CASE Support for the Development of Federated Information Systems

Ph. Thiran, A. Chougrani, J-L. Hainaut

InterDB Project[1], Database Applications Engineering Laboratory
Institute of Informatics, University of Namur, Belgium
pth@info.fundp.ac.be

Abstract

This paper analyses the requirements that CASE tools should meet for the development of federated information systems, and presents a general architecture for such CASE environments. The main characteristics of the integration process and architecture components are analyzed in order to collect a set of desired requirements. We finally present an operational CASE tool which is intended to address some of these requirements.

1. Introduction

1.1. The problem and its context

Federated Information Systems (FIS) [Sheth, 1990] provide transparent access to heterogeneous data sources. The development of FIS involves yielding local schemas, integrating them into one or more schemas, controlling complex mappings and building architecture components. The problem is complicated by the fact that the data sources have been developed independently, and naturally suffer from several problems of replication and semantic conflicts.
Like any complex process, developing FIS cannot be successful without the support of adequate tools called CASE tools [Conrad, 1999]. Nevertheless, completely automating this process is unrealistic for real world systems [Sheth, 1991]. Hence the need for computer-based assistance tools which address several aspects of the development of FIS.

1.2. State of Art

Several CASE tools are now available for building FIS. They are dedicated either to the integration process or to the building of federation components:

- **For the integration process**: [Hayne, 1992], [Gotthard, 1992] and [Ramesh, 1995] proposes tools for automated interschema relationship identification. [Schwarz, 1998] presents a set of tools that support different issues of the process, e.g., methodology, conflicts and similarities identification, semantics extraction.
- **For the building of federation components**: [Papakonstantinou, 1995] proposes an implementation toolkit that facilitates the rapid development of wrappers and mediators; [Vidal, 1998] presents a meta-mediator providing a single meta-mediator interface for all the sources. HERMES [Subrahmanian, 1995] provides a set of tools to support the construction of mediators.

Many of these tools, however, appear to be limited in scope, and are generally dedicated to a limited aspect of the FIS development. They do not attempt to integrate techniques and reasoning common to the integration process and the building of federation components, leaving the question of a general tool for developing FIS unanswered.
Moreover, it appears that the extraction of semantics information from existing systems is often left aside even though the FIS community ([Schwarz, 1998], [Conrad, 1999]) considers that schema integration requires a semantically rich description of the federation components.

---

[1] The InterDB project is supported by the Belgian *Région Wallonne*.

In [Thiran, 1998], we proposed the baselines for a methodology and a CASE support for the development of a federation. Theses baselines have been developed in [Hainaut, 1999] and [Thiran, 1999]. The current paper translates these principles into practical requirements that CASE tools should meet and presents the main aspects and components of a CASE tool that supports both the schema integration and the building of federation components.

1.3. About this paper

This paper is organized as fellows. Section 2 presents a standard architecture and a general methodology of FIS. Section 3 discusses some important requirements whish should be satisfied by a CASE tools for the development of FIS. Section 4 presents an operational CASE tool which is intended to address these requirements and describes in further detail some of the original principles and components of this CASE tool.

2. An overview of Federated Information Systems

2.1. Architecture

A general architecture is shown in Figure 1. The architecture comprises a hierarchy of mediators that dynamically transform queries based on a federated schema into physical queries based on the physical schema of the sources. To simplify the discussion, the view schemas have been ignored.

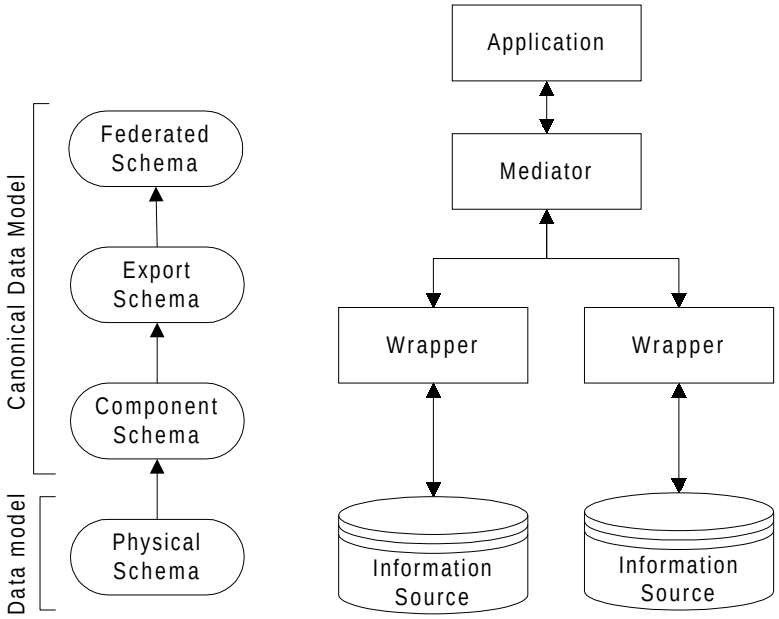

Figure 1 – A general architecture of FIS

2.1.1. Hierarchy architecture

The general architecture of federated information systems has been described in [Sheth, 1990]. It consists of a hierarchy of data descriptions that ensure independence according to different dimensions of heterogeneity. According to this framework (Cf. Figure 1), each local data source is described by its own **component schema**, from which a subset called **export schema**, is extracted. The latter are merged into **the federated schema**. The federated schema as well as the component and export schemas are expressed in a **canonical data model** which is independent of the underlying technologies. Object-oriented data models are commonly used as the canonical data model for the federations. Object-oriented data models provide rich data structuring possibilities, which enable to express the semantics of local schemas expresses in other models, such as the relational or XML models.

2.1.2. Component architecture

To address the problem of interoperability of information systems in general, the term **mediation** has been defined in [Wiederhold, 1995] as a service that links data resources and application programs.
A **mediator** is a software module that exploits encoded knowledge about some sets or subsets of data to create information for applications [Wiederhold, 1992]. A mediator hides details about the location and representation of relevant data to applications.
Above each source is a **wrapper**. A wrapper is a software component that performs the translation between the export schema and the physical schema of a source [Papakonstantinou, 1995]. That is, the wrapper (1) offers an export schema in the canonical data model (2) accepts queries against the export schema and translates them into queries understandable by the underlying database, and (3) transforms the results of the local queries into a format understood by the application.
Wrappers and mediators relies on schema descriptions and mappings to translate queries and to form the result instances. That is, they can be complex if the mapping rules are complex. Hence the need for a methodology and a CASE support to build them.

2.2. General methodology

Referring to the architecture of Figure 1, we distinguish four main tasks addressing the development of FIS: (1) recovering the semantics of the information sources, (2) integrating them, (3) defining the mappings and (4) building the architecture components. Since this has been presented in former papers (see [Thiran, 1998] and [Hainaut, 1999]), we will only outline the main steps of the methodology.

2.2.1. Semantics recovery

In this phase, schemas that correspond to the information sources being integrated are translated into schemas using the canonical data model. However, if additional information about the information sources can be provided during this process, a semantically richer description of the federation components may be exploited. Rich semantic descriptions are useful in a database federations, as they allow to more properly detect semantic heterogeneity ([Thiran, 1998], [Ram, 1999], [Conrad, 1999]).
Extracting a semantically rich description from a data source is the main goal of the **data-centered reverse engineering process** (DBRE). A general DBRE methodology has been developed in the DB-MAIN laboratory (see e.g. [Hainaut, 1996a]). This methodology can be specialized according to the various data models which most legacy systems are based on, such as standard files, or CODASYL, IMS and relational databases. Moreover, this methodology can be extended for semi-structured information source like XML.

2.2.2. Integration

The integration is the process of identifying the components of export schemas which are related to one another, **identifying** and **solving the conflicts** of these schemas, and finally, **merging export schemas into a federated one**. Conflicts fall into three possible categories: syntactic, semantic and instance. Solving the conflicts occurring in heterogeneous databases has been studied in numerous references, by e.g. [Batini, 1986], [Spaccapietra, 1991], [Vermeer, 1996], [Parent, 1998] and [Ram, 1999]. In [Hainaut, 1999], we have noted that most conflicts can be solved through three main techniques: renaming, transforming and discarding. Heuristics exist to cope with this problem [Spaccapietra, 1991]. Since the syntactic, semantic and instance conflicts have been resolved by restructuring the export schemas, merging the latter is fairly straightforward, and can be automated to a large extent [Hainaut, 1999].

2.2.3. Defining the Mappings

An analysis of the schema recovery and integration processes shows that deriving a schema from another one is performed through techniques such as renaming, translating, solving conflicts, which basically are schema transformations. As suggested in [Hainaut, 1996b], most data-centered engineering processes can be formalized as a chain of schema transformations. This is the case for RDBE and integration, as demonstrated in [Hainaut, 1999].

2.2.4. Building the architecture components

The architecture components rely on the structural mapping and the schema description for the queries translation. It depends also on the data mapping for the results formation [Hainaut, 1999]. The mappings are pure transformational functions that cannot be immediately translated into executable procedures in 3GL. However, it is fairly easy to produce procedural data conversion programs as shown in [Hainaut, 1996a].

3. CASE tool requirements

This section states some of the most important requirements an ideal CASE tool environment for the development of FIS should meet. Besides standard functions of a data-oriented CASE tool, an "ideal" CASE tool for the development of FIS should support the specific aspects of the development of FIS: (1) semantics recovery of the information sources, (2) integration, (3) mapping definition and (4) building of architecture components. The requirements are induced by the analysis of each of these aspects.

3.1. General support

Observations
Developing FIS is a data-oriented engineering activity.
CASE tool requirements
The CASE tool must offer standard functions that are now provided by most CASE tools dedicated to data-oriented engineering : specifications management, evaluation, graphical viewing, reporting and code generation, etc.

Observations
Schema transformation is at the core of methodologies that manipulate schemas. Therefore, automating these transformations increases the power and the reliability of the tool.
CASE tool requirements
The CASE tool must provide a rich set of transformation techniques.

Observations
Building FIS is basically exploratory and often unstructured activity. Some important aspects of higher level specifications cannot be deterministically inferred.
CASE tool requirements
The tool must allow the user to fellow working patterns, including unstructured ones. It should be methodologically neutral and allow various engineering strategies; ranging from formal approaches to informal and pragmatic ones. In addition, the tool must be highly interactive.

Observations
There is (and probably will be) no available tool that can satisfy all corporate needs in development of FIS. In addition, current CASE tools already provide elaborated techniques that deals with some specific aspects of the design process.
CASE tool requirements
A CASE tool must communicate easily with the other development tools (e.g., exchanging specifications through a common format such as XML, or a common repository).

3.2. Support of the semantics recovery

Observations
FIS architectures require to support a great variety of legacy systems running on different platforms. These legacy systems include not only structured information sources but also semi-structured and even unstructured information [Conrad, 1999].

CASE tool requirements
Customizable DBRE functions for automatic, interactive and assisted specification extraction should be available for each source types. The CASE tool must also provide sophisticated text analysis processors. They should be easy to customize and to program.

Observations
The canonical data model is designed to express all the semantics of the component schemas.
CASE tool requirements
The canonical data model must be highly generic and more flexible than the legacy data models [Garcia, 1997].

Observations
The semantic enrichment requires a great variety of information: data structure, data (from files or databases), CASE repository, documentation, domain knowledge, etc.
CASE tool requirements
The tool must provide several ways of viewing and querying these sources. The tools must also include customizable functions for automatic and assisted specification analysis.

3.3. Support of the integration

Observations
Several integration strategies can be applied, depending on the complexity and the heterogeneity of the source databases and on skill of the analyst [Hainaut, 1999].
CASE tool requirements
The tools must include a collection of basic techniques for the integration instead of a unique, automated, schema integrator.

Observations
There exists a collection of commonly used conflicts strategies that can be applied for conflicts solving [Subrahmanian, 1995].
CASE tool requirements
The CASE tool must include a set of pre-defined functions for detecting conflicts.

Observations
However, each federation is a new problem of its own, requiring specific reasoning and techniques. Integrating export schemas into a federated one appears as a learning activity.
CASE tool requirements
The pre-defined functions should be easy to customize and to program. Moreover, specific functions should be easy to develop, even for one-shot use.

Observations
Conflicts occur in three possible ways : syntactic, semantic and instance. Therefore, there resolution requires a great variety of information sources : schemas, data (from files, databases, spreadsheets, etc.), data structure, data mining analysis, domain knowledge, etc.
CASE tool requirements
The tool must include browsing and querying interfaces with these sources. Customizable functions for assisted specification analysis should be available for each of them. In particular, the tool must include data mining techniques for the instance conflicts identification.

Observations
[Conrad, 1999] notes that meta-data and ontologies are highly relevant for enabling interoperability.
CASE tool requirements
The canonical data model must have a meta-layer which can be customized for integrating meta-data.

3.4. Support of the mappings definition

Observations
Developing FIS includes at least three sets of documents: the data schemas, the export schemas and the federated schemas. The forward and backward mappings between the schemas specification must be precisely recorded.
CASE tool requirements
The repository of the CASE tool must record all the mappings between the schemas. These mappings should be formally defined in order to allow the architecture components to be building without human intervention.

3.5. Support for the building of the architecture components

Observations
Wrappers/Mediators specification is based on mappings definition.
CASE tool requirements
The CASE tool should automatically generate and maintain information about mappings between schemas of the federation. The tool must also include sophisticated automatic or assisted mappings analyzers. In particular, it must provide several ways of viewing both mappings definition and schemas.

Observations
Moreover, further information (e.g., transaction management or security) is necessary to build efficient wrapper/mediator.
CASE tool requirements
The CASE tool should maintain any type of information that could be used for specific need. The tool must provide customizable functions for accessing and analyzing such information.

Observations
Wrappers specification also depends on the set of DML primitives supported by the underlying data management systems (COBOL file primitives, SQL, DBTG DML, etc.) [Garcia, 1997].
CASE tool requirements
The CASE tool must manage the differences between the data management systems, i.e., it should be able to build specific wrappers for each type of systems.

Observations
In federated architectures, new wrappers and mediators can join an existing federation [Wiederhold, 1992].
CASE tool requirements
The CASE tool must be able to build new wrappers and mediators easily.

4. The DB-MAIN CASE tool

The DB-MAIN CASE environment [Hick, 1999] is a complete set of tools dedicated to database applications engineering. This graphical, repository-based, software engineering environment is dedicated to database applications engineering. The DB-MAIN CASE tool addresses the main requirements developed in the previous sections. As a large-scope CASE tool, DB-MAIN includes usual functions needed in data analysis and design, e.g. entry, browsing, management, validation, transformation, as well as code and report generation. The rest of this paper will concentrate only the main aspects and components of the tool which are directly related to the development of FIS.

4.1. General support

4.1.1. Repository and common model specification

The DB-MAIN repository can collect all the information related to a FIS development. The repository comprises three classes of information:

- a generic model for the schemas specification;
- a specification of a methodology followed to the FIS development (Cf. [Roland, 1997]);
- a history that records the mappings (Cf. Section 4.4).

The schema specification is based on a unique wide spectrum specification model, namely the **generic model**. This model is an abstract formalism intended to express data structures independently of the implementation technologies. In short, physical schemas, conceptual schemas, export schemas as well as federated schemas are expressed into an unique and generic entity/object-relationship model.
Besides the standard concepts, the generic model includes some **meta-objects** which can be customized according to specific needs. These features provide dynamic extensibility of the generic model. For instance, new concepts such as correspondence types can be represented by specializing the meta-objects.

4.1.2. User interface

User interaction uses a fairly standard GUI. Browsing through several sources require an adequate presentation of specifications. It appears that more than one way of viewing them is necessary. For instance, a graphical representation of schemas allows an easy detection of certain structural patterns, but it is useless to analyzes name correspondences and similarities. DB-MAIN currently offers six ways of presenting a schema (four hypertext views and two graphical views).

4.1.3. Transformation toolkits

DB-MAIN includes sophisticated transformational toolkits to manipulate the specifications with semantics-preserving operators (see [Hainaut, 1996b] or [Hick, 1999] for more detail).

4.1.4. Functional extensibility

No CASE tools can satisfy the needs of all users in any possible situation. DB-MAIN provides a set of built-in standard functions. As we will see in the following sections, DB-MAIN is sufficient to satisfy most basic needs in the development of FIS. However, specialized operators may be needed to deal with specific situations. In addition, other CASE tools may have powerful and useful functions.
To offer such extensions, DB-MAIN provides the ***Voyager 2*** development environment [Englebert, 1999] which offers: (1) predicative access to the repository; (2) analysis and generation of external text. *Voyager 2* is a complete, 4th generation, semi-procedural language. It allows the development of applications which can be incorporated in the tool. For instance, it allows building functions for detecting particular conflicts and for importing and exchanging specifications with other CASE tool.

4.2. Support of the semantics recovery

DB-MAIN offers functions and processors that are specific to reverse-engineering [Hainaut, 1996c]. The data structures extraction is carried out by a series of **extractors**. These processors identify and parse the declaration part of the source texts, or analyze catalog tables, and create corresponding abstractions in the repository. Extractors have been developed for SQL, COBOL, CODASYL, IMS and RPG data structures. Additional extractors can be developed easily thanks to the *Voyager 2* environment. For instance, a program can analyze texts in any language and according to any dialect.
The semantics discovery is supported by a collection of **processors** that help analyze program code, schemas and data in order to find semantics (e.g. a *variable dependency analyzer* that detects and displays the dependencies between the objects (variables, constants, records) of a program ; a *foreign key assistant* that help find the possible foreign keys of a schema). The schemas enrichment can be

performed in a reliable way thanks to the semantics-preserving **transformation toolset**. Transformation scripts that implement specific heuristics can be quickly developed. A programmable schema analysis processor can be used to detect structural patterns and problematic constructs to be further processed.

4.3. Support of the integration

Several integration strategies can be applied, depending on the complexity and the heterogeneity of the information sources and on skill of the analyst. As a consequence, DB-MAIN offers a toolbox for schema integration instead of a unique, automated, schema integrator. Together with the transformation toolbox, the **integration toolkit** allows manual, semi-automatic and fully automatic integration. The purpose of the integration toolkit is to assist the analyst in dealing with conflicts that might arise during objects integration [Hainaut, 1999]. The toolkit presents as set of integration strategies. For a selected strategy, it compares the objects properties and presents them whenever a conflict is detected. Solving the conflicts is up to the analyst.

4.4. Support of the mapping definition

DB-MAIN automatically generates and maintains a **history log** of all the transformations that are carried out when deriving an export schema from a physical schema and when merging exports schemas into a federated one. The history is completely formalized in such a way that it can be replayed, analyzed and transformed. Moreover, the history can be normalized in order to remove useless sequences and dead-end exploratory branches.

4.5. Support for the building of architecture components

As part of the InterDB project [Thiran, 1998], we have developed (1) wrappers generators and (2) a meta-mediator accessing the DB-MAIN repository (Cf. Figure 2).

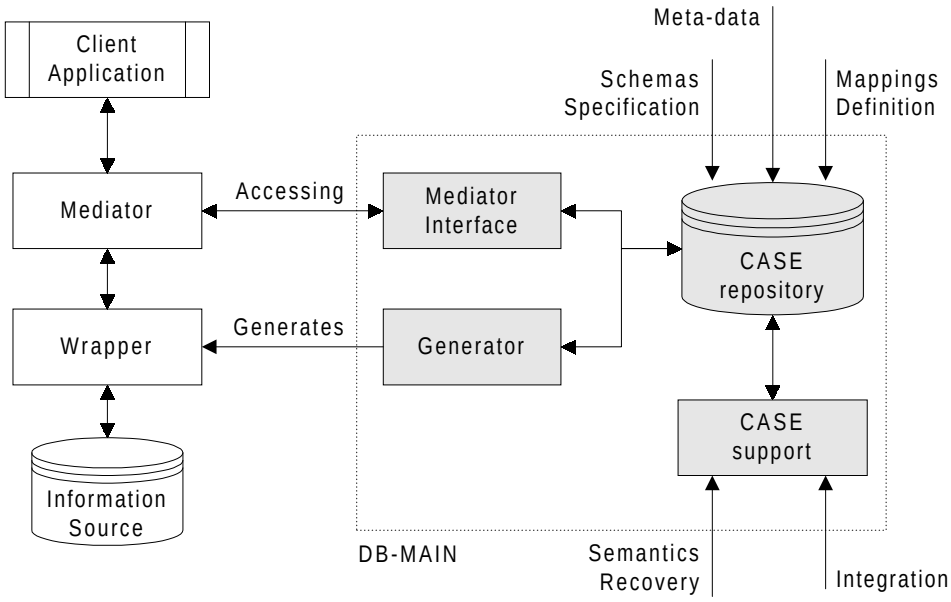

Figure 2 – DB-MAIN as support for the building of architecture components

For efficiency reasons, **wrappers** are developed as program components dedicated to an information source. In particular, mapping rules are hardcoded in the wrappers. **Wrappers Generators** have been developed in *Voyager 2* that allows to develop repository-based analyzers and to generate the wrapper code. The inter-schema mappings are used to generate the wrappers. At the current time, generators for COBOL files and RDB data structures are available.

Meta-Mediators are based on (1) the DB-MAIN repository that describes the federated schema, the export schemas of each information sources extension and (2) an extension of the DB-MAIN repository that holds the data location, and the relationships between export and federated schemas and other useful meta-information (e.g. security, source liability, etc.). All theses pieces of information concerning data replication, semantic conflicts and data heterogeneity allow the meta-mediator to interpret and distribute the global queries, and to collect and integrate the results sent back the wrappers.

5. Conclusions

Considering the requirements sketched in Section 3, few (if any) CASE tools offer the functions necessary for the development of FIS. In particular, they do not integrate all the aspects of the FIS development. Moreover, their functional and semantics extensibility is most often limited, so that hey cannot cope with unexpected problems that arise when integrating a particular FIS. The DB-MAIN CASE tool presented in this paper includes several functions which try to meet many of the requirements described in Section 3. In particular, this paper reports on two original aspects of DB-MAIN.

First, DB-MAIN gives the analyst an **integrated toolset** for the development of FIS, providing, for instance, functions for reverse engineering, integration, mappings definition and building architecture components.

The second original aspect of DB-MAIN is its **Meta-CASE layer**, which allows developers to customize the tool and its repository and to add new concepts, functions, and even models. In particular, DB-MAIN offers a complete development language, Voyager 2, through which mapping analyzers and transformers, schema analyzers, wrappers generators have been written as seamless add-ons to the standard CASE engine.

Though the tool has not been used on large scale projects yet, the first case studies show that complete and reliable federated systems based on distributed COBOL files and SQL databases can be generated in a semi-automated way.

6. References

[Batini, 1986] C. Batini, M. Lenserini, and S.B. Navathe, “A Comparative Analysis of Methodologies for Database Schema Integration”, *ACM Computing Surveys*, 18(4), pp. 323-364, December, 1986.

[Conrad, 1999] S. Conrad, W. Hasselbring, U. Hohenstein, R-D. Kutsche, M. Roantree, G. Saake, F. Saltor, “Engineering Federated Information Systems – Report of EFIS’99 Workshop”, *ACM SIGMOD Record*, 28(3), 1999.

[Englebert, 1999] V. Englebert, "The Voyager 2 language", *Research rapport*, Institut d'informatique, University of Namur, Namur, 1999.

[Gotthard, 1992] W. Gotthard, P.C. Lockemann, A. Neufeld, "System Guided View Integration for Object-Oriented Databases", *IEEE Transactions on Knowledge and Data Engineering*, Volume 4(1), pp. 1-22, 1992.

[Garcia, 1997] H. Garcia-Molina , Y. Papakonstantinou , D. Quass , A. Rajaraman , Y. Sagiv, J. Ullman, V. Vassalos , J. Widom, “The TSIMMIS approach to mediation: Data models and Languages”, *Journal of Intelligent Information Systems*, 1997.

[Hainaut, 1996a] Hainaut J.-L., Henrard J., Hick J.-M., Roland D., Englebert V., Database Design Recovery, in Proc. of the 8th Conf. on Advanced Information Systems Engineering (CAISE'96), Springer-Verlag, 1996.

[Hainaut, 1996b] J-L. Hainaut, “Specification preservation in schema transformations - Application to semantics and statistics”, *Data & Knowledge Engineering*, Elsevier Science Publish, 16(1), 1996.

[Hainaut, 1996c] Hainaut J.-L., Englebert V., Henrard J., Hick J.-M., Roland D., Database Reverse Engineering : from Requirement to CARE tools, *Journal of Automated Software Engineering*, Volume 3(2), Kluwer Academic Press, 1996.

[Hainaut, 1999] J-L. Hainaut, Ph. Thiran, J-M. Hick, S. Bodart, A. Deflorenne, "Methodology and CASE tools for the development of federated databases", the *International Journal of Cooperative Information Systems*, Volume 8(2-3), pp. 169-194, World Scientific, June and September, 1999.
[Hayne, 1992] S. Hayne and S. Ram. "Multi-user view integration system (MUVIS): an expert system for view integration, *Proceedings of the Sixth International Conference on Data Engineering*, IEEE, Los Almitos, February 1990.
[Hick, 1999] J.-M. Hick, V. Englebert, J. Henrard, D. Roland, J-L. Hainaut, "The DB-MAIN Database Engineering CASE Tool (version 5) - Functions Overview", *DB-MAIN Technical manual*, Institut d'informatique, University of Namur, December 1999.
[Papakonstantinou, 1995] Y. Papakonstantinou, A. Gupta, H. Garcia-Molina, J. Ullman, "A Query Translation Scheme for Rapid Implementation of Wrappers", *International Conference on Deductive and Object-Oriented Databases*, 1995.
[Parent, 1998] C. Parent and S. Spaccapietra, "Issues and Approaches of Database Integration", *Communications of the ACM*, 41(5), pp.166-178, 1998.
[Ramesh, 1995] V. Ramesh and S. Ram "A methodology for interschema relationship identification in heterogeneous databases, *Proceedings of the Hawaii International Conference on Systems and Sciences*, pp. 263-272, 1995.
[Roland, 1997] D. Roland, J-L. Hainaut, Database Engineering Process Modeling, in *Proc. of the Int. Conference on The Many Facets of Process Engineering*, Gammarth (Tunis), Sept. 1997.
[Sheth, 1990] A.P. Sheth and J.A. Larson "Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases", *ACM Computing Surveys*, 22(3):183-236, September 1990.
[Sheth, 1991] A.P. Sheth, "Issues in Schema Integration", Perspective of an Industrial Researcher", *ARO Workshop on Heterogeneous Database*, 1991.
[Schwarz, 1999] K. Schwarz, I. Schmitt, C. Türker, M. Höding, E. Hildebrandt, S. Balko, S. Conrad, G. Saake, "Design Support for Database Federations", in *proceedings of ER'99*, Paris, November 1999.
[Spaccapietra, 1991] S. Spaccapietra and C. Parent, "Conflicts and correspondence assertions in interoperable databases", *SIGMOD Record*, 20(4), pp. 49-54, December 1991.
[Subrahmanian, 1995] V.S. Subrahmanian, S. Adali, A. Brink, R.E. J.ames J.L.A. Rajput, T.J. Rogers, R. Ross, Ch. Ward, "HERMES: Heterogeneous Reasoning and Mediator System", University of Maryland, 1995.
[Thiran, 1998] Ph. Thiran, J-L. Hainaut, S. Bodart, A. Deflorenne, J-M. Hick, "Interoperation of Independent, Heterogeneous and Distributed Databases. Methodology and CASE Support: the InterDB Approach" in *Proceedings of CoopIS'98*, IEEE, New-York, August 1998.
[Thiran, 1999] Ph. Thiran, A. Chougrani, J-M. Hick, J-L. Hainaut, "Generation of Conceptual Wrappers for Legacy Databases", in *proceedings of DEXA'99*, LCNS, Springer-Verlag, Florence, September 1999.
[Vermeer, 1996] M.W.W. Vermeer and P.M.G Apers, "On the Applicability of Schema Integration Techniques to Database Interoperation", in *Proc. Of 15th Int. Conf. On Conceptual Modeling*, ER'96, Cottbus, pp. 179-194, Oct. 1996.
[Vidal, 1998] M.E. Vidal, L. Rashid, J.R. Gruser "A Meta-Wrapper for Scaling up to Multiple Autonomous Distributed Information Sources", in *proceedings of CoopIS'98*, IEEE, New-York, August 1999.
[Wiederhold, 1992] G. Wiederhold, "Mediators in the Architecture of Future Information Systems", *IEEE Computer*, pp. 38-49, March 1992.
[Wiederhold, 1995] G. Wiederhold, "Value-Added Mediation in Large-Scale Information Systems", *IFIP Data Semantics (DS-6)*, Atlanta, Georgia, 1995.