Chapter VIII

The Role of Standards for Interoperating Information Systems

Wilhelm Hasselbring
INFOLAB, Tilburg University

INTRODUCTION

For integrating heterogeneous information systems, semantic interoperability is necessary to ensure that exchange of information makes sense — that the provider and requester of information have a common understanding of the 'meaning' of the requested services and data. Effective exchange of information between heterogeneous systems needs to be based on a common understanding of the transferred data. This paper discusses the role of domain-specific standards for managing semantic heterogeneity among dissimilar information sources. The *process* of integrating such heterogeneous information systems is also discussed in this context, whereby standards play a central role for 'initiating' *top-down* processes by means of defining common data models for the involved information sources.

BACKGROUND

Traditionally, the integration of heterogeneous information systems proceeds in a *bottom-up* process. Information stored in existing legacy systems is analyzed with respect to potential overlaps, whereby overlapping data in dissimilar systems describes the same or related information. The overlapping areas of related information sources are subsequently integrated. The integration is usually realized by means of mediators, federated database systems or such-like system architectures. Typical goals for the integration of existing systems are the development of global applications that access the data from multiple sources as well as consistency management of information that is stored in related systems.

Let us consider digital libraries as an example domain where the integration of existing information sources is one of the central problems to be solved (Schatz & Chen, 1996). As one result of a bottom-up integration of those existing information sources, the structure of the merged common data model (schema) is determined by the overlaps among the local data models, and not by the requirements of global applications. The maintenance of such integrated models is a problem, because those merged models rapidly become very complex; usually more complex than required for the actual integration goals. This situation can lead to severe scalability problems with respect to execution performance, usability and maintenance.

ISSUES, CONTROVERSIES AND PROBLEMS WITH THE TRADITIONAL BOTTOM-UP INTEGRATION PROCESS

Various approaches for integrating heterogeneous information systems — e.g., federated database systems or mediator and agent architectures — have been proposed (Hurson, Bright & Pakzad, 1993, Elmagarmid, Rusinkiewicz & Sheth, 1998, Sheth, 1998, Wiederhold, 1996, Jennings & Wooldridge, 1998). We illustrate the traditional bottom-up process of integrating such heterogeneous information systems, by means of schema integration in federated database systems (Sheth & Larson, 1990). A federated database system is an integration of autonomous database systems, where both local applications and global applications accessing multiple database systems are supported. For federated database systems, the traditional three-level schema architecture must be extended to support the dimensions of distribution, heterogeneity, and autonomy. The five-level-schema-architecture of Sheth & Larson (1990) is generally accepted as the basic structure for schema integration in federated database systems or at least for comparison with other architectures of specific federated database systems (Conrad, Eaglestone, Hasselbring, Roantree, Saltor, Schönhoff, Strässler & Vermeer 1997).

Figure 1 illustrates the bottom-up process for constructing the schema architecture in federated database systems which starts with an analysis of information stored in the local systems. To explain the schema types displayed in Figure 1): A *local* schema is the conceptual schema of a component database system, which is expressed in the (native) data model of that component database system. In a first step, the local schemas are translated into component schemas in the canonical data model of the federation layer. Then, export schemas are filtered from the component schemas and merged into a common federated schema. A *component* schema is a local schema transformed into the so-called *canonical* data model of the federation layer. The component, export and federated schemas are defined in this *canonical* data model. An *export* schema is derived from a component schema and defines an interface to the local data that is made available to the federation. A *federated* schema is the result of the integration of multiple export schemas, and thus provides a uniform interface for global applications. An *external* schema is a specific view on a federated schema or on a local schema, which serves as a specific interface for applications (local or global). To keep it simple, no external schemas are shown in Figure 1. For local applications, external schemas can be filtered from the local schemas. For global applications, external schemas are filtered from the federated schema.

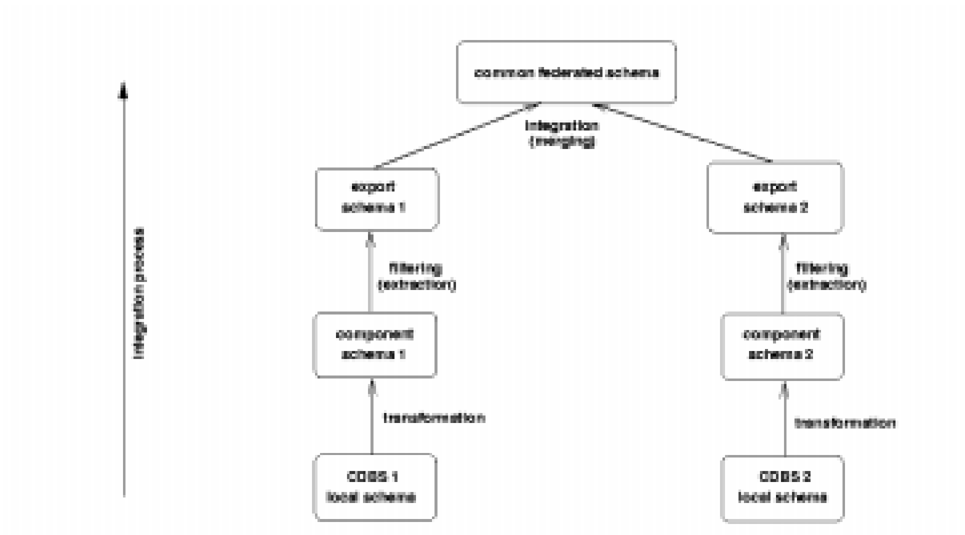

Figure 1: Bottom-up integration on the schema level. The arrows illustrate the development process that starts with the local data models of some existing legacy systems.

The translation from local to component schemas eliminates the data model heterogeneity among the various local schemas. Since the component and export schemas are provided in a canonical data model, no model translation is needed on the federation layer. However, many discrepancies between the export schemas may exist. For example, similar entities may be specified at different abstraction levels, or equivalent attributes may have different data types.

As one characteristic result obtained by a bottom-up construction of the schema architecture, the structure of the common federated schema is determined by the overlaps among the local schemas. Those schemas *overlap* when the intersections of the corresponding extensions are not empty. For instance, most object-oriented integration approaches resolve semantical overlappings by introducing generalized classes such that the original inheritance hierarchies are sub-hierarchies of the resulting merged hierarchy (upward-inheritance principle (Dayal & Hwang, 1984, Schrefl & Neuhold, 1988). These approaches take over the inheritance hierarchies from the local schemas to the integrated schema level and adapt them to each other. Consequently, the resulting merged hierarchies become needlessly complex.

We illustrate these problems by means of a small example for bottom-up integration of two digital library databases (this example is adopted from Schmitt & Saake (1998)). The first local database to be integrated is a library database storing information about publications, where books and journal papers are special types of publications (see Figure 2(a)). The second local database is a project publication database which stores information about publications related to a project. In the project database, books and technical reports are considered non-refereed publications (see Figure 2(b)).

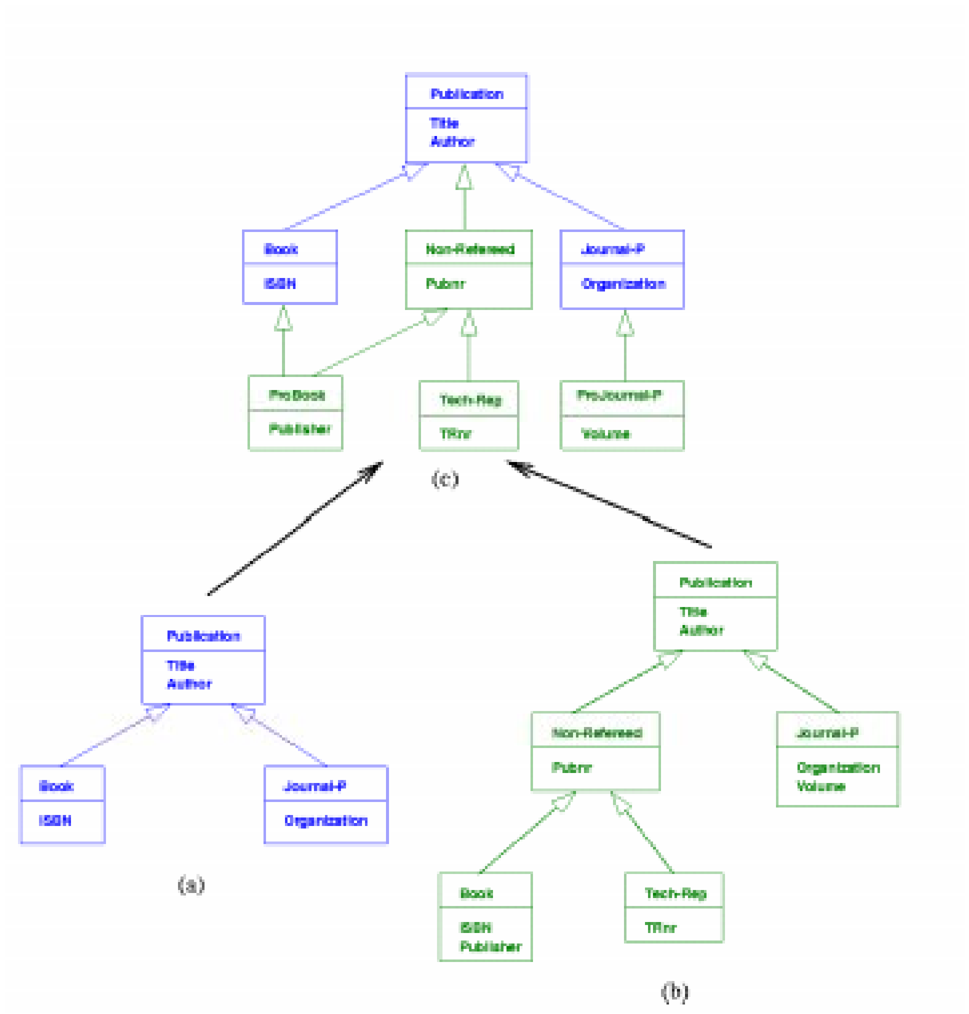

Figure 2: Local schemas of the library (a) and project publication (b) databases as well as the integrated schema as a result of merging (c). The schemas are modeled in the UML notation for class diagrams (Booch, Rumbaugh, & Jacobson 1999).

Integrating these local databases requires an analysis of the extensional overlappings among the classes to be integrated, whereby the *extension* of a class represents the set of *possible* objects. For this example, we assume that the Publication extensions of both databases overlap such that the corresponding merged class' extension represents the union of the local extensions. Some publications in the project publication database are published simultaneously as a technical report and as a journal paper. For simplicity, we assume that attribute conflicts are already resolved, i.e., attributes with identical names have the same semantics.

By means of a decomposition of the extensional overlappings into base extensions and their subsequent integration, an integrated schema (Figure 2(c)) with seven classes can be constructed, whereas the project publication database contains only five classes and the three classes for the library database are almost identical to a subset of the project's classes. Due to the upward-inheritance principle, the integrated schema contains (approximately) the sum of the classes contained in the individual component schemas. For reducing this complexity within the integrated schema, classes may be merged (Schmitt & Saake, 1998):

Vertical merging means merging a class with its direct superclass. In Figure 2(c), the classes Journal-P and ProJournal-P could be merged vertically. This vertical merging would change Volume into an *optional* attribute within the merged class (in the local databases all attributes are mandatory, i.e., NULL values are not allowed).

Horizontal merging means merging classes that have a direct specialization relation to the same superclass. In Figure 2(c), the classes Book and Non-Refereed could be merged horizontally. This horizontal merging would change both Pubnr and ISBN into *optional* attributes within the merged class; thus again introducing optional attributes. Furthermore, the additional integrity constraint would be introduced, which requires at least one attribute in the merged class holding a valid value.

To summarize: This small example illustrates the fact that with the traditional bottom-up process, merged class hierarchies are usually more complex than the integrated local schemas. Optimizing these merged hierarchies with respect to minimizing the number of classes may introduce new constraints; thus, complicating the common federated schema. These problems are largely due to the upward-inheritance principle, i.e., the original inheritance hierarchies are sub-hierarchies of the resulting merged hierarchy. New integrity constraints and optional attributes are often introduced.

Under those traditional integration paradigms (Batini, Lenzerini & Navathe, 1986), the integrated view/schema depends directly on the source schemas. An integration engineer defines the desired integrated view by examining all the existing systems to be integrated. The bottom-up approach is to sum up the capacity of existing information systems in one global model. As a result, the usability and maintainability of such integrated schemas can become a serious problem.

SOLUTIONS AND RECOMMENDATIONS

To approach a more 'ideal' top-down integration process, we start with a look at domain-specific software development before the top-down integration process is discussed in Subsection 4.2. A combined 'yo-yo' approach is discussed in Subsection 4.3. Appendix A presents a list of some domain-specific standards that are relevant for interoperability of information systems.

Domain engineering

Domain engineering is an activity for building reusable components, whereby the systematic creation of domain models and architectures is addressed. Domain engineering aims at supporting *application engineering* which uses the domain models and architectures to build concrete systems. The emphasis is on reuse and product lines. The Domain-Specific Software Architecture (DSSA) (Taylor, Tracz & Coglianese, 1995) engineering process was introduced to promote a clear distinction between domain and application requirements. A Domain-Specific Software Architecture consists of a domain model and a reference architecture as modeled in the blue part of Figure□3. The DSSA process consists of domain analysis, architecture modeling, design, and implementation stages as illustrated in Figure□4.

Domain models represent the set of requirements that are common to systems within a specific domain. Usually, those systems can be grouped into product lines (Dikel, Kane, Ornburn, Loftus, and Wilson, 1997), for instance, for the insurance or banking domain. There may be many domains, or areas of expertise, represented in a single product line and a single domain may span multiple product lines. *Domain analysis* is the process of identifying, collecting, organizing, and representing the relevant information in a domain, based upon the study of existing systems and their development histories and knowledge captured from domain experts. Figure□3 illustrates the relations between some roles (domain experts and application engineers) and the artifacts in the DSSA engineering process.

The *architecture* of a software system defines that system in terms of components and interactions / connections among those components. It is not the *design* of that system which is more detailed. The architecture shows the correspondence between the requirements and the constructed system, thereby providing some rationale for the design decisions (Shaw & Garlan, 1996). Reference architectures are the structures used to build systems in a product line. The domain model characterizes the *problem space*, while the *reference architecture* addresses the *solution space* (design). The *reference requirements* within the domain model define the (generic) functional requirements for applications in a domain.

In *application engineering*, a developer uses the domain models within the product line to understand the capabilities offered by the reference architecture and specifies a system for development. The developer then uses the reference architecture to build the system. An architectural model is developed in this phase from which detailed design and implementation can be done. Application engineers use the domain models with the users to elicit information about particular systems and to define the requirements for the planned software systems. By so doing, the models frame the userÕs needs in terms of existing models. Those needs not covered by a domain model are new requirements. Once a tentative set of features have been identified, the engineers analyze the interaction among the features to assess feasibility and identify additional requirements and contexts not described in a domain model. The domain engineers may choose to update a domain model with the new requirements. As indicated by the dashed arrows in Figure□4, various forms of feedback are possible.

Domain engineering and application engineering are complementary, interacting, parallel processes that comprise a reuse-oriented software production. An application engineering process should develop software systems from reusable components created by a domain engineering process (see Figure□4). The focus of application engineering is a single system whereas the focus of domain engineering is on multiple related systems within a domain. Typical application engineering

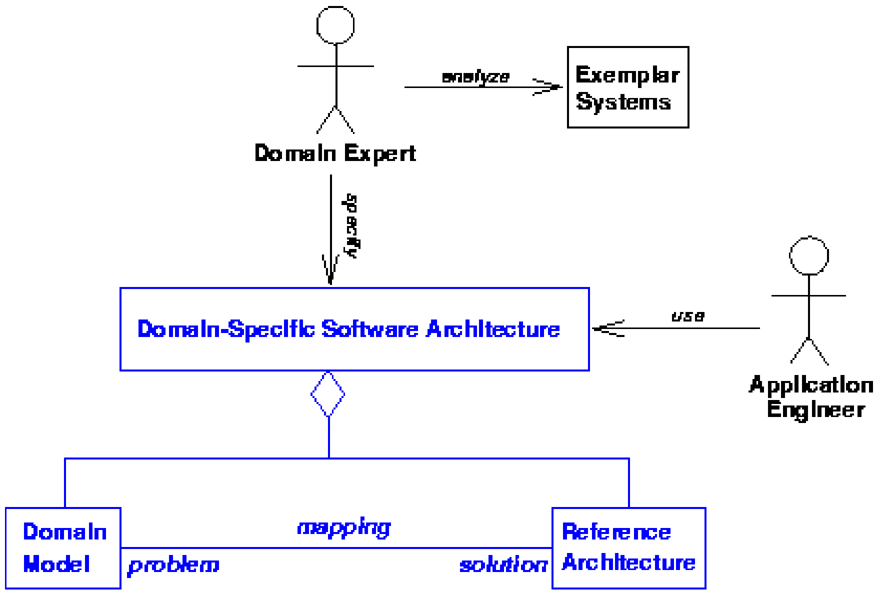

Figure 3: Relations between some roles and artifacts in the DSSA engineering process. Hollow diamonds indicate part-of relations. We use the UML notation for actors to model the roles (Booch et al., 1999).

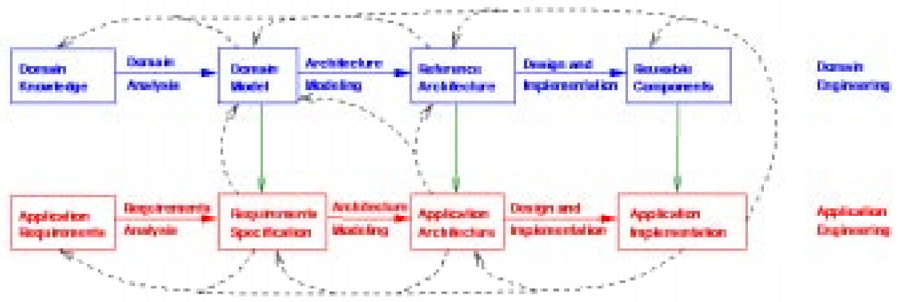

Figure 4: The DSSA engineering process. In application engineering, software systems are developed from reusable components created by a domain engineering process. As indicated by the dashed arrows, various forms of feedback are possible.

activities include using a domain model to identify customer requirements, and a (generic) reference architecture to specify an application architecture. Domain engineering emphasizes on the principle of *design-for-reuse* whereas application engineering should be based on the principle of *design-with-reuse*.

Top-Down Integration

Despite the fact that engineering of (new) global applications will usually require the integration of existing information sources, we argue that the integration process should proceed in a top-down way starting with the data models that are common to all the involved local systems, i.e. with domain models in the context of a DSSA engineering process. For such a *top-down* integration of those heterogeneous information systems, we propose the use of domain-specific standards as the basis for the common data models. Some relevant standards are listed in Appendix A.

Figure 5 illustrates the top-down process for constructing the schema architecture which starts with the common federated data schema that should be based on requirements of global applications and on relevant standards. The individual local schemas are integrated into this common schema via the component and export schemas. Export schemas are filtered from the common federated schema, instead of merging them into the common federated schema, as illustrated in Figure 5. Afterwards, these export schemas are mapped to the component schemas, which are transformed into the native data models of the local systems.

To 'hook' a local information system's model into a common (domain-specific) model, the responsive integration engineer has to understand his or her local information system and the corresponding domain model, but does *not* have to understand the other local information systems and the constraints that could be

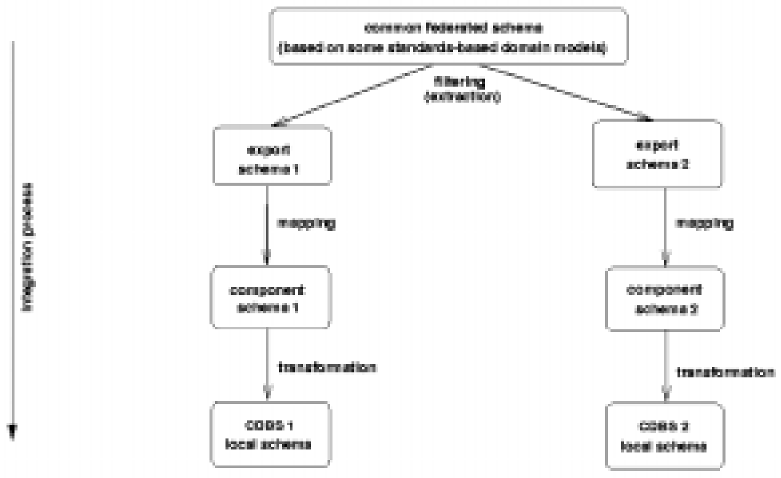

Figure 5: Top-down integration on the schema level. The arrows illustrate the development process that starts with the common standards-based data schema.

introduced by them into the common model using a bottom-up process. In the domain of digital libraries, for instance, Z39.50 and the Dublin Core can be the basis for the common models (see Appendix A). The approach is to define a common model based on the information we want to be in it. Then we map to the relevant bits of data in the local information systems. This should result in workable approach.

A Combined Yo-Yo Approach

In practice, we can also expect a *yo-yo* approach as illustrated in Figure 6: the integration process alternates with bottom-up and top-down steps. A top-down process can be expected to support global applications — such as workflow management or decision support functions to be supported by a data warehouse. A bottom-up process can be expected in the case that some local systems regularly need information from other systems — e.g., if one library system needs citation information from another library system.

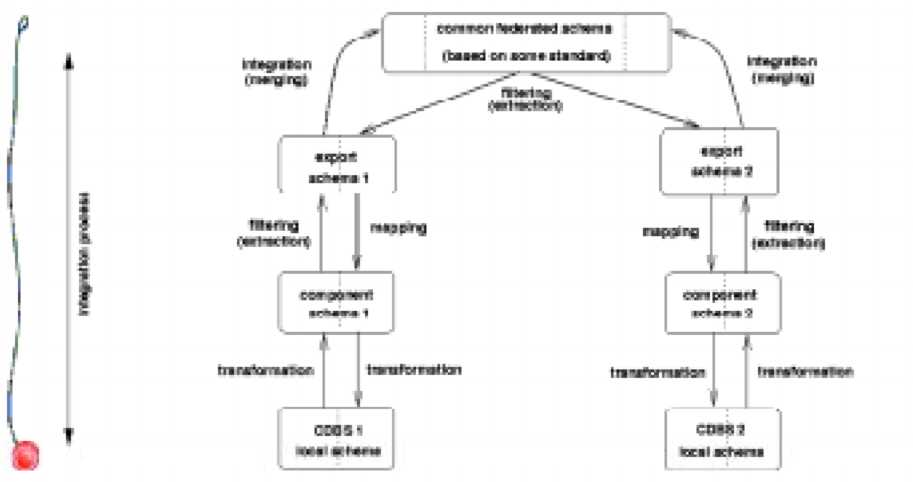

Figure 6: A combined yo-yo *approach to exploit the benefits of both strategies.*

The bottom-up process may provide input for extending the domain-specific standards as indicated in Figure 4. Then, a method for new information which resides in a local information system, but was not originally intended in the common model, is required. It should be easy to update the common model with such a 'yo-yo' approach. The evolution of common ontologies for mediation of information sharing is discussed in Kahng & McLeod (1998).

CONCLUSIONS

Some problems with the traditional bottom-up approach to the integration of heterogeneous information system have been discussed and a 'ideal' top-down approach using domain-specific standards to achieve more usable and scalable software architectures for heterogeneous information systems is proposed. The bottom-up process starts with an analysis of information stored in existing legacy systems to explore potential overlaps among information stored within the individual components. The top-down process starts with a common data model that should be based on domain-specific standards.

Enabling existing legacy systems to exchange information will typically start at the bottom and designing new global applications with access to several local systems will typically start at the top. Anyway, engineering of (new) global applications will usually require the integration of existing information sources. Despite this fact, we argue that the integration process should proceed in a top-down way starting with the data models that are common to all the involved local systems, i.e. with standards-based domain models. The combined *yo-yo* approach aims at exploiting the benefits of both strategies and to serve as a migration path form the traditional bottom-up approach towards an ideal top-down approach.

Usually, one result of the traditional bottom-up approach to the integration of heterogeneous information systems is that the structure of the common, merged models is determined by the overlaps among the local models. As opposed to the bottom-up approach, with the top-down approach the structure of the common model is *not* determined by the overlaps among the local models, but by the requirements of global applications and domain-specific standards, which can become the basis for *semantic* interoperability.

Particularly, the use of domain-specific standards as the basis for the common data model should alleviate the integration of commercial components that offer standards-compliant interfaces. For such a *top-down* integration of those heterogeneous information systems, we propose the use of domain-specific standards as the basis for the common data models. The use of such standards within the integration system also encourages a (smooth) migration towards modern standards-compliant systems. With standards-compliant interfaces, it becomes straightforward to 'hook' the local information of these subsystems into the common model. To be successful, it is obvious that both the standards and the local applications must be in the same domain. In any case, the selected domain-specific standards must cover the application domains of the integrated information systems. Only extracts of the standards will be used for actual applications. Common models should be restricted to specific domains (Missier, Rusinkiewicz & Silberschatz, 1995).

To quote Kleewein (1996) on practical issues with commercial use of federated databases:

> "Schema integration is one aspect of usability that impedes federation. There are often thousands of tables or views involved in a federation making maintenance of a global schema difficult."

Starting with the global schema and basing it on standards avoids changes to the fundamental structure of this schema making integration more usable and scalable. Only when interoperability is based on standards will this be realizable. Appropriate standards may enable interoperability, for instance Z39.50 and the Dublin Core in the domain of digital libraries. To quote Paepcke, Chang, Garcia-Molina & Winograd (1998) on the relevance of standards for interoperability for digital libraries worldwide:

> "An appropriate standard that is widely adhered to provides a powerful interoperability tool."

However, it is not easy to find those domain-specific standards that are detailed enough to allow for real interoperability. This is an important area for future work.

Appendix A: Some Domain-Specific Standards

This list of some relevant domain-specific standards is not meant to be comprehensive, but representative. The standards are grouped into domains and within groups the standards are listed in alphabetical order. References to corresponding web pages are provided. Exemplary, standards for the domains of digital libraries, healthcare, manufacturing, enterprise application integration, and electronic commerce are presented.

A.1 Digital Libraries

Dublin Core The Dublin Core is a metadata element set intended to describe electronic information sources. Originally conceived for author-generated description of Web resources, it has attracted the attention of formal resource description communities such as digital libraries.
More information is available at http://purl.org/dc/

Z39.50 The Z39.50 standard for searching library catalogs is a standard for the retrieval of bibliographic records independent of the type of system on which they are stored. The Z39.50 standard is an open systems protocol established and maintained by the library community. This standard is the basis for open systems connectivity among different library systems and information sources.
More information is available at http://lcweb.loc.gov/z3950/agency/

A.2 Healthcare

CEN TC 251 The scope of Working Group I (Information Models) of the Technical Committee 251 for Health Informatics within the European Committee for Standardization is the development of European standards to facilitate communication between independent information systems within and between organizations, for health related purposes, e.g., blood transfusion, physiology, pharmacology, psychiatry and nursing. The standards are based on information models - generic models of aspects of health care or health care information. The domain of this work are standards for electronic medical records and messages to meet specific healthcare business needs for the communication of healthcare information.
More information is available at http://www.centc251.org/

HL7 The Health Level 7 (HL7) protocol has been designed to standardize the data transfer within hospitals. It is an application level protocol and so relates to level seven of the ISO/OSI-protocol hierarchy. HL7 covers various aspects of data exchange in hospitals, e.g. admission, discharge and transfer of patients, as well

as the exchange of analysis and treatment data. The HL7 standard represents hospital related transactions as standardized messages. HL7 is a de-facto standard for data exchange between commercial systems for hospitals.
More information is available at http://www.mcis.duke.edu/standards/HL7/hl7.htm

A.3 Manufacturing

OMG Manufacturing DTF Within the standardization efforts of the Object Management Group (OMG) Domain Task Forces (DTF) several Business Object Facilities are standardized. The Manufacturing DTF is one of them.
More information is available at http://www.omg.org/

STEP The STEP Standard for Product Data Exchange is a data transfer standard which supports design, reuse, and data retention, and provides access to data across a productÕs entire life cycle. STEP is an actual file format (i.e., a common neutral format for exchanging CAD/CAM data among dissimilar systems).
More information is available at http://www.ukcic.org/step/ (retrieved 08-06-1999)

Enterprise Application Integration

OAGIS The Open Applications Group is a non-profit industry consortium of business-application software vendors. The Open Applications Group Integration Specification (OAGIS) defines inter-application integration scenarios, and details the content required for connecting the applications. The OAGIS encompasses interoperability specifications for front-office, back-office, and supply-chain business-software components.
More information is available at http://www.openapplications.org/

OMG Within the standardization efforts of the Object Management Group (OMG) Domain-Specific Task Forces several Business Object Facilities are standardized. The scope of the OMG's Financial Domain Task Force (CORBAfinancials Task Force) comprises financial services and accounting. The Insurance Working Group is part of the OMG's Financial Domain Task Force. The Insurance Working Group aims at developing domain-specific interfaces that will enable insurance companies and other financial institutions to leverage purchased componentry and integrate their data. Another area is addressed by the Manufacturing Domain Task Force, whose goals are interoperable manufacturing domain software components through CORBA technology.
More information is available at http://www.omg.org/

Electronic Commerce

With respect to the situation in the area of electronic commerce, we can quote Andrew Tannenbaum:

"The good thing about standards is that there are so many to choose from."

Obviously, a consolidation is required in this domain. Enterprise application integration is highly related to electronic commerce applications.

OMG Electronic Commerce DTF Within the standardization efforts of the Object Management Group (OMG) Domain Task Forces (DTF) several Business Object Facilities are standardized. The Electronic Commerce DTF is one of them.
More information is available at http://www.omg.org/

OBI The Open Buying on the Internet (OBI) Consortium is a non-profit organization dedicated to developing open standards for business-to-business Internet commerce. The OBI Consortium is an independent collaborative managed by CommerceNet.
More information is available at http://www.openbuy.org/
http://www.software.ibm.com/commerce/net.commerce/obi.html

OFX Open Financial Exchange (OFX) is a specification for the electronic exchange of financial data between financial institutions, business and consumers via the Internet.
More information is available at http://www.ofx.net/

OTP The Open Trading Protocol (OTP) was developed by a number of organizations, working co-operatively to make widespread Internet trading a convenient and secure reality. OTP is a protocol for the development of software products to permit product interoperability for the electronic purchase that is independent of the chosen payment mechanism. OTP encapsulates the payment with the offers/invoice/receipts for payment and delivery.
More information is available at http://www.otp.org/

PIP RosettaNet is an international organization dedicated to the adoption and deployment of open and common business interfaces in the IT industry. It defines Partner Interface Processes (PIP) to provide common business/data models and documents enabling system developers to implement RosettaNet eBusiness interfaces.
More information is available at http://www.rosettanet.org/

UN/EDIFACT This standard defines the United Nations rules for Electronic Data Interchange For Administration, Commerce and Transport, which comprises guidelines for the electronic interchange of structured data, and in particular that related to trade in goods and services between independent, computerized information systems.
More information is available at http://www.unece.org/trade/untdid/Welcome.html

REFERENCES

Batini, C., Lenzerini, M., & Navathe, S. (1986). A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4):323-364.

Booch, G., Rumbaugh, J., & Jacobson, I. (1999). *Unified Modeling Language User*

Guide. Object Technology Series. Addison-Wesley, Reading, MA.

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

Dayal, U. & Hwang, H.-Y. (1984). View Definition and Generalization for Database Integration in a Multidatabase System. *IEEE Transactions on Software Engineering*, 10(6):628-644.

Dikel, D., Kane, D., Ornburn, S., Loftus, W., & Wilson, J. (1997). Applying software product-line architecture. *Communications of the ACM*, 30(8):49-55.

Elmagarmid, A., Rusinkiewicz, M., & Sheth, A., editors (1998). *Management of Heterogenous and Autonomous Database Systems*. Morgan Kaufmann.

Hurson, A. R., Bright, M. W., & Pakzad, S. (1993).*Multidatabase Systems: An Advanced Solution for Global Information Sharing*. IEEE Computer Society Press.

Jennings, N. & Wooldridge, M., editors (1998). *Agent Technology: Foundations, Applications, and Markets*. Springer-Verlag.

Kahng, J. & McLeod, D. (1998). Dynamic classificational ontologies: Mediation of information sharing in cooperative federated database systems. In Papazoglou, M. and Schlageter, G., editors, *Cooperative Information Systems: Trends and Directions*, pages 179-203. Academic Press, San Diego.

Kleewein, J. (1996). Practical issues with commercial use of federated databases. In *Proc. 22th International Conference on Very Large Data Bases (VLDB'96)*, page 580, Bombay, India. Morgan Kaufmann.

Missier, P., Rusinkiewicz, M., & Silberschatz, A. (1995). Providing multidatabase access - an association approach. In *Proc. Sixth Workshop on Database Interoperability*.

Paepcke, A., Chang, C.-C., Garcia-Molina, H., & Winograd, T. (1998). Interoperability for digital libraries worldwide. *Communications of the ACM*, 41(4):33-43.

Schatz, B. & Chen, H. (1996). Building large-scale digital libraries. *IEEE Computer*, 29(5):22-26.

Schmitt, I. & Saake, G. (1998). Merging inheritance hierarchies for database integration. In Halper, M., editor, *Proc. Third IFCIS International Conference on Cooperative Information Systems (CoopIS'98)*, pages 322-331, New York City, NY. IEEE Computer Society Press.

Schrefl, M. & Neuhold, E. (1988). Object Class Definition by Generalization Using Upward Inheritance. In *Proceedings 4th International Conference on Data Engineering (ICDE'88)*, pages 4-13. IEEE Computer Society Press.

Shaw, M. & Garlan, D. (1996). *Software architecture: perspectives on an emerging discipline*. Prentice Hall.

Sheth, A. (1998). Changing focus on interoperability in information systems: From system, syntax, structure to semantics. In Goodchild, M., Egenhofer, M., Fegeas, R., and Kottman, C., editors, *Interoperating Geographic Information Systems*. Kluwer.

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

Taylor, R., Tracz, W., and Coglianese, L. (1995). Software development using domain-specific software architectures. *ACM SIGSOFT Software Engineering*

Notes, 20(5):27-38.

Wiederhold, G., editor (1996). *Intelligent Integration of Information*. Kluwer Academic Publishers, Boston.