Strategies for the Conceptual Design of Federated Information Systems

Susanne Busse, Ralf-Detlef Kutsche, Ulf Leser

TU Berlin, Computergestützte Informationssysteme CIS, EN7,
Einsteinufer 17, D-10587 Berlin, Germany
{sbusse, rkutsche, leser}@cs.tu-berlin.de

Abstract. We analyse two basic strategies for the development of tightly coupled, federated information systems: top-down and bottom-up. Both approaches are compared relative to the specific requirements of information integration in evolving environments, such as the intention of treating different kinds of heterogeneity, preserving source autonomy and enabling change management while ensuring consistency. We describe in detail how the intrinsic properties of such strategies affect their ability to cope with these requirements.
Based on the results of this study and on extensive experiences, we propose a combined strategy based on the intensive use of object-oriented modelling concepts. We show that this approach is well-suited to fulfil our vision of a consistent evolution within the paradigm of continuous engineering of federated information systems.

1 Introduction

1.1 Federated Information Systems (FIS)

In this paper, we discuss strategies for the development of large information systems, as part of long-lived enterprise-wide or world-wide information solutions. For a cost-effective and evolutionary development of such solutions, obviously valuable 'legacy' information resources have to be considered as members of the fedaration, which are in general, up to a different extent,

- physically distributed,
- heterogeneous, with respect to technical aspects, software, operational environment, structure, environment, data model and semantic issues,
- and finally, operationally independent, i. e. autonomous.

We consider systems where these resources are mainly constituted by a number of databases with local applications around them. The user's or customer's interest is to obtain an integrated access to the information resources as a "global" information system, see Fig. 1. We call such a system a *federated information system (FIS)* [3]. Federated information systems in our understanding differ from federated database systems as described in [15] in the fact that information resources are not restricted to database systems, but may include also other kinds of information provision.

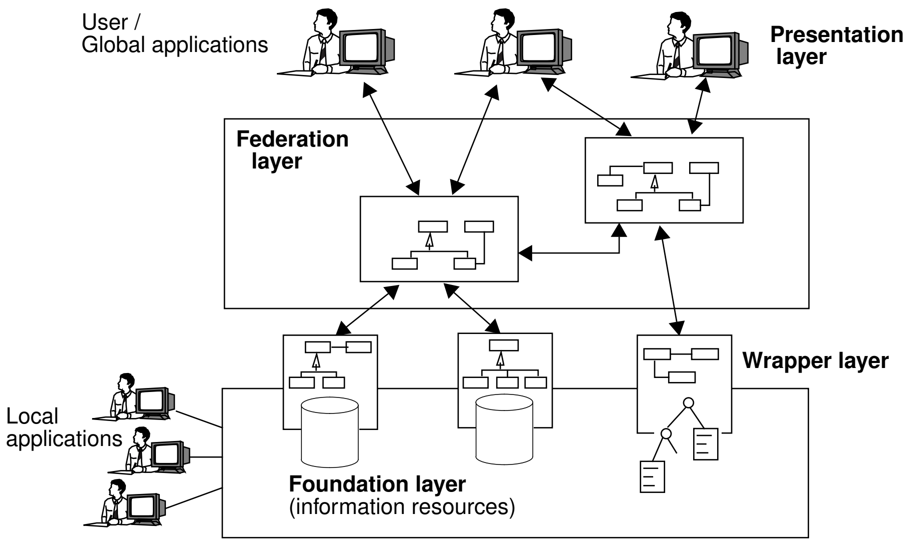

Figure 1: Architecture of Tightly Coupled FIS

The meanwhile classical work of Sheth and Larson ([15]) defines a reference schema architecture for tightly coupled federated database systems which use a global schema for integration.[1] They distinguish five different schema layers: The *local conceptual schemas* of the information resources are translated into the common data model (CDM) to form the *component schemas*. The relevant part of an information resource is identified in its *export schema*, related to the wrapper schema in our architecture (Fig. 1). Export schemas are integrated into *federation schemas* resolving structural and semantic heterogeneity of data sources. User's views on the information system are represented by *export schemas defined on the federation layer*, similar to views in centralized database systems.

Note that this scenario does not imply that federated schemas are developed from export schemas by schema integration, although this is a possibility frequently used. We call this the bottom-up strategy. Another approach is the top-down strategy:

- The *bottom-up* approach builds federations starting from the information resources towards the global schema, using a database integration method ([17], [16], [10]). The structure of the federated schema depends directly on the integrated export schemas and the integration method used.
- The *top-down* approach builds federations starting from a global schema determined by the information needs of the global information system. This involves a classical schema design problem and can for instance use object-oriented analysis and design methods or view integration techniques ([7], [13]). Component schemas are related to the federated schema in an extra step by means of correspondence specifications.

1. In contrast, loosely coupled systems only provide a uniform query language. We only consider tightly coupled FIS in this paper.

In this paper, we first characterise both strategies and highlight their respective advantages and pitfalls. Our results are based on extensive experiences in the development of FIS in diverse application domains like environmental information systems ([2]) and bioinformatics ([9]). After comparing both methods, we propose a combined strategy for the development of federated information systems which aims at combining the good and avoiding the bad aspects of the two approaches.

1.2 Evolution of Information Systems

Under the observation that software development today never starts from scratch, the classical view of the software engineering discipline has to be revised, recognizing software development as an continuously on-going process. This predominant paradigmatic issue in (particularly: federated!) information systems development can be subsumed under the term '*continuous software engineering*' (CSE, [12]).

Our CSE approach aims to define a methodology for a smooth and consistent process of development facing continuous change. It is based on classical techniques from software reengineering, including many valuable experiences from:

- classical forward engineering of new systems components, including the steps of analysis, design, and implementation,
- reverse engineering of existing legacy components, as far as relevant issues for a further development are missing, and
- the re-engineering basics starting from simple maintenance aspects via different levels of change and modification tasks until larger projects of renovation or even replacement of complete components (see e.g. [5]).

Focusing on the conceptual design of federated information systems, the CSE paradigm leads to the requirement that the development process has to be able to cope with continuous change and evolution. Having the reference architecture of Fig. 1 in mind, one recognizes two important situations that need to be addressed:

- at the bottom level, there will be a continuous change, emerging from newly developed, modified or additionally offered information resources. These new or changing components are in first place designed in the context of local requirements and do usually not conform to global requirements such as the structure of a global schema.
- from the top level, the desire for new information services will appear as soon as the global value of the given information structure is recognized by a relevant number of users, and an idea of possibly new sources exists.

The paradigm of 'continuous software engineering' is reflected in our strategy for the development of federated information systems by two main principles:

- The strategy combines top-down and bottom-up development steps, including both forward and reverse engineering.
- We extend the object-oriented modeling technique by model correspondences which explicitly specify the relationships between schemas. The explicit specification enables the continuous change on schema and configuration level.

2 Development Strategies for FIS

The two different strategies of engineering a tightly coupled FIS – bottom-up vs. top-down – have fundamental implications on the scope of usable development techniques, the possible degree of automation of the development process, the "strength" of the resulting relationships between the system layers, and particularly on the evolvability regarding the continuous engineering of the information system.

In this section, we will carefully analyse both strategies according to these points. We do not want to propagate one of the strategies for the development of all kinds of FIS, but rather to characterize them to give a basis to choose the best one of them according to the specific application requirements.

2.1 Bottom-up Strategy

Building a tightly integrated FIS bottom-up means that the initial requirement is the need to have an integrated access to a given number of data sources. A typical scenario is the need to have detailed and uniform access to all databases of a company to build global applications, possibly ahead of a migration. The information needs are on the same abstraction level as the data sources, i.e. aggregations, as they are typical for data warehouse scenarios, are not required.

In such a setting, a very tight connection between the federated and the component schemas is required, particularly if updates in sources should be done through the federation layer.[2] Therefore (semi-) formal integration methods should be applied which comply with the four classical requirements for schema integration, i.e., completeness, correctness, understandability and minimality ([4]).

The general integration process of these methods starts with the analysis of horizontal correspondences between component schemas. Based on those, an integrated schema is derived together with the correspondences between the integrated schema and the export schemas (see Fig. 2a). These correspondences typically take the form of view definitions; however, if updates are allowed, even those views must obey strict rules to avoid the notorious "update through view" problems. Only the second step can be done automatically (or with few user interaction); the definition of semantic correspondences between components on the contrary requires considerable user interaction.

The tight connection between federation and foundation layer greatly affects the evolvability of the system. A change in the configuration or in the export schema of one component triggers a new integration process, resulting in an adapted federated schema. Apart from this being a costly process which is impossible to automate, it often entails even more modifications, considering dependencies of global applications on the federation layer.

We conclude that a bottom-up strategy is appropriate to small and medium-sized FIS which require a read or write access to well integrated specific information resources without the need of further data aggregations. Because this strategy has several disad-

2. In the following, we will use the term 'component schema' synonymously for the export schema of a component.

vantages with respect to autonomy and evolution, it should be applied only in those scenarios where the configuration of the infrastructure and the sources are relatively stable ([11]).

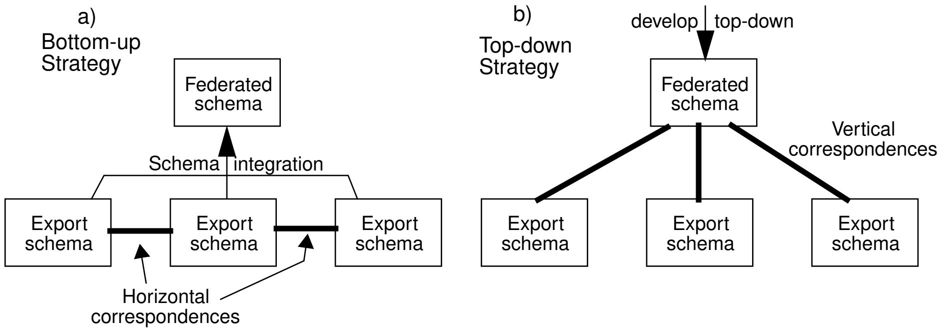

Figure 2: Development Strategies for FIS

2.2 Top-down Strategy

Top-down strategies root in classical database design methods: Starting from global information needs, the according views are integrated into one conceptual schema whose storage schema is then possibly assigned to distributed components.

Although we have to consider existing information resources, top-down approaches are oriented to global information needs, whereas the connection to the sources is not as relevant in first place. For instance, a company might want to offer the service to find the lowest book prices from different Internet stores; or a decision support system might want to integrate certain customer information that is spread over multiple department databases. In these cases, the actual schema of components does not matter for the design of the federated schema. From the four classical requirements for schema integration, i.e., completeness, correctness, understandability and minimality, two do not apply. Firstly, only relevant parts of schemas must be included (completeness). Secondly, there is no need to represent data on the global level exactly as in the components, if only derived or abstracted data is required (correctness; the mapping becomes uni-directional). One might for instance decide to cluster customers into salary groups and not to store the precise income. Note that such aggregations effectively exclude the possibilities of updating sources through the federated schema.

The global schema can either be generated ad-hoc or by a more formal analysis process, starting from use-case descriptions and ending by view integration techniques. Furthermore, applications which use common ontologies or standard schemas are also inherently top-down (e.g. STEP schemas [6] or OMG's domain standards).

Component schemas are considered in a second step, when vertical correspondences between the global schema and source schemas are established to allow for the translation of queries (see Fig. 2b). Each component can be considered separately from other sources when relating it to the federated schema. Correspondences in top-down approaches are necessarily more complex than in bottom-up strategies because the federated schema and the component schemas are designed independently. For instance, abstractions and aggregations are possible, and a much broader class of heterogeneity conflicts can occur. Correspondence specification languages need to provide enough expressiveness to cope with such problems.

Regarding the continuous engineering, we conclude that top-down strategies have advantages compared to bottom-up approaches. They are more appropriate in scenarios where sources are quickly evolving, if it often happens that sources are removed or new sources are added, if schema integration is infeasible or too expensive, or if the global requirements themselves are changing [8]. However, they typically result in a less 'tight' integration than bottom-up approaches and are bound up with less formal integration methods. For instance, the nature of the correspondences often prevents any update operations, restricting top-down to read-only scenarios.

2.3 Mixed Strategies

From this discussion, we can conclude that as well bottom-up approaches as top-down approaches in their pure form may apply well in particular FIS development scenarios. However, in the general case of complex information infrastructures based on several (previously unrelated) autonomous components and their evolution over long periods of time, we have to take the view of well-known software engineering methodologies, which typically are based on appropriate mixtures of bottom-up with top-down steps in analysis, design and realization of a given software product.

3 A Combined Top-Down/Bottom-Up Development Strategy based on Extensions of Object-oriented Modeling

Based on experiences in our current application project areas of environmental information systems (see for example [2]) and molecular biology ([9]) we propose a development strategy for the conceptual design of large-scale FIS which provide a global read-only access to autonomous legacy information systems.

To meet both the requirements of complex information needs and integration of existing information resources we combine a top-down strategy with a bottom-up approach. The central integration concept are model correspondence assertions which explicitly specify the relationships between schemas.

3.1 General Strategy

Our approach towards a model-based conceptual design of FIS includes the following steps:

- modelling use cases for the additional (global) presentation layer connecting the design in the information viewpoint to the enterprise viewpoint;
- modelling information structures and behavioural principles of the new component considering the domain of interest in general (using/building an ontology and classifications of the application area) as well as the specific information needs;

- modelling information structures and application contexts of the underlying legacy information systems only as far as required;
- if necessary, relating the underlying models as far as required (horizontal correspondences);
- relating the underlying models to the (only virtually existing) new structures (vertical correspondences);
- designing domain-specific and generic structural and behavioural models to allow reusability in future;
- iterating this process again and again, always specifying further refinements of the models themselves, but particularly of the dependencies and consistency conditions between them.

These steps form a multi-cycle development model. Note that they in general cannot be ordered into one process sequence but have to be carried out in parallel, reflecting their interdependencies. On one hand, the information which will be provided on-top naturally depends on existing resources; on the other hand, the legacy systems need to be chosen and analysed relative to the information needs on-top.

Our strategy results in a revised model architecture:

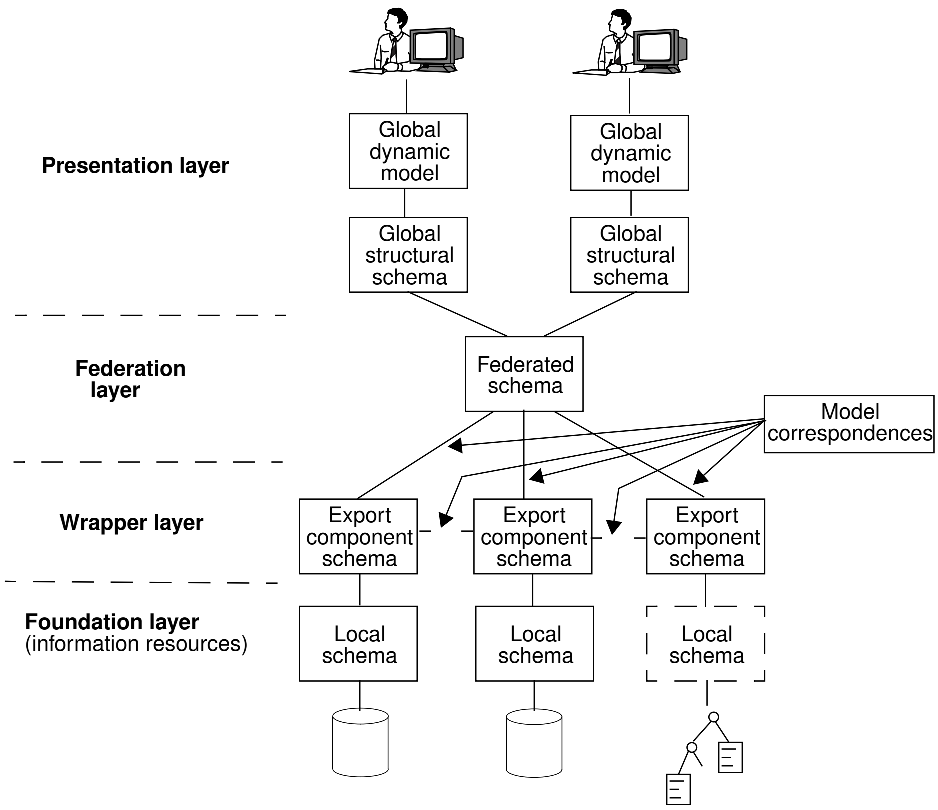

Figure 3: Model Architecture for FIS

This shows that complex information needs require modelling of structural as well as behavioural aspects. The underlying information resources are modelled as far as required. The federated schema is built top-down under consideration of existing information resources. This implies that we concentrate on the vertical correspondences to benefit from the advantages regarding schema changes as described above.

We use explicitly formulated model correspondences which relate the federated schema to the underlying export schemas. They are specified separately for each component to emphasize the autonomy of components. Correspondence specifications, which in the essence are a special kind of meta data, should be treated as explicit data in the federation layer. By doing so, one can react on changes by data manipulation operations, like updates or deletion, instead of reprogramming and recompilation.

3.2 Applying Object-Oriented Modelling Languages

Technically, we make use of the manifold of techniques in the Unified Modeling Language (UML [14]) and supplementing techniques within the object-oriented paradigm.

Our experiences show that the object-oriented paradigm supports the design and evolution of federated information systems in different directions:

- It helps in (reverse) modelling of relevant parts of legacy systems as well as in the (forward) analysis and design of new components.
- It helps to relate information structures and their use within business processes by integrating structural and behavioural modelling.
- It helps to relate conceptual and architectural design within a component-based engineering approach — one important aspect in continuous engineering of the system regarding all engineering viewpoints.
- They are extensible with techniques to support the specification of relationships between autonomous information models — one important aspect in the context of FIS.

Some of the UML diagramming languages have been proven useful in the concrete application of our general development strategy. For the (top-down-related) modelling starting from use cases towards a federated ’goal’ schema use case diagrams and their refinements via activity and sequence diagrams can be applied instead of textual descriptions. Class diagrams are used for structural modelling of the federation layer (top-down again) and of underlying resources (in a sense of bottom-up reverse modelling of different sources using a uniform notation). Finally, behaviour will be modelled as far as needed by using state diagrams for internal behaviour; sequence diagrams or collaboration diagrams for typical collaboration sequences in the whole system. The semantic equivalence between sequence and collaboration diagrams, moreover, helps in improving coherence between structural and behavioural models.

Relating different schemas and resolving heterogeneity conflicts is the most important task in conceptual design of FIS. [1] introduces a formal specification language for model correspondence assertions in federated information systems. It allows both the specification of horizontal and vertical correspondences and considers the special requirements on correspondences of our development strategy. As such correspondence specifications are not appropriately expressible in existing UML notation, we suggest an extension to the class diagrams for explicitation of these correspondences by combining the graphical notation with our formal specification language.

4 Conclusions

Based on experiences in state environmental information systems and systems in molecular biology, we analysed development strategies for tightly coupled, federated information systems. We focused on the main requirements for any such approach:

- The need to integrate existing components that are in physically distributed, heterogeneous, and autonomous;
- The need to comfort continuous change and evolution.

We conlude that a bottom-up strategy is appropriate for scenarios in which a tight integration of a given set of immutable information resources without data abstraction is required. Therein, formal integration methods can be applied satisfying the requirements of completeness and correctness. In contrast, a top-down strategy is beneficial if more complex information needs exist, or if continous change is prevailing. Generally, top-down approaches result in a less 'tight', but more flexible integration than bottom-up approaches.

Based on this analysis we described a combined object-oriented approach. It basically follows a top-down approach to build the federated schema considering both structural and behavioural aspects. The information resources are analysed and modelled as far as required simultaneously to the global level. The connection between federation and foundation layer is established using explicitly formulated model correspondence assertions. The separation of correspondences of different sources and the explicit specification supports the evolution of the resulting information infrastructure.

We argued for object-oriented modelling techniques such as use-cases and structural and behavioural modelling, extended with special techniques to specify vertical and horizontal correspondences. Even without introducing language and method extensions, object-oriented techniques seem to be appropriate since they support a continuous software engineering process, including forward as well as reverse engineering steps.

References

[1] S. Busse, *A specification language for Model Correspondence Assertions*, *Part I: Overlap Correspondences*, Technical Report Nr. 99-8, TU Berlin, 1999.

[2] S. Busse, R.-D. Kutsche, *Metainformationsmodelle für flexibles Information Retrieval in vernetzten Umweltinformationsstrukturen*, in: H.-D. Haasis, K.C. Ranze (Hrsg.), *Umweltinformatik '98 — Vernetzte Strukturen in Informatik, Umwelt und Wirtschaft*, Proc. 12. Int. Symposium der GI, Bremen, Sept. 1998, pp. 583-596, Metropolis, 1998.

[3] S. Busse, R.-D. Kutsche, U. Leser, H. Weber, *Federated Information Systems: concepts, terminology and architectures*, Technical Report Nr. 99-9, TU Berlin, 1999.

[4] C. Batini, M. Lenzerini, S.B. Navathe, *A comparative analysis of methodologies for database schema integration*, ACM Computing Surveys, Vol. 18, No. 4, pp. 323-364, Dec. 1986.

[5] E. Chikofsky, J. Cross, *Reverse engineering and design recovery*, IEEE Software, Jan. 1990.

[6] ISO International Standard 10303, *Industrial automation systems and integration – Product data representation and exchange*, 1994 and 1997.

[7] I. Jacobson, G. Booch, J. Rumbaugh, *The Unified Software Development Process*, Addison-Wesley, 1999.

[8] U. Leser, *Maintenance and mediation in federated databases*. 8th Workshop on Information Technology and Systems, Helsinki, Finland, TR-19, University of Jyvaeskylae pp. 187-196, 1998.

[9] U. Leser, Designing a global information resource for molecular biology, Datenbanken in Büro, Technik und Wissenschaft (BTW), Freiburg, Germany, pp. 362-369, Springer, 1999.

[10] J.A. Larson, S.B. Navathe, R. Elmasri, *A theory of attribute equivalence in databases with applications to schema integration*, IEEE Transactions on Software Engineering, Vol. 15, No. 4, pp. 449-463, Apr. 1989.

[11] R. Motz, *Propagation of structural modifications to an integrated schema*. 2nd East European Symposium on Advances in Databases and Information Systems; LNCS 1475, Poznan, Poland pp. 163-174, 1998.

[12] H. Müller, H. Weber (eds.), *Continuous engineering of industrial-scale software systems*, seminar report #98092, 2.-6. März 1998, IBFI, Schloß Dagstuhl, 1998.

[13] S.B. Navathe, S.G. Gadgil, *A methodology for view integration in logical database design*, Proc. 8th Conf. on Very Large Databases, VLDB, Mexico City, pp. 142-164, Sept. 1982.

[14] Object Management Group, *The Unified Modeling Language (UML) Specification – Version 1.3*, available at http://www.omg.org/, 1999.

[15] A.P. Sheth, J.A. Larson, *Federated database systems for managing distributed, heterogeneous, and autonomous databases*, ACM Computing Surveys, Vol. 22, No. 3, pp. 183-236, Sep. 1990.

[16] S. Spaccapietra, C. Parent, Y. Dupont, *Model independent assertions for integration of heterogeneous schemas*, The VLDB Jounal, Vol. 1, No. 1, pp. 81-126, Jul. 1992.

[17] I. Schmitt, G. Saake, *Schema integration and view generation by resolving intensional and extensional overlappings*, in: K. Yetongnon, S. Hariri (eds.), Proc. 9th ICSA Int. Conf. on Parallel and Distributed Computing Systems (PCDS'96), pp. 751-758, Sep. 1996.