Meta-level architectures : a conceptual tool or a banana skin ?

Vincent Ribaud, Alain Plantec, Philippe Saliou

Soumis à Reflection'99

Meta-level architectures : a conceptual tool or a banana skin ?

V. Ribaud[1], A. Plantec[2], and P. Saliou[2]

[1] LIBr Faculté des Sciences, BP 809, 29285 Brest Cedex, France
ribaud@univ-brest.fr
[2] Syseca, 34 quai de la Douane, 29200 Brest, France
{plantec,saliou}@SYSECA.thomson.fr

Abstract. The present paper presents the different modelisation levels. The approach is discussed following the IRDS framework. The cornerstone of the IRDS framework is the concept of four data levels and the associated three "level pairs". Omitting this useful concept leads to misunderstanding and misemployment of the meta paradigm. Moreover, the unique concept provided by the "instance of" relationship should be nuanced. Three applications using a meta-level architecture are described and the pros and cons of the architecture are discussed.

1 Introduction

Data management uses a lot of formal models (relational, object-oriented, etc). The programming environments often offer the possibility to describe all the used models. This description is structured in a dictionary or meta-model, which, following Codd's idea [Dat90] is usually implemented with the same programming environment constructions as the models themselves. Applying this process to the meta-model itself leads to a meta-level architecture.

Users understand (somewhat intuitively) these different meta-levels and are able to use this meta-information directly or with the appropriate tools. Three operational applications using meta-level techniques are presented, and pros and cons are discussed.

In early 1990, IRDS (*Information Resource Dictionary System*) proposed a framework of four levels and three associated "level pairs". At the moment, the presentation of meta-theory focuses on levels rather on level pairs. Moreover, no differences are made between the "level pairs"; the UML adopts a unique "instance of" relationship for describing the link between data belonging to two adjacent levels. This statement tends to uniformize problems of a different nature and entails proposing inappropriate solutions in some cases.

The uniform point of view of the UML should be nuanced. It leads to misunderstanding and misemployment of the meta paradigm, this shall be illustrated with some problems encountered in the three applications presented.

This article is organized as follows. In section 2, we present the four-level decomposition of information. In section 3 we depict three operational applications, using these meta-levels techniques. Section 4 discusses about the pros and cons induced by the use of meta-levels. We will finish with a conclusion.

2 A four-level architecture

2.1 Definitions

Meta The prefix *meta* in front of a word X express the fact that the meta-X applies to Xs [Pit90].

Model [OMG97] : A semantically closed abstraction of a system.

Metamodel [OMG97] : A model that defines the language used to express a model.

Meta-metamodel [OMG97] : A model that defines the language used to express a metamodel. The relationship between a meta-metamodel and a metamodel is analogous to the relationship between a metamodel and a model.

2.2 Examples of architecture

CDIF (*CASE Data Interchange Format*) [EIA94] is a proposal intended to facilitate the cooperation of Computer-Aided Software Engineering (CASE) tools and the exchange of models between the vendor's tools.

IRDS (*Information Resource Dictionary System*) [ISO90] is an ISO-standard intended to provide a common basis for the development of Information Resource Dictionaries.

Both are used to manage metadata and data. Both define a four-level architecture : data, model, metamodel, meta-metamodel; and use an *entity-attribute-relationship* (EAR or simply ER) model (IRDS) or an *object-relationship* (OR) model (CDIF) to describe each level.

Meta-metamodel The first level is the "*meta-metamodel*" level (called IRD Definition Schema Level in IRDS). The purpose of this level is to prescribe the types of object about which data may be recorded on the second level.

This level is usually fixed and frozen : no entities (classes) or relationships can be added at this level.

Metamodel The second level of the architecture is the "*metamodel*" level (called IRD Definition Level in IRDS). The purpose of this level is to contain methodological definitions about the methods and processes used.

Model The third level is the "*model*" level (called IRD Level in IRDS). The purpose of this level is to contain model definitions or type definitions (in the sense of a programmatic language).

Data The fourth level (called Application Level in IRDS) is the level on which instances of data are recorded, retrieved and updated.

3 Three data management applications

Three operational applications that we built at Syseca Inc. are briefly presented. The attention is focused on meta-level use rather than on overall functionalities. The pros and cons of meta-level techniques are discussed later in section 4.3. In this section, we concentrate on the user's requirements and the resulting functionalities.

3.1 Technical information management

IFREMER (*Institut Francais pour la Recherche et l'Exploitation de la MER*) is a governemental research center, covering most of the domains of the sea. Each research team constitutes relational databases of relevant data. The flat model of SQL does not fit the complex modelization needs of researchers. Thus, we designed and built a tool for technical information management. It was composed of :

- An object-oriented modelization language and its associated compiler. The language allows the definition of information models which are implemented in a relational database management system.
- A human-interface makes it possible for the user to query the **data** of the object models; this is accomplished by automatically generating and executing SQL statements.

From a usual standpoint, the object-oriented model represents knowledge in the domain and is used to formulate hypothesis (queries), which is a typical meta-level activity. Then data are retrieved according to the queries and are used to validate the hypothesis. In other words, the design of the tool uses two levels :

- a model-level handling classes of the model,
- a value-level allowing the selection and the filtering of instances belonging to the classes handled at the top-level.

But from user's point of view, the data itself are more important to formulate the queries than the models. The user needs to see values and links between data dynamically in order to refine his/her comprehension.

Thus, the keypoint is that the support of reasoning for the user is not only the model but the data too. So the tool human interface unifies the levels: there is no difference between adding/removing a class and filtering instances.

The reasoning navigates between levels : once classes are selected, the user wishes to visualize **all** the data concerned, to filter data according to certain criteria, to examine the results that could eventually conduct the tool to discard a class from the model if there are no more result data belonging to the class.

3.2 Hydrographic Data Base

The Hydrographic and Oceanic Departement of French Navy (*Service Hydrographique et Oceanographique de la Marine*), called SHOM, is in charge of the production of nautical charts.

The Hydrographic Services used a data representation and exchange standard called DX90. Data are transferred intertwined with the model describing them. It means that if thousand instances of the same entity are transferred, the name of the entity and the name of attributes will appear thousand times in the exchange file. So, cartographs were not able to distinguish data levels.

But, cartographs uses another standard called S57. The standard is a model of the cartographic process and normalizes the name and the structure of entities used in the domain, e.g. the entity LITHOU (standing for "lighthouse" is described with the attribute GEO (geographic position). The composition of entities is also an important feature. Cartographic entities (e.g. a lighthouse) are complex entities, formed by the composition of other entities (e.g. an infrastructure, a light, a fog-horn, etc). The S57 model is described using a meta-model; so cartographs are "metamodel" end-users.

During the conception and the implementation of a database intended to support the chart production process, the designers partly reproduced the structure of a DX90 file, and partly used meta-level techniques.

This leads to a non-conventional design :
all keys of *all* instances of *all* entities were recorded in a single table called **OBJETS**
all attribute values of *all* instances of *all* entities were recorded in a single, enormous table called **ATTRIBUT_OBJETS**, each row of this table being linked to the appropriate row of the **OBJETS** table
and a huge **OBJ_OBJ** table allowing **each** object of **any** type to be linked to **any other** object).

Hence, the structure of the entities (belonging to the metamodel-level) was lost. This structure was re-created with the help of a metabase :
CLASSE_OBJETS describing each entity,
ATTRIBUTS describing the name and the type of each attribute,
...

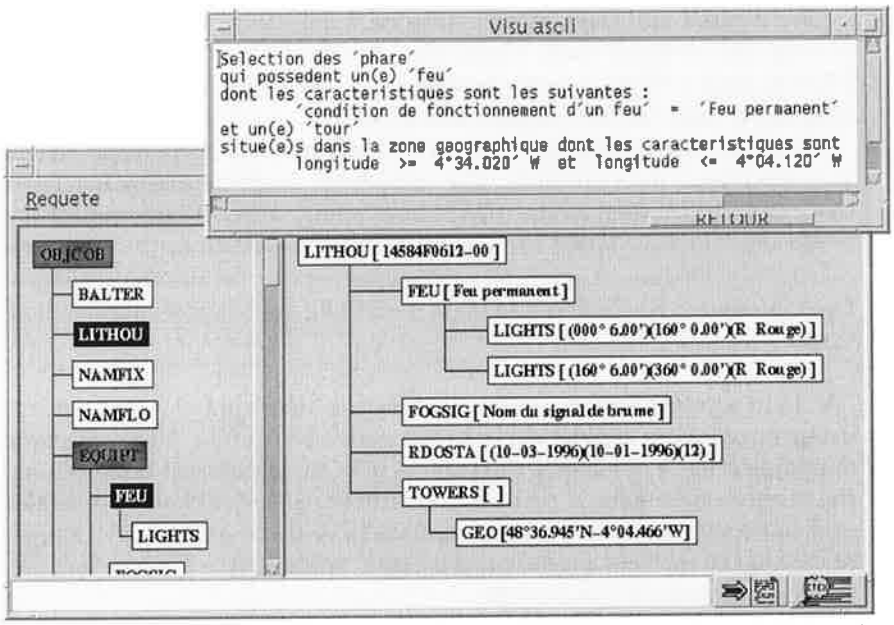
All these features contribute to the overall use of meta-model in the application.

We implemented a data management tool for this database. Thanks to the database (and metabase) structure, the tool is generic and dynamic in its very nature, consulting the meta-model in order to query data.

This tool encompasses all data management functions and can be applied to many domains :

- A data editor offers all elementary actions for inserting, updating, deleting, decomposing, renaming and duplicating the data.

- An information center allows the user to graphically elaborate queries; this is accomplished through a graphical representation of the structure of data models in the left window and the disposition of values in the right window according to the left-hand structure (see figure 3.2), the SQL is automatically generated and an interpretation in pseudo-french is given.
- A report generator supports data formatting in order to be visualized or printed.



A lighthouse is composed with permanent lights (LIGHTS), a fog signal (FOGSIG), an infrastructure (TOWERS), etc. The top window is the interpretation in pseudo-french of the query.

3.3 The STEP approach

STEP (ISO 10303) is an international standard for the computer interpretable representation and exchange of product data [ISO94b]. This standard provides a modelling language, *EXPRESS* (ISO 10303-11) [ISO94c], a neutral data encoding (ISO 10303-21) [ISO94d] and an application program interface to data called Standard Data Access Interface (*SDAI*, ISO 10303-22) [ISO94a]. The data are kept in a storage facility (files, data bases, etc) called a repository.

The functionalities of a data management application can be drawn from the goals of the *SDAI*:

- to access and manipulate data which are described using a conceptual language so that access to a database happens through a conceptual schema, not a physical schema,
- to allow access to multiple data repositories by a single application at the same time,
- to allow commit and rollback on a set of data operations,
- to allow access to the dictionary definitions of all data elements that can be manipulated by an application process, and
- to allow the validation of the constraints expressed on the data.

We devised and built several software applications using STEP in order to manage and exchange data. Classically, data models are first designed using EXPRESS, then a SDAI (usually formed by classes in an object oriented language binding) is used for the management of data.

A SDAI can be implemented as an interpretor of EXPRESS schemata or as a specialized data interface. The interpretor implementation is referred to in the standard [ISO94a] as the SDAI late binding. A SDAI late binding is generic in nature. The specialized implementation is referred to in the standard as the SDAI early binding. A particular early binding can be automatically generated from the source EXPRESS schemata describing application data handled by the target.

We built several SDAI generators (generating SDAIs in C, C++, Java, Smalltalk) using an generator builder environment called EUGENE. This environment was developped for A. Plantec's PhD and is now an operational tool used at Syseca Inc. This environment is presented in [PR96], [PR98], [Pla99]. An SDAI generator built with EUGENE uses meta-data in order to generate the target SDAI. Meta-data come from a source specification analysis (the models) and are stored in different kinds of meta-model.

Thanks to the SDAI generating process, the SDAI offers some introspective capabilities, for example the knowledge of the type of an attribute. Moreover meta-levels are not clearly separated, e.g. some of the generated classes allows the user to browse through all the instances of an entity, or an instance can be directly linked to some meta-information, such as the name of its class.

4 All is not meta on the western front

4.1 Level versus level pair

Within IRDS, a *processor* is defined as an abstract conceptualization of an executable piece of code and a *service* is defined as a capability provided by a processor to other processors.

IRDS generalizes the concept of "types" and "instances" in the following way. The definition of a "type", such as an EMPLOYEE, creates an open-ended data container. When data about a specific EMPLOYEE needs to be recorded or accessed, it is necessary to refer to the type of this data. The IRDS concept

of data levels is an extension of this basic type and instance concept which one can regard as having two levels and one level pair (which are in fact the bottom two of the four previously described) [ISO90].

IRDS services may be thought of as operating on a level pair consisting of two adjacent data levels. The subset of the data at the upper level which defines types for the level below, is termed the schema for the lower level.

The first level pair [ISO90] defines IRD Definition Services (*services provided at the IRD Definition level*) as operating on the IRD Definition, and these services operate by reference to the IRD Definition Schema (*on the IRD Definition Schema Level*).

This means that in a four-level architecture, the highest services furnished operate on the second level (define and refine the metamodel level) by reference to the first level (the meta-metamodel level). As an example of the use of this first pair (called IRD Definition Level Pair in IRDS), a dictionary administrator will define the types of data that subsequently may be used within the second pair by an application designer (the administrator defines the modelling paradigm).

The second level pair The second pair (called IRD Level Pair in IRDS), on parallel lines with the first pair, operates on the third level (define and refine models) by reference to the second level (the metamodel or dictionary level). When the designer of an application is evolving and documenting the design of a database, he/she will use the services provided on this second pair.

Semantic We agree with the definition of a *Meta-metamodel* in [OMG97] : The relationship between a meta-metamodel and a metamodel is analogous to the relationship between a metamodel and a model.

IRDS follows the same meaning because the same data modelling facility is used for the first and second pair.

The third level pair Like the two other pairs, the third pair (called Application Level Pair in IRDS, which denotes the fact that we are working at the usual application level) operates on the fourth level (insert and delete data) by reference to the third level (the model).

Semantic The semantic of the third level pair is not *exactly* the same as other pairs (this point shall be discussed in the next section). We disagree with the fact that the relationship between a metamodel and a model is analogous to the relationship between a model and an instance.

Note that no functionality is prescribed for the third level pair in IRDS, because data are out of the scope of IRDS (intended to document an enterprise's information resources). Hence, IRDS is not explicit on this problem, but it is only indicated that one or more data modelling facilities can be used for the third pair, potentially differently from the two other pairs.

4.2 Instanciation versus meta-information

The metalanguage, the language used at the upper level to describe the lower level, is generally the same as the language used at the lower level. The benefits are uniformity and re-usability of tools used to manipulate a level.

We disagree with some point of view which will uniformize the different level pairing in a unique association type-occurrence or class-instance. Hence, we disagree with UML point of view which provides a unique "instance of" relationship, as between meta-metamodel and metamodel, metamodel and model and between classes and objects. We think that at least two different kind of level pairs exist, and each of them bears a different semantic.

Uniformizing the meta-relationship with the instanciation relationship is certainly convenient from an implementation point of view. Successful environments like Smalltalk benefit from this point of view (an object is an instance of its class just as a class is an instance of its metaclass). This may be the goal aimed at OMG by providing a unique "instance of" relationship.

In class-based object oriented language, the class is a factory for objects and instanciating a class yields new objects. A meta-object (in the UML sense : generic terms for all meta-entities in a metamodelling language) is not a factory for objects, but could be seen from two perspectives :

- A meta-object holds the definition concerning objects, and in a sense contains all the possible objects. This perspective is close to (but not equivalent to) the "factory" concept and maybe explains the possible confusion between meta and instanciation.
- Each object relates to its own meta-object, hence there is no possible factorisation. Obviously, a meta-object has a type and shares characteristics with other meta-objects (related to objects which themselves share characteristics like belonging to the same class and so on).

4.3 Pros and cons in meta-level applications

Technical Information Management The tool described in 3.1 enables end-users to define models, working at the second level pair. Queries can be specified at the same level, as usual. We are in the normal situation of a CASE tool, which generates a SQL code from meta-data, at the second level pair; this code being executed at the third level pair.

The main requirement for the tool was a user interface, allowing the user to see the resulting data and the *corresponding* model. The user did not distinguish between a search criteria, such as *some value* $>$ *10* and a join operation, such as *the entities linked to another entity*, even this appears as totally different in the software implementation.

This requirement was difficult to comprehend from our computer designer standpoint. It might be one of the motivations of Query-By-Example designers. QBE uses a single paradigm -in other words, values- to define different operations, such as search criteria and join.

So, we designed and built the tool as required. Due to the meta-level structure of Oracle and to the nature of SQL, we were unable to process dynamically all the possible queries. So we used an artifice :
at the moment the user-defined model was compiled to verify the syntax and to feed the meta-data, we generated views depicting **all** possible paths between entities belonging to the model.

This leads to a strong limitation to the models; no circular paths were permitted between entities. The limitation guaranteed the existence of a unique path between two different entities of the model, making it possible for us to generate all existing paths. The end-user does not complain about this fact, but we suspect that little use is made of the tool, or that it is only used with small models which are unaffected by this limit.

Hydrographic Data Base The database presented in 3.2 was a non-sense from a database designer's point of view. As subcontractors, we had no choice in the type of design, hence the necessity for us to build a tool able to manage this kind of structure. The tools generates queries mixing meta-model and model levels.

"Why did the initial designers choose this solution ?"

They (rightly) felt that they needed some meta-features, so they studied Oracle metabase (operating at the second level). The services of this level were similar to the services they needed, so they reproduced the *implementation* of the second level for the third level.

A more suitable solution is to implement differently the third level from the other two, by generating complex SQL queries from the meta-data rather than using directly SQL mixing data and meta-data. Nevertheless the choosen solution corresponded to its role, except for performance considerations.

STEP An application using STEP is equipped with a SDAI in order to manipulate its own data (described by its own model). During the generation process of the SDAI, the model is parsed and is stored in *EXPRESS* meta-schemata and then manipulated through a *SDAI*. Thus, it makes it possible for us to incorporate the meta-schemata into the final application, enabling a meta-level architecture.

The SDAI works at two level pairs. For performance considerations, a specialized SDAI is more suitable to deal with data, while a generic SDAI is more flexible to manage metadata.

5 Conclusion

After several years building meta-level applications, we recognize that we do not have a firm position faced with the problems encountered. We often slipped on the meta-paradigm : confusing levels or applying inappropriate techniques to

one level while suitable to another level. The "level pair" concept helped us to avoid these banana skins.

Even if a meta-level architecture is used, the end-user does not really need to apprehend this level-notion, if the application is able to hide this architecture in a practical interface.

The uniform "instance of" concept should be nuanced. Managing a large amount of data on the third pair is an implementation problem and requires dedicated techniques.

References

[Dat90] C. J. Date. *An Introduction to Database Systems, vol. 1.* Addison Wesley, Reading MA, 1990.

[EIA94] EIA. *CDIF - Framework for Modeling and Extensibility, Interim Standard,* 1994.

[ISO90] ISO/IEC 10027. *Information technology - Information resource Dictionary System (IRDS) framework,* 1990.

[ISO94a] ISO TC184/SC4 CD 10303-22. *Part 22: Standard Data Access Interface,* 1994.

[ISO94b] ISO TC184/SC4 IS 10303-1. *STEP Part 1: Overview and fundamental principles,* 1994.

[ISO94c] ISO TC184/SC4 IS 10303-11. *Part 11: EXPRESS Language Reference Manual,* 1994.

[ISO94d] ISO TC184/SC4, IS 10303-21. *Part 21: Clear Text Encoding of the Exchange Structure,* 1994.

[OMG97] Object Management Group OMG. *UML Glossary,* 1997.

[Pit90] Jacques Pitrat. *Métaconnaissance.* Editions Hermès, 1990.

[Pla99] Alain Plantec. *Exploitation de la norme STEP pour la spécification et la mise en œuvre de générateurs de code.* PhD thesis, Université de Rennes I, 35065 Rennes cedex, France, 1999.

[PR96] A. Plantec and V. Ribaud. Data management : From EXPRESS schemata to user interface. In *ICCI'96 Proceedings.* Journal of Computing and Information, 1996.

[PR98] Alain Plantec and Vincent Ribaud. EUGENE : a STEP-based Framework to build Application Generators . *Australian Workshop on Constructing Software Engineering Tools, ASWEC'98,,* 1998.