Data Modelling in the Large

Martin Bertram
CAP debis Dienstleistungen GmbH
Am Salzhaus 4; D-60311 Frankfurt am Main Germany

1. INTRODUCTION

Semantic data modelling[1] is the established method for the requirements definition and the conceptual specification of application systems. In large projects and especially in enterprise data models the cost of creating a data model amount to a large proportion of the overall cost. On the other hand there is a general pressure to reduce the cost of data modelling for application systems to harness the skyrocketing costs of data processing in a company.

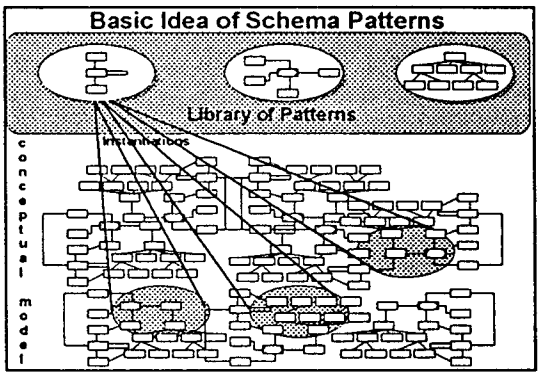
The standard textbook modelling process calls for the modelling of single entities to represent simple facts and combining these into a model in a bottom up fashion: *'An entity is a concept, person, thing ,....* One way to overcome this limitation is to reuse the components and structures of data models. This paper presents an approach to reuse by applying a modelling method that is based on the reuse of large complex abstract elements to build data models much faster by reusing these elements as building blocks. The approach is based upon a library of schema-patterns. This resembles the 'Programming in the Large' where the implementation of an application system is done by reusing predefined modules.

The main advantages of this approach are, in addition to the facilitation and shortening of the analysis phase, a substantial reduction of the costs during design and implementation achieved through the reuse of the prefabricated implementations of the schema-patterns. Additionally the quality of the application schema as a whole will improve[2].

The first part defines the term *schema-pattern* and describes shortly the process of building, maintaining and using the library of schema-patterns. In the main part, the quality criteria for these schema-patterns itself are prsented. Finally the advantages and main problem areas of the approach are discussed.

2. RELATED WORK

Most of the discussion about reuse during the systems development process is centred around the reuse of



the code of software products. Some work has been done on the reuse of a set of entities and relationships as part of a model of a specific application[3]. On the other hand some companies promote the complete adoption of a standardised reference model created for specific industry as a means of reducing the modelling effort for specific contexts.

The use of object oriented patterns has been described by [4]. This paper contains examples of patterns but offers little help about how to apply patterns in the context of an specific model. The issue of **'what constitutes a good pattern'** is hardly addressed at all.

3. THE PATTERN-LIBRARY

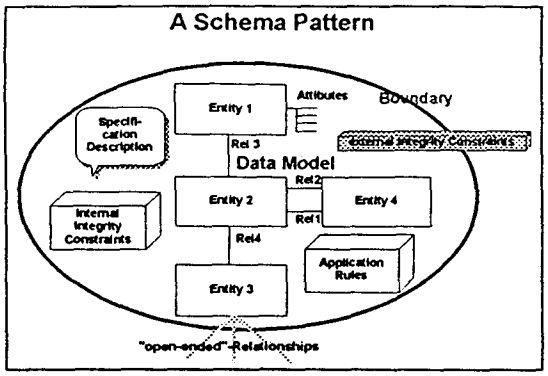
3.1. SCHEMA-PATTERN

A schema-pattern is a subset of an abstract data model. It can be instantiated to be used in application specific models. To be useful for modelling purposes, it must contain the following components:

1 As introduced by [5].

2 For schema quality, especially in enterprise data models, see [2]

3 The reuse of aplication-specific schemas is described by [3]



SPECIFICATION

- A formal model of the content i.e. what facts of an application are represented by this pattern. This should include a graphical representation, examples and synonyms.
- A systemic description what is inside the system, what outside, definition of the boundary.
- An interface which contains the 'open ended' relationships which relate the pattern to entities outside of the schema pattern, those integrity constraints involving elements inside and outside of the pattern and anything else that extends across the boundary.
- Application rules
  How to use the schema pattern, where to use, where not to use, examples of usages
- Description of the variants of the original pattern
- A classification and links to 'similar' patterns

MODEL

- A data model with abstract entities and relationships including their definitions
- Attributes with their data types
- The implicit integrity constraints
  (e.g. the cardinalities).
- Explicit integrity constraints which can not be defined using the formalism alone.

ADDITIONAL COMPONENTS

The following additional components of the definition should be included:

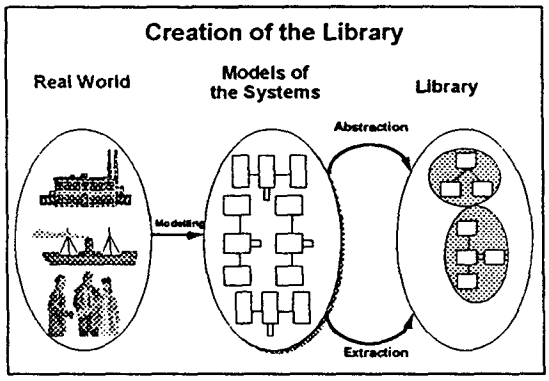
- Functionality
  To facilitate the modelling of functions as well as the modelling of data, the schema-patterns should include a set of basic abstract functions.
- State Model
  A life cycle model for the elements represented by the pattern as well as for the pattern as a whole.
- Design concepts
  The schema-pattern should include concepts for the design of the application system.
- Implementation
  For each schema-pattern there should be at least one generic implementation. The description has to include the conditions under which this implementation is optimal, as well as the conditions which make using it unfeasible.

All the additional components described above have to include the specified variants.

3.2. THE PROCESSES

The central concept in the reuse of schemes is a library which contains the schema-patterns and provides the necessary infrastructure to support the (re-)users. There are three key processes:

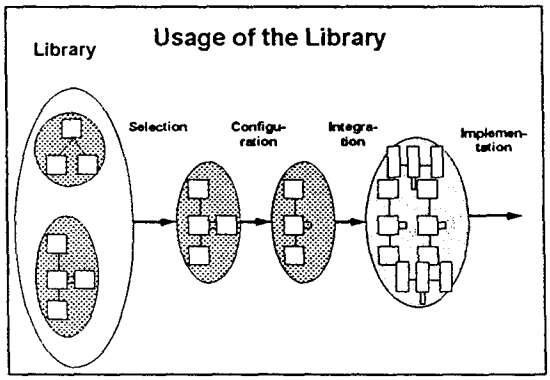
THE CREATION OF THE LIBRARY



The essential problems are here:

- How to identify the relevant schemes?
- How to describe a schema?
- What constitutes a good schema?

THE USAGE OF THE LIBRARY

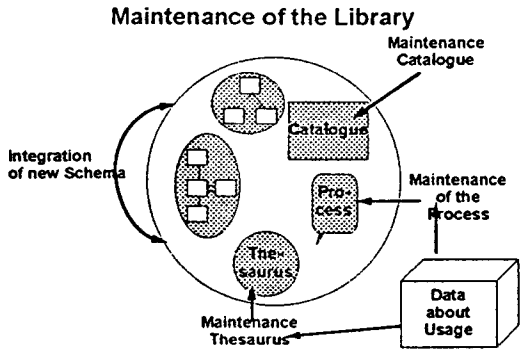


The process of modelling an application poses the following problem areas:

- How to select the applicable schema?
- How to adapt the schema to the requirements?

- How to integrate a schema in the data model of the application?
- How to evaluate the efficiency and quality of the library?

The Maintenance of the Library



The essential problems of maintenance are:

- How to integrate new schemes into the library?
- How to maintain the efficiency and quality of the library?
- How to maintain the efficiency and quality of the processes?

This paper treats the basic problem within this context: **'What constitutes a good schema-pattern.'**

The quality of the schema-pattern determines the efficiency and quality of all the processes which use the library as well as the quality of the model of the application which is based on these schemes.

4. Quality Criteria

The quality of a pattern can be described by a set of requirements regarding those characteristics which ensure that its application in a specific modelling project is both possible and efficient. Those requirements resemble those of the reuse of software components[4] respectively of a high quality decomposition architecture of software[5].

4.1. Semantic Requirements

Well Defined Application Semantic

The schema-patterns should be models of a clearly defined set of application facts. It should represent exactly one abstract application concept. The scope of this concept should be chosen neither too large to remain manageable nor too small or trivial to make up useful building blocks for the process of modelling.

Precise Specification

Specification means the description of **What** the component does, not **How** it achieves its purpose. That means that the knowledge of the internal modelling of a schema-pattern must not be a necessary prerequisite to understanding and applying it. It is essential, that such a specification is understandable not only for the systems analyst, but as well for the specialists of the application domain who have to asses the correctness and consistency of the model.

Minimal Cognitive Distance

It has to be less expensive, in terms of time and mental effort, to understand and apply the schema-pattern as it is to model this part of the application domain from scratch. The cognitive distance can be reduced by providing sufficient documentation and using the appropriate abstraction concepts.

For highly abstract patterns the effort to adapt the schema-pattern to the specific application domain may become prohibitively large.

Precise Rules for Utilisation

There have to be clear and explicit rules for the identification of those parts of the application domain where the pattern can be used as well as rules for the use of the schema-pattern itself. Their primary objective is to describe the circumstances in which the schema-pattern can be used as well as the limits of application.

Sufficient Universality

The schema-patterns should be sufficiently abstract to be applied to many different application models. They should on the other hand be not too abstract to keep the cognitive distance small.

Maximum Flexibility

The schema-patterns should be flexible enough to be adapted to most changes within the application without changing or replacing the pattern itself. This flexibility could be achieved by defining the business rules in the form of data for example by providing templates or meta structures within the data model.

4.2. Constructive Requirements

Constructability

It has to be possible to model the schema-pattern using the formalism upon which the model is based. The formalism consists not only of the graphics for the representation of entities and relationships, but as well of the language which is used to formulate the functions, rules and integrity constraints and the means of description for data types.

4 An overview is contained in [6].

5 Criteria for modularisation are given by [7].

The data model of the pattern itself should conform to the generally accepted quality standards.

COMPLETENESS

The pattern has to contain all statements about the problem domain which are correct and relevant. The pattern may not contain any statements about facts outside the problem domain or statements which are not correct or contradictory.

COMPACTNESS

A quality criterion for structure and scope of a schema-pattern is to combine only such concepts in it, which really do belong together in the sense of a functional respectively an informal coupling[6]. No two patterns should model overlapping combinations of basic concepts. One of the basic criteria is the use of a common domain. In respect to the model upon which the schema-pattern is based this means that all the model parts connected either directly by relationships, by common rules or integrity constraints.

ORTHOGONALITY

The individual schema-patterns should be constructed in a way to make as many combinations between them feasible as possible. This way the set of schema-patterns can be kept small. A prerequisite for this is, that no abstract concept should be covered by more than one pattern.

The schema-pattern has to define the exact places where other patterns can be substituted as well as the minimal requirements about the definition and domain of the patterns to be substituted there. It should not contain any references to where itself can be used or not in other patterns.

WELL DEFINED NARROW INTERFACE

A schema-pattern has to have a clearly defined interface with their context within the model to be as independent from its environment as possible. The width of the interface, i.e. the number of relationships and the complexity of the rules, should be minimal.

Based upon a precise interface definition it becomes possible to exchange one schema-pattern within the model by an other, if the new pattern has the same or a compatible interface. Such an exchange might become necessary if the changes in the application domain exceed the built-in flexibility of the schema-pattern.

EXISTENCE OF VARIANTS

To limit the amount of modelling constructs the set of possible variants of each basic construct should be kept small. The more flexibility a pattern possesses, the less variants are needed. Variants should be used only for substantial differences form the original pattern, not for differences which could be achieved by using parameter. The description of variants only has to define the differences to the original patter res. the variant upon which it is based.

The set of variants should represent a semi-ordered set with the intention of upward compatibility which makes it possible to extend a model by exchanging a variant by a more powerful one.

CLEARLY DEFINED BUSINESS RULES

The schema-pattern must contained all the business rules which describe the integrity and consistency of the modelled part of the application domain. One part of those rules describes the embedding of the schema-pattern in the larger model, the other part describes internal conditions.

To keep this set of rules as flexible as possible, the rules should be modelled in data as much as possible. This way the rules can be handled and updated by the end users as well. The data models to represent this kind of rules constitute schema-patterns as well.

4.3. EXTENDED REQUIREMENTS

FUNCTIONALITY

If provided, the defined set of functions should exceed the basic set of 'create, update, delete, read' functions. It should include the necessary functions relating to the concept of the schema-pattern as well as those functions necessary to cover the life cycle model.

They have to be described using the generally defined formalism. The quality criteria are those which are normally be applied to function models.

STATE MODEL

The state-model should be sufficiently abstract to be applied to many different application models. It should not be too abstract to keep the cognitive distance small.

DESIGN AND IMPLEMENTATION

The design concepts and prefabricated implementations should be formulated as independent of the actual environment as possible. The limitations and rules describing the applicability have to be provided.

6 A definition is contained in [1].

4.4. Requirements Regarding the Environment

Library

A pattern library is needed, which contains all patterns. A set of requirements has to be fulfilled for each pattern to make the library as efficient as possible

Every schema-pattern has to be classified according to a hierarchical res. a faceted classification contained in the library catalogue. The possible classifications are contained in a thesaurus. Additionally entries in a keyword index are needed. Hypertext-like links between similar, related and substitutable patterns are a useful addition.

It has to be noted, that the formalism used for the definition of the patterns influences the choice of the formalism used to model the application domain, since it is obviously easier to use the same formalism for both purposes. In most cases it is inefficient to provide several data models in different formalisms for each schema pattern to support their application in different projects and application systems.

The responsibility for the library has to remain centralised.

User Support

To improve the use and maintenance of patterns and the data models based on them, it is necessary to maintain the connection between the pattern and all data model in which it is used in the (logically) central data dictionary.

With this reference it is possible to maintain the consistency of the whole modelling process and the resulting models, whenever the basic schema-patterns have to be changed. Secondly it will be easier to find the applicable variant of the used pattern or a completely new pattern, if the data model has to be changed because of changing requirements.

Consulting Services

Since the processes of creating, using and maintaining the library are quite complex care has to be taken to disseminate the necessary knowledge.

5. Conclusion

This paper presented an improvement of the process of data modelling by extending the set of applicable constructs and providing a library of reusable constructs as well as the necessary infrastructure for efficient modelling. The quality requirements for such patterns, the contents of this specification and a generally applicable way of recognising them were described and related to the process using the schema-patterns.

The presented process has the following advantages:

- Reduction of the effort to develop the data model.
- Increase of the quality of the model schema through improved overall consistency of the data model.
- Improvement of the cost of implementation since one implementation of a pattern can be used several times in projects.

The main disadvantages are:

- The effort needed to create an maintain the library.
- The extra cost to the project, to define patterns.
- The analyst may be mislead into using a pattern prematurely.
- The acceptance of the completed model might suffer from the 'not invented here' problem, since large parts of the model were conceived elsewhere.

The advantages of this approach outweigh its disadvantages especially for those organisations which perform much data modelling in different settings. The costs of building and maintaining the library should be regarded as an investment, the library itself as an asset.

Bibliography

[1] Balzert H.,
Die Entwicklung von Software-Systemen;
Bibliogr. Institut 1982.

[2] Bertram M.,
Aspekte der Qualitätssicherung von Unternehmensdatenmodellen; Proceedings 'Wirtschaftsinformatik '93', pp 230-242

[3] Castano, DeAntonellis, Zonta,
Classifying and Reusing Conceptual Schema;
Proceedings of the 11th ER-Conference 1992, pp.121-138.

[4] Coad P.,
Object Oriented Patterns; Communications of the acm Vol.35(9) 1992, pp.152-159.

[5] Hammer M., McLeod D.,
Database Description with SDM; acm TODS, Vol.6 No.3, (1981), pp.351-386.

[6] Krueger C.W.
Software Reuse. acm Computing Surveys; Vol.24 No.2 (1992), pp.131-184.

[7] Parnas D.L.
On the Criteria to be used in Decomposing Systems into Modules; Communications of the acm; Vol.15 No.12 (1972), pp. 1053-1058.