic states of the system. In the past you also wrote about the importance of distinguishing Alethic and Deontic modality in constraints? These distinctions are normally not made very explicit in conceptual modeling approaches (neither from a language nor from a methodological point of view). Could you comment on that?

Terry: That example was indeed contrasting the difference between open world semantics and closed world semantics. On a separate issue, ORM allows any constraint to be declared alethic (must be obeyed) or deontic (ought to be obeyed). For example, in a monogamous society, a deontic uniqueness constraint might be declared that each person ought to have at most one wife. Updates that violate an alethic constraint are simply rejected. If a deontic constraint is violated, the update is allowed and the violation is noted (e.g. a message may be generated about the violation).

Bernhard: The largest ORM schema I know and have seen so far contains more than 10.000 types (it is the corridor wall painting in the Boing research center tower - there you have to run from floor to floor for capturing the connections in the schema). How you can handle abstraction nowadays in ORM, e.g. zoom-in and zoom-out facilities? How you can handle consistency among constraints?

Terry: The largest ORM model I've seen was at 3M many years ago. The generated relational schema had at least 1000 tables, and filled various walls. Various abstraction mechanisms can be used in ORM. For example, NORMA supports modularization (e.g. divide a schema into as many pages as you wish), object-type zoom (select any desired object type and view its immediate neighbourhood), etc. See pp. 852-857 of Halpin & Morgan (2008) for further discussion. The NORMA tool automatically analyses constraint patterns and provides a warning message for most cases of constraint inconsistency. However, it doesn't cover all possible cases. Enrico Franconi's team at Bolzano university has developed a plug-in to NORMA that performs a much deeper logical analysis allowing it to detect some inconsistencies not detectable by NORMA alone.

Erik Proper: Great overview Terry. Things certainly evolved since our time in OZ. A concern I have is that sometimes discussions around reference schemes are really driven by "database needs" or more general "administrative needs". As such, people may argue that reference schemes are not (not always at least) conceptual. What are your views on this?

Terry: Good to hear from you Erik. I guess you are referring to the fairly widespread practice in industry of using meaningless object identifiers (oids), as the primary way to reference objects. This is sometimes useful, especially for performance and stability, but unless a meaningful linguistic reference scheme is also provided, there is no reliable way to validate the model with domain experts. I once attended a "master class" on UML where the presenter was unaware of many mistakes in his models simply because he never used real examples to populate them. That said, I have also met some UML modelers who are good at modelling.

Anonymous: How does you background as a philosopher influenced your work in conceptual modeling? Logics is an obvious influence, but did ontology (in the philosophical sense) also play a role there? Do you think there are open philosophical problems that if solved could make an impact in Conceptual Modeling?

Terry: Logic enabled me to fully formalize ORM, and really helped with schema equivalence and derivation rules. Ontology, epistemology and linguistic philosophy also helped. Consider objectification of relationships. In UML an association class is both an association and a class. In contrast, ORM facts are propositions taken to be true by the business, and objectification of a fact is a case of situational nominalization (the object formed is a situation or state of affairs in 1:1 correspondence to but not identical with the original fact) not propositional nominalization. I believe this matches reality better than UML's approach.

Anonymous: Why did you develop your own modeling language instead of fixing an existing one that was already used by people? Did you consider all of them too far down the wrong road for your goals?

Terry: My work on ORM began by fixing and extending NIAM. Later on, the company I was working for in developing tool support wanted a name to distinguish it from NIAM, so I chose "ORM" based on the original name used by Falkenberg in his PhD thesis ("object-role model").

Tatyana: Dr. Halpin, thanks a lot for presenting an analysis of languages in a great methodological manner. Have you planned to consider other modelling problems in the same way in your lectures or publications? for instance, relations modeling and reference schemes for relations. Thanks a lot!

Terry: Thanks for your kind comments Tatyana. I guess by "relations" you are talking about relationships or relationship sets. In ORM, relationship types may be unary (e.g. Person smokes), binary (e.g. Person plays Sport), ternary (e.g. Person played Sport for Country), or longer. A fact type is a non-empty set of typed predicates, e.g. "Person plays Sport" and "Sport is played by Person" are fact type readings for the same fact type. In ORM, objects have reference schemes, but relationships do not. Objectification of a relationship in ORM results in an object that is not identical to that relationship. See two comments above for some more on that.

Giancarlo: It is very interesting that, beyond the language itself, the approach has such a strong connection to other aspects that enhance the "comprehensibility appropriateness" of the approach. This includes obviously verbalization, population generation but also the concrete syntax itself seems to have been design with aspects of visual pragmatics in mind. Could you comment on these aspects?

Terry: Verbalization and (positive and negative) populations are critical for validating models with clients. ORM's graphical syntax is designed to make it easier for the modeler to visualize the model and to capture the wide variety of constraints found in real business domains. The primitive elements (e.g. mandatory role, uniqueness and frequency constraints) are orthogonal, facilitating changes and efficient implementation.