Formalizing and Comparing Software Architectures of Federated Database Management Systems

Wilhelm Hasselbring

Infolab, Department of Information Management and Computer Science
Tilburg University, 5000 LE Tilburg, Netherlands
Tel.: ++31 +13-466.8080, Fax: ++31 +13-466.3069, E-mail: hasselbring@kub.nl

Abstract

Constructing a formal specification requires some effort that should be justified. A motivation for formalizing software architectures is to enable a precise analysis and comparison of specific architectures for evaluating and selecting appropriate architectural variations.

This contribution discusses a detailed investigation of schema architectures, which are central components in the software architecture of federated database management systems. It is shown how specific architectures can be compared to the reference architecture and to each other. To achieve this, we combine the semi-formal object-oriented modeling language UML with the formal object-oriented specification language Object-Z.

1. Introduction

Software architectures are usually described informally with diagrams using boxes, circles and lines together with accompanying prose. The prose explains the diagrams and provides some rationale for the chosen architecture. Such figures often give an intuitive picture of the system's construction, but different people may interpret the semantics of the components and their connections/interactions in different ways (due to the informality). Through formalization, software architectures become amenable to analysis and comparison with such-like architectures. This helps to evaluate architectures and to guide in the selection of architectural variations as solutions to specific problems.

Usually, formal specifications consist of interleaved passages of formal, mathematical text and informal prose explanation. We propose a three-level interleaving of formality in the specification:

1. Informal prose explanation (including examples).
2. Semi-formal object-oriented modeling (we use the UML for this purpose).
3. Rigorous formal specification (we use Object-Z for this purpose)

An important goal is to obtain a well-structured formal specification. Formal specifications are often criticized, because it is hard to understand them. To some extent, the problems are due to missing (visual) structure in the specification. With our approach, the object-oriented diagrams provide an overview of the formal specification, and a first level of (semi) formalization.

Software architectures hide many of the implementation details, which are not relevant to understanding the important differences among alternate architectures. A formal specification language like Object-Z is well suited for concentrating on the essential concerns and neglecting irrelevant details through abstraction. The graphical, semi-formal UML specification is employed for providing an overview of the specified architecture as well as the structure for the formal Object-Z specification itself.

2. Federated Database Management Systems as an Example Domain

A federated database system integrates heterogeneous, autonomous database systems, whereby both local applications and global applications accessing multiple component database systems are supported [9]. Such a federated database system is a complex *system of systems*, which requires a well designed organization at the software architecture level. A problem that federated database systems face, is the organization of schemas in a schema architecture.

For federated database systems, the traditional three-level database schema architecture [3] must be extended to support the dimensions of distribution, heterogeneity, and autonomy. The generally accepted reference architecture for schemas in federated database systems is presented in [9]. As reported in [2], this reference schema architecture is generally accepted as the basic structure in federated database systems or at least for comparison with other specific architectures. However, several modifications have been proposed, as shall be discussed below.

A reference architecture for those schemas is useful to clarify the various issues and choices within complex federated systems. It provides the framework in which to understand, categorize and compare different architectural options for developing specific systems. Reference architectures are the structures used to build systems in a *product line*.

3. The Semi-Formal UML Specification

The Unified Modeling Language (UML) is a graphical modeling language that has been standardized by the Object Management Group [1]. Figure 1 displays the first step towards formally specifying the reference schema architecture using the UML notation for class diagrams. In this model, some of the constraints and options for the architecture, which are informally discussed in [9], are defined by means of the *multiplicities* at the associations among classes and other notational means. The schema architectures of some specific systems (see Section 5) are also defined this way. With these models, some high-level comparison is already possible.

This UML diagram already specifies several details of the schema architecture, but still many details are missing in the graphical model. Several constraints cannot be specified graphically within this class diagram. It is necessary to specify them textually by means of informal prose and/or the UML Object Constraint Language [1]. The Object Constraint Language allows some Boolean expressions to be specified. However, the class diagram in Figure 1 is a *semi-formal* specification as the *semantics* of the UML notation has not been specified formally (only with the UML itself and accompanying informal prose [7]). Only the *syntax* of the graphical notation provides some formality. The use of the Object Constraint Language would not resolve that problem. Therefore, we take the second step to further formalize the schema architecture by means of a formal specification language.

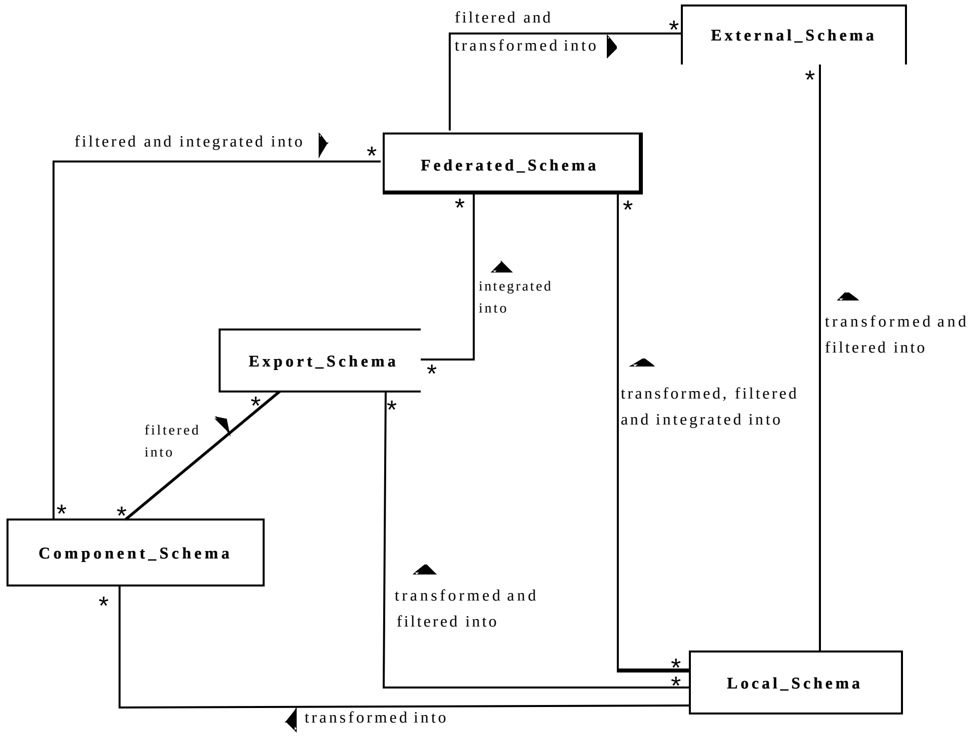
filtered and transformed into
External_Schema
filtered and integrated into
Federated_Schema
integrated into
transformed and filtered into
Export_Schema
transformed, filtered and integrated into
filtered into
Component_Schema
transformed and filtered into
Local_Schema
transformed into
*

Figure 1: Extract of the UML class diagram [1] for the 5-level reference schema architecture.

4. The Formal Object-Z Specification

Object-Z [4,10] is an extension of the formal specification language Z [11] to facilitate specification in an object-oriented style. It is a conservative extension in the sense that the existing syntax and semantics of Z are retained in Object-Z. Due to space restrictions, we cannot present details of the formal specification in this short position paper. Instead, we illustrate the overall structure of the formal specification below and the way specific architectures are compared.

5. Analysis and Comparison of some Specific Systems

We relate the concepts of the reference architecture to those realized in some specific federated database management systems. The purpose is to show how the reference architecture can be compared to the architectures of various federated database systems. Such a representation supports the task of studying and comparing these systems.

The question arises: *how* to specify the reference architecture such that it can be compared to the architectures of specific systems? Figure 2 illustrates an extract of the structure of our Object-Z specifications as a UML class diagram. The 'generic' components (displayed with dashed lines) of a schema architecture are sets of schemas and mappings among the schemas in these sets. Both the reference architecture and the architectures of most of the selected systems are specializations of the class Schema_Mapping. For instance, The IRO-DB [5] schema architecture is very similar to the reference architecture. Some schema types have different names in this system and external schemas are left out. IBM's DataJoiner [12] schema architecture can be seen as some kind of minimal approach, which imposes several restrictions, if we compare it to the reference architecture. The FOKIS [6] schema architecture has been designed according to the specific requirements of integrating replicated information among heterogeneous information systems. The mechanisms for publishing and subscribing to specific data items required some extensions to the generic part, while retaining its fundamental structure. The BLOOM [8] schema architecture extends the reference architecture with several structures for security mechanisms. These extensions require more fundamental changes to the generic specification components; thus placing this architecture at a more distant place in the specification hierarchy than it is the case with the former systems.

6. Conclusions and Future Work

The different ways of specializing the generic parts serve as a means to compare the specific systems with the reference architecture and with each other. The specialization hierarchy already offers a gross overview of the similarities and differences. The detailed comparison is done by means of the formal specification.

The individual federated database management systems were developed independently, as far as we know. However, the published papers on these systems all refer to the reference architecture of [9]; thus, all systems were inspired by that basic reference. Our comparison illustrates the similarities and differences of the realized specific architectures.

Federated database management systems are already a research topic for many years, and database systems in general are an well-understood research area with a reasonable formal underpinning. These established fundamentals helped with formalizing and comparing the architectures. In the future, we intend to apply the presented approach to less established models, such as agent and mediator architectures, to see whether reference architectures can (already) be identified for those domains.

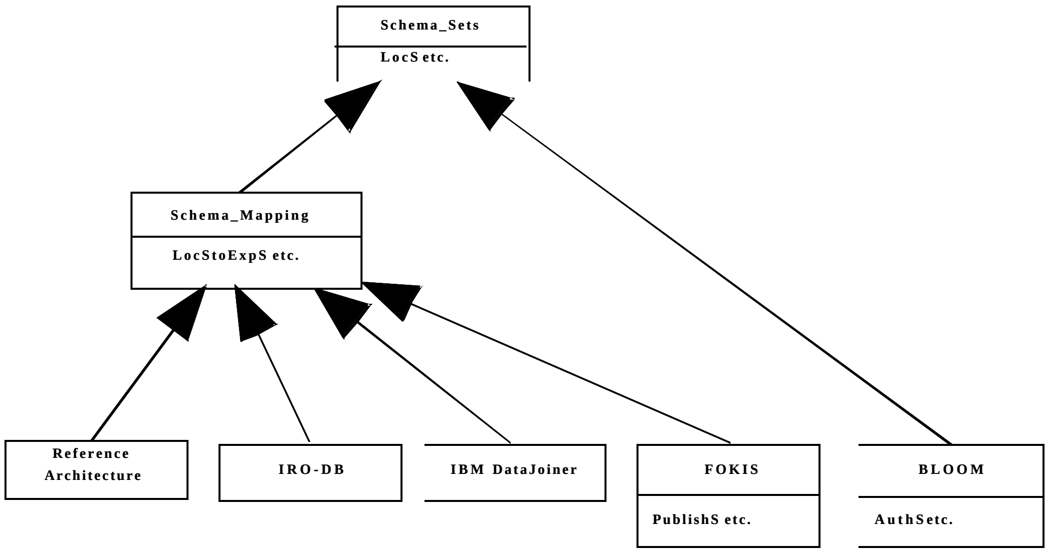

Figure 2: An extract of the structure of our Object-Z specification. Both the reference architecture and those of the selected systems are specializations of the generic classes.

References

[1] G. Booch, J. Rumbaugh, and I. Jacobson. *Unified Modeling Language User Guide*. Object Technology Series. Addison-Wesley, Reading, MA, 1999.

[2] S. Conrad, B. Eaglestone, W. Hasselbring, M. Roantree, F. Saltor, M. Schönhoff, M. Strässler, and M. Vermeer. Research Issues in Federated Database Systems (Report of EFDBS '97 Workshop). *SIGMOD Record*, 26(4):54–56, December 1997.

[3] C. J. Date. *An introduction to database systems*. Addison-Wesley, 6th edition, 1995.

[4] R. Duke, G. Rose, and G. Smith. Object-Z: A Specification Language Advocated for the Description of Standards. *Computer Standards and Interfaces*, 17:511–533, September 1995.

[5] G. Gardarin, B. Finance, and P. Fankhauser. Federating object-oriented and relational databases: The IRO-DB experience. In *Proc. 2nd IFCIS Intl. Conf. on Cooperative Information Systems (CoopIS'97)*, Kiawah Island, SC, 1997. IEEE Computer Society Press.

[6] W. Hasselbring. Federated integration of replicated information within hospitals. *International Journal on Digital Libraries*, 1(3):192–208, November 1997.

[7] Rational Software Corporation. The Unified Modeling Language. UML Document Set Version 1.3, Santa Clara, CA, 1999. (available from www.rational.com/uml/).

[8] F. Saltor, B. Campderrich, E. Rodriguez, and L. C. Rodriguez. On Schema Levels for Federated DB Systems. In *Proc. Parallel and Distributed Computing Systems (PDCS'96)*, pages 766–771, 1996.

[9] A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[10] G. Smith. *The Object-Z Specification Language*. Kluwer Academic Publishers, Boston, 1999.

[11] J. M. Spivey. *The Z* Notation: A Reference Manual. Prentice-Hall, 2nd edition, 1992.

[12] S. Venkataraman and T. Zhang. Heterogeneous database query optimization in DB2 Universal DataJoiner. In A. Gupta, O. Shmueli, and J. Widom, editors, *Proc. 24rd International Conference on Very Large Data Bases (VLDB'98)*, pages 685–689. Morgan Kaufmann, 1998.