Data Modeling Patterns using Fully Communication Oriented Information Modeling (FCO-IM)

Fazat Nur Azizah, Guido Bakema

FN.Azizah@han.nl, Guido.Bakema@han.nl
Research and Competence Group Data Architectures & Metadata Management
Informatics and Communication Academy, HAN University of Applied Science
Beverweerdlaan 3, 6825AE Arnhem, The Netherlands.
Phone: +31-26-3658271. Fax: +31-26-3658126.

Abstract. Data modeling patterns is an emerging field of research in the data modeling area. Its aims are to create a body of knowledge to help understand data modeling problems better and to create better data models. Current data modeling patterns are generally discussed at the instance level (only applicable in a specific domain, e.g. a business situation) and with an Entity-Relationship Modeling (ERM) way of thinking. This paper discusses data modeling patterns using the expressive power of Fully Communication Oriented Information Modeling (FCO-IM), a Dutch fact oriented modeling (FOM) method. We also consider more abstract higher level data patterns – meta level patterns – and describe a few basic meta level data modeling patterns in brief as well as a meta level pattern in content versioning.

1 Introduction

The vast bulk of current work on patterns is carried out in the field of object oriented applications, but some notable work has been done in the field of data modeling as well [7, 12]. The importance of data modeling patterns is clear: although data modeling is a crucial part of the information systems development life cycle, data modeling is a rather expensive activity, and it is in general not easy to come up with a good data model. Patterns can help data modelers to do their job better and so reduce costs.

Most data modeling patterns are described at the instance level; such patterns are applicable only to one particular domain, for example a given business case. It is interesting to investigate patterns from the structure level point of view of data modeling itself, instead of related to the content.

Current work on data modeling patterns generally employs Entity Relationship Modeling (ERM) for presenting patterns, not only as the means of describing the proposed data models, but also for discussing the way of thinking behind them. In the fact oriented data modeling community however, patterns are hardly discussed yet, although Fact Oriented Modeling (FOM) techniques provide a promising approach for dealing with data modeling patterns, because of their different way of thinking.

2 Patterns

Generally speaking, people try to recognize patterns to understand the world better and to use these patterns to their advantage. The need of finding and codifying patterns arises in particular from the wish to reuse existing and proven solutions to particular recurring problems. This makes it an attractive subject, not only for researchers, but also for practitioners.

The goal of pattern finding is to create a body of knowledge that helps us to understand and to resolve recurring problems, by documenting insight and knowledge gained from problem solving experience and by putting these into a shared vocabulary that helps us to exchange these solutions and the lessons learned from them.

2.1 Definition

A widely accepted scientific definition reveals a pattern as *a proven solution to a problem in a context* [2]. This definition is derived from one given by Christopher Alexander[1]: “each pattern is a three-part rule, which expresses a relation between a certain *context*, a *problem*, and a *solution*.” [1]. Although others provide slightly different definitions, this one seems adequate.

So: a pattern is an instruction or a description of a solution to a (recurring) problem with its goals and constraints, which takes place in a certain context. But it does more than just *describe* a solution; it should also *explain* why the solution is adequate. For further discussions on this topic, see [2, 3].

2.2 Elements of Patterns

From the definition, the three basic elements of a pattern can already be seen: *context*, *problem*, *solution*. Several authors describe a few extra properties that should be present as well. For details on this subject, see [3]. Despite all differences in formats, the following essential elements are often mentioned [10]:

1. *Name*: a meaningful designation to refer to the pattern and the knowledge and structure it describes.
2. *Problem*: a statement that describes the intent of the pattern; i.e. the goals and objectives it wants to reach within the given context.
3. *Context*: the preconditions under which the problem and its solution seem to occur and for which the solution is desirable.
4. *Solution*: rules describing how to realize the desired outcome, often equivalent to giving instructions describing how to construct the necessary work products.
5. *Examples*: one or more sample applications which illustrate: a specific initial context; how the pattern is applied to it and transforms it, and the resulting context.

[1] Christopher Alexander, a physical architect, is known for his writings on patterns in urban planning and building architecture. His writings influenced people from other fields, including software engineers.

3 Data Modeling Patterns

Ever since the data modeling area was established by its founding fathers, such as Edward Codd, Peter Chen, Sjir Nijssen and others, data modeling has played a crucial role in the development life cycle of software and information systems. Data modeling methods, such as ERM and FOM, were invented to master the vast amount of data owned by organizations and to derive physical data models to store, retrieve and manipulate these data efficiently. With the increasing number of data models created to solve various types of problems, data modelers started to think about patterns in these data models to reuse proven solutions.

The work of David Hay [7] is appreciated by researchers and practitioners in the software and information systems development area. He composed a collection of patterns of data models applicable in several business situations that is more or less accepted as a standard and used in practice. Other researchers refer to him for further study on patterns in data modeling. Other noteworthy work was done by Silverstone [12], though this is more a collection of data models, rather than a collection of data model patterns. In [8], Ralph Kimball provides a useful and often consulted set of (dimensional) data models.

3.1 Levels of Data Modeling Patterns

The data modeling patterns mentioned above contain specific aspects that often occur in business applications, such as organization structure, product, manufacturing, and contract. The patterns are found in solutions to data modeling problems in businesses. Therefore, these patterns are only suitable for assisting the data modeling process in these business situations. If someone wants to create a data model for another type of application, then the patterns will probably not be useful anymore, or perhaps only a small portion of the patterns can be used for the new situation.

From the perspective of abstraction, such domain specific data modeling patterns are at the lowest abstraction level. They have a particular domain of application, certain particular terminologies, and certain particular semantics of objects and relationships. As a pattern, they are at the instance level.

However, when the structure of these instance level patterns is observed more carefully, it turns out that a higher abstraction level is sometimes present as well. Such a kind of pattern shows up in several instance level patterns. For example: the model of the hierarchy of *Geographic Location* in David Hay's *People and Organizations* pattern is essentially the same as the model of the hierarchy of *Activity* in his *Activities* pattern [7], and it is even similar to the way Martin Fowler modeled his *Organization Structure* object oriented pattern [6]. All these hierarchies have a structure in common, which is simply a cyclic binary homogeneous relationship (i.e: the same entity type occurs at both ends of a binary relationship type, a structure that is sometimes incorrectly called recursive).

Such an abstract pattern as this hierarchy showing up three times in different instance level patterns, can be used to build other instance level patterns. Therefore these abstract patterns are related to a *class* of problems rather than to one typical content problem. Hence, they belong to a higher abstraction level than the instance level patterns. Such patterns are at the meta level.

Above this meta level, there might be an even higher level of abstraction with respect to patterns: the meta meta level patterns. These patterns capture the entire idea of patterns and provide knowledge about patterns themselves. This constitutes the most interesting and also the most difficult topic of studying patterns, which we intend to study further in the near future. See figure 1.

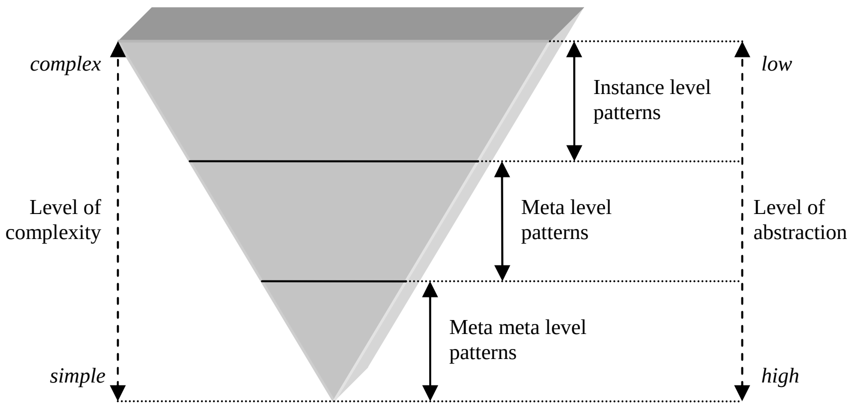

Fig. 1. The levels of patterns.

Figure 1 also depicts the level of complexity of data model patterns, which goes in the opposite direction of the level of abstraction. This means that the higher the level of abstraction is, the simpler the pattern becomes, and vice versa. This conveys the principle of pattern generativity [3], which succinctly states: a complexer pattern is built from simpler patterns.

In the following sections of this article, the full emphasis is on meta level patterns.

3.2 Data Modeling Patterns in FOM

The primary focus of defining patterns is not so much on technology, but on higher level descriptions. Choosing an appropriate methodology is not about creating diagrams that express how the data is represented or stored. Especially with respect to data modeling patterns, the methodology should not only provide a way of modeling, but primarily a way of thinking in viewing a problem and a way of communicating ideas for problem solving among analysts and domain experts.

The ERM approach, the most popular approach to data modeling and describing data modeling patterns, considers the Universe of Discourse (UoD) as a collection of entities having relationships which each other. In a similar fashion, an Object Oriented Modeling (OOM) approach[2] sees the UoD as a collection of objects interacting

[2] Although formally the Object Oriented Modeling approach is not a data modeling technique, some parts of it are commonly used for data modeling; e.g. class diagrams in UML.

with each other. FOM approaches consider the UoD to be described by a set of declarative facts. These different approaches provide different ways of looking at patterns.

In the FOM community, not much discussion has taken place till now about data modeling patterns. A FOM approach offers several valuable things for finding and describing patterns. Firstly, it provides a more detailed way, even at an atomic level, of looking at things, because it analyzes *elementary* facts instead of grouped facts in the form of entities or objects. This provides an advantage in examining the problem from the most basic level. Secondly, a FOM approach has a higher level of conceptuality compared with the ERM or OOM approaches. It is closer to the user world by providing a better way of communication. Finally, some FOM methods, such as FCO-IM, offer the possibility of using powerful modeling constructs such as generalization and recursive identification [5], providing a broader opportunity for pattern application. In [4] a complex example of recursive identification is discussed. FCO-IM consistently presents all of the power of fact oriented data modeling mentioned above.

3.3 Quality of Data Modeling Patterns

Patterns in data modeling are found from perhaps hundreds of different solutions to a particular problem, designed by different data modelers. It is possible that all these solutions correctly capture the user requirements. Nevertheless, not all of these solutions have the quality to be considered as good data models. So, it is desirable to be able to select a qualitatively good solution to a particular problem given a certain context. After all, the goal of data modeling patterns is to help information modelers to create better data models. So, we have to define criteria for good data modeling patterns.

Some experts list which qualities should be present in a good pattern [3, 9]. However, it is Christopher Alexander who has described a stunning definition of quality that he called the ‘Quality Without A Name’ (QWAN). This quality imparts incommunicable beauty and immeasurable value to a structure. Alexander proposes the existence of an objective quality of aesthetic beauty that is universally recognizable [3].

Nevertheless it seems desirable to define more precisely which qualities a data modeling pattern should possess. No author has written about quality of data modeling patterns[3]. Not much has been written about the quality of data models either, apart from models (using ERM) produced by students in simple and contrived exercises.

Still, there are some useful works. Graeme Simsion provides ten quality factors for data models [13]. Some of these quality factors may be applicable to patterns as well; for example *non-redundancy*[4] (which is more or less automatically assured when a FOM method, like FCO-IM, is used). Another quality factor might be the key factor for determining the QWAN for data modeling patterns: *elegance*. Elegance is a hard concept to pin down, but most data modelers will admit to experience an “aha!” moment, when a data model seems to be particularly neat. The concept of elegance is associated with *consistency*, *conciseness* and *comprehensiveness*. A few criteria that may be useful for measuring the elegance of a data model:

[3] In [5] David Hay speaks about drawing up a good data model, but he does not mention anything about the quality of data model patterns itself.

[4] FOM methods, when employed properly, guarantee 5NF data models.

1. *Syntactic, semantic, and positional standards*
   A standard style of defining the data model leads to a more consistent data model, by consequently using standards of syntax, semantics, and position.
2. *Size of a data model*
   In general: the smaller the data model for a given UoD, the more concise it is. In FCO-IM terms: the fewer elementary fact types (either nominalized or not nominalized), the better the model; and for each fact type: the fewer roles , the better.
3. *Abstraction level of a data model*
   Conciseness of data models can be achieved by putting more general (higher level of abstraction) structures in the data model. General structures can also improve the level of comprehensiveness. More abstract data models might also be reuseable in broader contexts. But introducing abstraction must be done in a controlled way, because overabstraction could lead to the contrary: loss of comprehensiveness, apart from difficulties with respect to the implementation,.
4. *Completeness of a data model*
   The data model must cover all aspects of the UoD that should be modeled. This should not be sacrificed for the sake of conciseness.

In the end, the pursuit of an equilibrium among these aspects is the most important factor for defining the elegance of a data model as well as a data model pattern.

4 Meta Level Data Modeling Patterns

Finding a good pattern is a difficult task. However, the principle of generativity of patterns gives us a clue on how to find patterns. This principle states that patterns are built from building blocks that are also patterns; a procedure which ends with the most basic building blocks. In the case of meta level data modeling patterns these basic building blocks can be determined from the smallest recurring structures that can be used to build larger building blocks. From the level of complexity of the pattern point of view, this provides a bottom up approach in finding larger meta level data modeling patterns.

However, since meta level patterns can be found from the study of instance level patterns, a top down approach can also be employed by studying several instance level patterns, or even by studying cases in which no pattern has been defined before, that possess similar structures. These similar structures can be abstracted into the meta level patterns.

4.1 Basic Meta Level Data Modeling Patterns

The basic meta level data modeling patterns provide the highest level of abstraction of the meta level data modeling patterns. From the point of view of complexity, they could be the simplest ones however. They provide the basic pattern structures that can be employed by other meta and instance level patterns.

There are several basic constructs in FCO-IM that can serve as basic meta level patterns. Some of them are:

1. Semantically Equivalent Transformation Patterns (e.g. object to fact type transformation, nominalization, and specialization/subtyping).
2. Generalization Patterns (e.g. generalization with synonymy, generalization with homonymy, and recursive identification).

The full description of all these constructs can be found in [5], including denormalization, handling exception cases, etc.

4.2 A Pattern in Content Versioning

This section discusses a pattern in cases with content versioning. It is an example of a meta level pattern derived from instance level patterns and cases. The meta level case is about the management of the content of documents that undergo versioning, i.e. the history of revisions on the document must be recorded. This can occur in many fields, from legislation to manufacturing industry. Consequently, it is a useful pattern. Two cases have been studied:

Legislation[5]**.** This case reflects the management of revisions of laws. The case – coming from the principle that every citizen must have access to laws – asks for a repository for laws and their successive versions.

A volume of laws consists of several parts. These parts contain chapters and chapters contain sections. Each section consists of articles. Each of these elements can be subjected to revision. For example, revision may occur at the article level, in which case the version of the section (the next higher level) does not change, just the version of the article in question. But it can also occur that a whole law changes, which leads to a different version of the entire law.

Each revision must be in one of the following states: draft, review, approved, or historical. The content of each revision is stored in an XML file.

Bill of Materials[6]**.** The bill of materials case is related to the planning and assembling of products in manufacturing processes. Each product comprises several components (which are also products). When a product is ordered, it means that its components must be ordered. Each of these components may consist of several other components which must be ordered as well then.

Revision may take place at every level of hierarchy. This revision may be in one of at least two statuses: current or historical. For each level of the bill of materials, the quantity ordered is recorded, and some other attributes.

Both cases deal with versioning / revision of objects. Each revision may have several attributes; an attribute in common is the state of the revision. Both cases deal with compound objects, i.e. objects that are composed of one or more elements. In the legislation case, it is the law document that possesses the hierarchical structure of a docu-

[5] This case is a simplified version of a case used in an Enterprise Content Management course in the study program Master of Information Systems Development of HAN University.

[6] For a more detailed description of the bill of materials case, see for example http://www.ciras.iastate.edu/publications/CIRASNews/fall97/bom.html.

ment; in the bill of materials case, it is the bill of materials that may consist of several levels of components.

The following description can be made for the pattern found in these content versioning cases:

1. *Name*: Content Versioning Pattern
2. *Problem*: Create a data model for a structure that deals with an object that may have several versions due to revision acts. The object may be a compound object i.e. an object that is composed of one or more elements that may be compound as well. In the case of a compound object, each of the elements may be subjected to revision.
3. *Context*: No particular context is required.
4. *Solution*: Data modelers can think of several ways of modeling this. The following proposed solution is (with some adaptation and simplification) based on a data model that was created in practice for the legislation case. We chose this data model because of its simplicity in solving a relatively complex problem. It models the compound objects as a single object type (even though it requires a new way of identifying the elements of the compound object type), showing a high level of abstraction in the understanding of the compound object.

   The general solution is shown in figure 2. Revision is an object type, i.e. a nominalized fact type, with two roles; one role played by 'revision code', the other played by an object type Element, which represents the compoundness of objects. Each element has a level that indicates the position of the element within the hierarchy of the compound object. For example: in the law document case it can be article or section or chapter, whereas in the bill of materials case a level is usually indicated by a positive integer (1, 2, 3, etc.). The hierarchy itself is modeled by defining the parent of an element (which is itself also an element). For example: in the law document, element 11 which is an article can have element 10 as its parent, which is a section.

   Element and Revision can have their own attributes, which are depicted as extra fact types played by the respective object types. These attributes are defined according to the situation in which the pattern is implemented.
5. *Example*: The application of the pattern for the legislation problem is shown in figure 3. To cut the discussion short, only the data model solution is given. In this data model, the general construct is the same as the general solution of the pattern. The differences lie in the attribute fact types played by Revision and Element.

5 Conclusion and Future Work

Although still in a very preliminary stage, the result of this study gives a promising prospect of finding more meta level data modeling patterns. However, finding and writing a pattern is not an easy task and feedback from other parties is required.

The difficulty in defining meta level patterns lies in two aspects:

- understanding deeply the underlying structure;
- choosing the best solution out of all existing proven solutions.

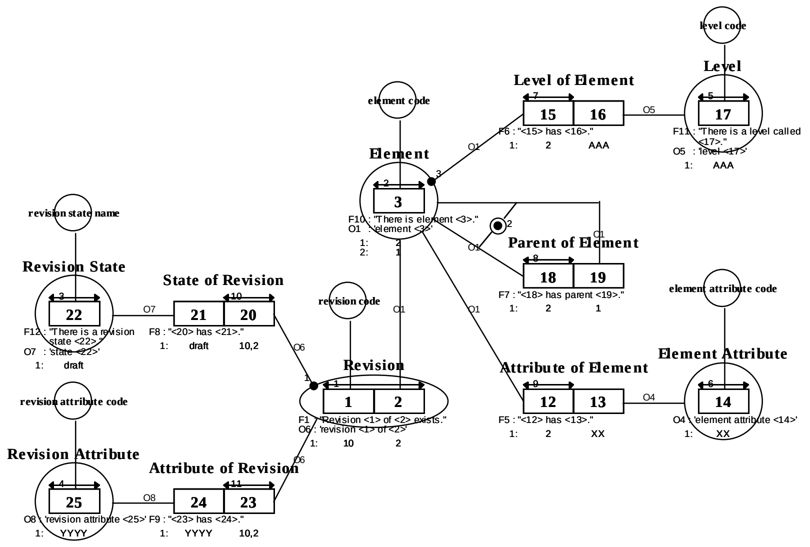

Fig. 2. General FCO-IM data model solution for the Content Versioning Pattern

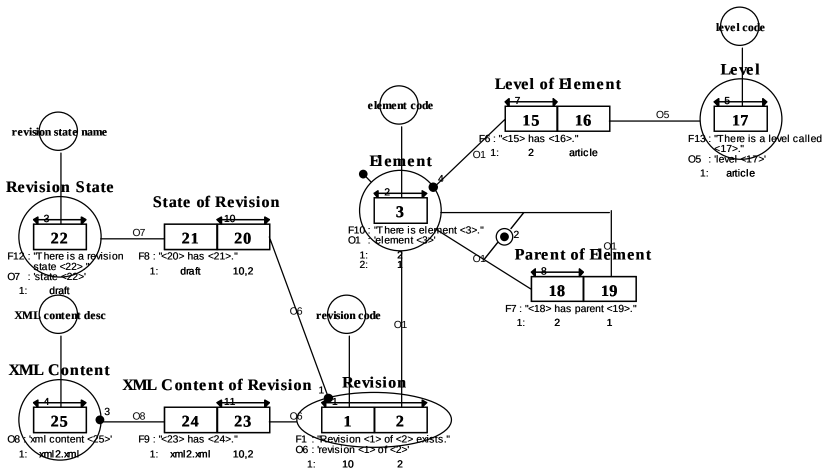

Fig. 3. Data model for legislation problem based on the Content Versioning Pattern

Even in the simple case of the content versioning pattern, these difficulties are present. Therefore, it is important to study quality criteria for data modeling patterns further and to find proper criteria for measuring the quality of patterns. Nevertheless, in cases that have the same structure as the content versioning pattern, this solution can be reused, with a little adaptation perhaps.

The use of a FOM method, such as FCO-IM, provides a more detailed and yet more conceptual view: the real structure of the solution of the pattern can be captured properly. It is not very difficult to adapt the described pattern to a new situation that contains the same type of structure. FCO-IM provides also the advantage of describing some basic meta level patterns, such as generalization and recursive identification, that prove to be powerful meta level patterns [4, 11].

We would like to underline the concept of levels of patterns. With more data models created for a wide range of areas, it is essential to acquire a broader view on patterns. This does not mean that instance level patterns are not useful, but meta level patterns will provide a more consistent way for describing patterns. This will help to create a comprehensive body of knowledge of data modeling patterns.

Further study is needed to investigate the role of meta meta level patterns, based on the hypothesis that meta meta level patterns will provide a powerful way for defining the concept of pattern description.

Acknowledgement. Jan Pieter Zwart is gratefully acknowledged for reviewing the phrasing in this paper.

References

1 Alexander, C.: *The Timeless Way of Building*, Oxford University Press, USA, 1979.
2 *A Pattern Definition*, http://hillside.net/patternsdefinition.html, downloaded at 19/04/2006.
3 Appleton, B: *Pattern and Software: Essential Concepts and Terminology*, http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html, downloaded at 19/04/2006.
4 Azizah, F.N: *A Case Study of Recursive Data Modeling*, Libyan First International Symposium on Information Systems Modeling and Development, working papers, Tripoli, Libya, 2006.
5 Bakema, Guido; Zwart, Jan Pieter; Lek, Harm van der: *Fully Communication Oriented Information Modeling (FCO-IM)*, 2002. The book can be downloaded for free from http://www.casetalk.com/php/index.php?FCO-IM%20English%20Book.
6 Fowler, M.: *Patterns in Enterprise Software*, http://www.martinfowler.com/articles/enterprisePatterns.html, downloaded 19/04/2006.
7 Hay, D.C.: *Data Model Patterns*, Dorset House Publishing, New York, 1996.
8 Kimball, R., Ross, M.: *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*, John Wiley & Sons, 2nd Edition 2002.
9 Lea, D: Christopher Alexander: *An Introduction for Object-Oriented Designers*, http://g.oswego.edu/dl/ca/ca/ca.html, downloaded at 19/04/2006.
10 Lehti, L., Ruokonen, A.: *Foundation of the patterns*, http://www.cs.tut.fi/~kk/webstuff/Foundationofpatterns.pdf, downloaded at 06/07/2006.
11 van der Lek, H.: *On the structure of an Information Grammar* , NIAM-ISDM 1993 Conference, Working Papers, Utrecht (1993).
12 Silverstone, L.: *The Data Model Resource Book: Revised Edition*, Volume 1 and 2, John Wiley & Sons, Inc., 2001.
13 Simsion, G: *Better Data Models – Today, Understanding Data Model Quality*, http://www.tdan.com/i034ht01.htm, downloaded at 19/04/2006.