The Problems of Data Modeling in Software Practice

Harald Huber
USU Softwarehaus, Spitalhof, D-71696 Möglingen

Abstract

This paper presents, from the author's perspective, the problems that occur in practice during data modelling. The author's experiences are a result of a considerable number of projects which he carried out in the framework of his consultancy role at USU Softwarehaus in Möglingen (Germany).

These projects concerned the following themes:

- Corporate Datamodelling
- Comparing Datamodels
- Project (Application)- related Data modelling.

In all cases, E/R-notation was the chosen representation-form. From these experiences, the author formed an impression of the problems that occur in practice when defining a data model. These problems have, however, also led to the author's increased interest in knowledge representation, in turn leading to his usage of KR-methods in practice. This has shown itself to be quite effective.

Sections 2 and 3 briefly illustrate the recommendations and the experiences arising from their usage in projects.

1 Datamodelling in Practice - the Problems

Datamodelling was still up until recently the buzzword with which one believed to be able to solve the software crisis. CASE products concentrated on this area, meta-databases were created using a data-modelling process (E/R), and large companies invested millions in order to acquire a corporate data model. Although this trend has subsided a little, the theme in general is still of current interest. What Chen already recognised as an important benefit when presenting the E/R-Model, is today still seen as a key effect of a data model: the representation provides a standard communications basis with which understanding between DP and users is more easily accomplished.

This however, unfortunately seems to hold just for small data models. For larger areas of attention, the methodology starts to become ineffective, and no longer provides the overview required. Apparently, there are just a few 'gurus' who are able to create a complete complex data model. Often this data model quickly decreases in value, as soon as that person leaves the company. Director's offices exist in which the corporate data model is hanging up behind glass - however, this is regrettably the only place in which the data model is noticed or paid heed to.

The following problems, among others, have been recognised:

1.1 Low Expressivness of a Data Model in E/R-Form

During the analysis phase, many of the organisation's interdependencies and processes are identified. These are subsequently, to use the relativly inadequate language of the E/R-Model, abstracted and generalised. This often requires a change in terminology; in other words a unified, formal language is compulsory. What many authors (e.g. Vetter) see as an advantage of data modelling (exactly this coming-into-being of a corporate, unified terminology) often turns out to be a disadvantage: the terms used in the data model are not understood by the user departments. To make matters worse, these terms are mostly held in commentary form (if at all). Also the cross-reference of the new, unified terminology to the terms used in the departments is, in most cases, not documented at all. This makes understanding the data Model afterwards very difficult (see 1.6).

1.2 The Development of the Data Model is not Documented

A model undergoes many changes during the modelling phase. Requirements, ideas and practical examples from the user department contribute to the permanent extension and improvement of the model. Consequently, variations in the Business Processes are represented by generalisations, and classes (e.g. Subtypes) are created in order to denote similar 'things' in the model. The problem is that in nearly every case the documentation of this development is missing, i.e. reasons and reflections on which the model's structures and elements are founded will be lost after a short time. This results in difficulties if the model is changed due to further development or new requirements.

1.3 The Ideal Model is Developed

Although the user departments are consulted during the analysis phase, in practice one is often left with the impression that the DP-staff's ideal model is developed. This trend is strengthened by the fact that the creation of the data model requires a change in terminology and a certain generalisation (see 1.1). The user department staff usually see themselves therefore as incapable of effectively contradicting the 'high-flying' ideas of their DP-colleagues. The result is mostly a model which gives the impression of absolute perfection, but which neither makes the day-to-day business its priority, nor is so understandable that the user-departments can work with it.

1.4 Weak Methodology of the Developer

The possibilities of graphical development tools and the resulting excellent representation often disguises the weakness in the developer's understanding of the methodology. In this way, entities such as 'Total Turnover' and 'Turnover per Customer' can actually be modelled. Most developers tend to model concepts as entities, instead of taking the expressive character of entities in general into account. (This behaviour is also to be seen in a completely different form, where the developers come from a very technical background and mean tables or files instead of entities. Let's leave this point for the moment - it will be touched upon again in point 1.8). Another weakness is the missing experience in interview technique. Very often, the interviewer's question is formulated like "And how can I show that in E/R?" instead of "Which process stati occur in practice - let's leave E/R out of it for the moment?".

1.5 Exceptional Cases Become the Core of Model

Since the daily business of a company is in most cases comparatively simple to represent, Data Modelling projects often rush headlong into attempting to build every case imaginable into the model, as if the knowledge for treating each of these cases really had to be documented. The effect of this is that the models quickly become too detailed and difficult to understand - so much so that the user-departments, who really should judge the model's 'correctness' - more or less make this judgement on the basis of 'gut-feeling'. If they see well-known terms and recognise relationships between them that are held to be necessary, then the model seems to them to be complete and correct, even though in many cases they cannot follow it through to the lowest detail.

1.6 Assumption of Understanding

The relatively low expressiveness of an E/R-model all-too-seldom leads to recognition of this 'inadequacy in meaning'. Often this inadequacy is compensated for by an overkill of interpretation, which means that the model, which really should be the basis for a common understanding, often becomes a problem of understanding. The real world is then no longer the topic of discussion (in which the question of understanding certainly arises) - rather, one discusses entities and relationships, whose meanings are comparatively trivial and thereby are a matter of interpretation and alteration when trying to understand the 'fact-content' behind them.

1.7 Missionary character of DP

DP tends to over-estimate itself in many organisations. This inaccurate estimation doesn't particularly affect the importance of DP for the organisation's success so much (This could certainly be the subject of heated discussion both in theory and in an organisation's leadership). This obviously false judgement of one's own situation affects the implementation of standards and norms much more. The standardisation of terminology (mentioned under point 1.1) which the DP-Department carries out during data modelling is here an excellent example. It implies however, that 0.5 - 2 % of the company can dictate the terminology of the remaining employees. This over-estimation, together with the problem outlined in 1.3, means that DP doesn't model according to requirements, rather use their own ideas as basis for the 'ideal model'.

1.8 Too much Technical Thinking

Since most modellers come from a technical background (e.g. Application Development), they find it extremely difficult to ignore this technical knowledge when modelling. In the past, many cases occured where performance considerations were incorporated into the E/R-Diagram. The problem, however, goes much deeper than that. Most modellers cannot imagine any way to represent the characteristics of entities other than with attributes. Two entities with the same attributes are hastily made one, without considering that they express a classification on a logical level.

2 Suggestion for a Solution

The approach this solution takes is basically to use to best effect the developer's (and the user department's) tendency to express himself in concepts. This means that in the initial Data modelling phase, one creates a model of these concepts in the form of a semantic network. It's quite possible that other, more modern, representations are more suitable for this task. However, since the author has his roots in the Data modelling world, moving towards semantic networks was the easier way for him to come to terms with knowledge representation methodology.

The author makes the following suggestion for the development of a Data Model (relational or E/R):

- **Creation of various semantic networks for parts of total area of attention.** These semantic networks contain all statements-of-fact and requirements issued by the user-department, in order not to let any information fall by the wayside. Representative questions from the user-departments can also be noted here.
- **Consolidation of the various networks.** The aforementioned networks are consolidated. Synonyms and homonyms are not 'cleaned up'.

This means that there is no unification of language necessary. Rather, the individual terms are cross-referenced to one another.

- **Generation of an E/R-model.** The user department requirements can be generated using all of the semantic networks. The E/R-Model can be worked on using this basis and can be tested using the requirements represented in the networks. This model is then the basis for the creation of the relational model.

To make the consolidation of several semantic networks developed by several developers possible, a standardized, unified representation of the networks is suggested. This means that only two types of associations are allowed, represented by lines; all other relevant concepts and associations appear as nodes. This restriction forces the unified representation necessary for the consolidation. The following two types of associations are allowed to be represented by lines:

- Type 1, which describes just the extension of a concept
- Type 2, which defines the intention.

Note that these associations are not defined by their symbolic meaning, rather by a relatively formal context. This has the advantage that the semantics of these associations are not interpretation-dependent.

3 Experiences from Projects

The suggested methodology solves the aformentioned problems. The interviewers interview-technique is positively affected, because his annotation is not subject to the restrictions of the E/R-model. The developement of the model is also documented, whereby the supplementary information discovered during the analysis phase, is held in the model.

- The tendency to strong generalisation and 'artificial terms' is restricted - the terminology can still be understood by the user department.
- The selection process (what's an entity?) can be re-created and checked in reviews. The user-department staff can concentrate more on the model's content, thereby avoiding 'ideal structures'.
- The cabability to consolidate the various parts means that the model in the user-department stays relatively small.
- There are, however, also disadvantages.

If one uses a strictly formal representation, as suggested above, the model becomes difficult to grasp in its entirety. Furthermore, during the interviews, the interviewer requires considerable concentration in order to express the facts in the required manner. In practice, however, during the interview a somewhat less formal representation is chosen, which is subsequently translated into a formal model.

Note that the principle elements of the model are concepts, and not other elements such as entities, even if a less formal notation is used.