Reengineering for Evolution of Distributed Information Systems

Holger Giese, Jörg P. Wadsack
Software Engineering Group, Department of Mathematics and Computer Science
University of Paderborn
Warburger Str. 100, D-33098 Paderborn
Germany
[hg|maroc]@upb.de

Keywords
System evolution, reengineering process, distributed information systems, middleware, schema integration, application integration, internet access

1 INTRODUCTION

The growing e-commerce market engender the need to open the existing, in local area networks utilized, information systems (IS) to the web. This requires the well understanding of the existing IS and makes *reengineering* a central aspect. However, most of todays reengineering activities and results address monolythic systems such as the centralized systems developed in the seventies. The required methods to reach the current state of reengineering for such systems were developed over several years. For the more recently developed distributed IS (DIS), the integration into the internet requires new adjusted reengineering techniques. The high complexity and the rapid evolution of DIS nowadays requires continous DIS reengineering (DISRE).

In this paper, we conceive evolution as any change in or of the DIS and emphasize the following three actions:

- *union* of multiple running (D)IS to a single DIS,
- *extention* of a DIS by a new part or
- *reduction* of a DIS by discard a part

Union and extension induces the idea of integration, whereas reduction entails "reverse integration" which can be seen as a variant of decomposition. In this context, we interprete integration and reverse integration in evolution of DIS with focus on a suitable middleware concepts.

In this paper, a systematic DISRE process based on an analysis of available integration concepts and technologies is developed. The application of existing and required new reengineering techniques in such a process is skeched.

The rest of the paper is structured as follows, in Section 2 we discuss the advantages and restrictions of the use of middleware of DIS. In the next section (Section 3), we propose a DISRE process, considering the trade of between existing and new (adjusted) reengineering techniques. Finally, in Section 4 we draw some conclusions.

2 MIDDLEWARE FOR DISTRIBUTED INFORMATION SYSTEMS

To clarify the architectural scenario we assume a three-tier model. On the left side of Figure 1, we have the schema layer of the model. We choose the term *schema* to describe the information (data) structure of our DIS. Furthermore, we select distributed databases (DDB) as more concrete case study, there the term schema fits better. The application layer contains all the functionality to manipulate the informations stored in the schemas. Finally, we have the WWW layer, which represents the interface to the real world, i.e. users.

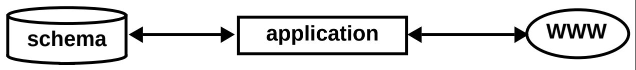

Figure 1 Three-tier model

Integration and support for evolution of distributed information systems by means of an additional middleware can be done in various ways. To structure and analyze the advantages and restrictions of different approaches we consider the two following extreme views: (1) *schema integration* (SI) which provides a consistent virtual schema located between the schema and application layer containing the integrated schemas; (2) *application integration* (AI) which results in an addition virtual application layer in-between the application and WWW layer.

In Figure 2 the SI approach for integration and evolution is presented. The different application specific schemas are integrated into an overall virtual schema (grey) which is served by a single homogenous distributed database management system. System evolution in form of an additional application therefore would result in an updated virtual schema and a new application (dashed box) . Batini et al [BLN86] present several methodologies for (database) SI. An approach to preserve semantic updates in SI is sketched in [VW94].

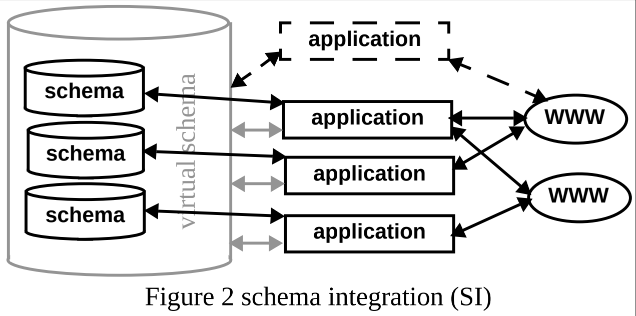

Figure 2 schema integration (SI)

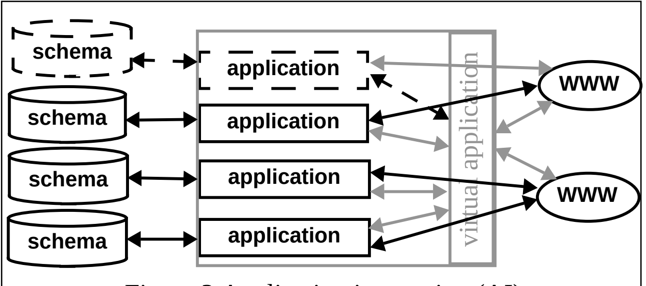

Figure 3 Application integration (AI)

The AI scenario in contrast would integrate an additional application without modifying the different schemas, cf. Figure 3. Instead, the virtual application layer (grey) is used to coordinate the applications as required. This additional application layer does further permit to use different heterogeneous and physically distributed database management systems when support for distributed transaction processing standards [XA94] is present. For example extending the DIS by a new application with own new schema, cf. dashed parts in Figure 3.

The required coordination however has to be realized by code in the virtual application layer. The identified AI strategy is strongly related to enterprise application integration (EAI) [Lin99] approaches, but we assume that the applications composition does not involve event processing as realized either by polling on a shared database or using specific application APIs.

The proposed middleware layer for application and information system integration has to be build on top of available middleware technology. The common solution for database integration is *distributed transaction processing* (DTP) [XA94] as provided by transaction monitors [Hud94, Hal96] and middleware transaction services [JTS99, OTS98]. Another more scalable solution is *reliable messaging* [Lew99, Hou98] which results in a reliable asynchronous processing scenario.

evolution schema	SI requires update for virtual schema
evolution application	AI requires update for virtual application
consistency	SI failures result in loss of data while inconsistencies are possible for AI failures
redundancy schema	SI has to exclude redundancy in the schema while AI control it at application level
redundancy application	SI does not address redundancy at the application level while AI can help to identify redundant functions which may be united in next evolution steps

Table 1: Evolution, consistency and redundancy

The AI approach requires the realization of coordination activities. A suitable solution is to employ available middleware technology. For the SI approach the same technology can be applied when the virtual schema layer is realized by code, however, also database technology like views can be used. Depending on whether a code or strict database solution has been chosen, the flexibility of the virtual layer permits variations to a great extent.

To compare both extreme concepts we further consider their impacts on evolution, consistency and redundancy in Table 1. The observations reveal that both strategies are rather contrary. While SI focuses on consistency and therefore try to exclude redundancy, AI avoids the integration of schemas and instead demand the proper coordination of the applications.

The specific aspects of distributed systems such as autonomy, heterogeneity and scalability are addressed in Table 2. The AI approach additionally permits integration of arbitrary *legacy systems*. When the virtual application layer provides the same interfaces as the applications the degree of transparency ensured by the SI strategy is also possible for a AI solution. In practice, the provided interface may include synthesized functionality and therefore a clear separation between the virtual application layer and added functionality is often difficult.

autonomy	AI results in still autonomous applications (services) while the virtual schema of SI results in a higher degree of coupling
heterogeneity	A code based layer can cope with datatype conversion problems in more flexible manner than a database layer
scalability	AI can exploit semantically independent application functionality to increase system scalability while SI is restricted to a single virtual schema and therefore has to employ more general approaches for scalability

Table 2: Distributed systems characteristics

The differences of both approaches can be further clarified by considering whether state or transitions of the resulting system are composed. While the SI approach provides a unique state (virtual schema) and relates all application specific schemas accordingly (mapping), the AI approach permits partial states (the application specific schemas) but ensures, that all via the application code initiated state, transitions are coordinated in an appropriate manner.

The goal of information system integration is the linking and interconnection of beforehand independent systems. The resulting coupling, however, also result in a less reliable system. When the organizational coexistence is acceptable or even required, the AI approach and even temporary inconsistency may be seen in favor. If in contrast data consistency has highest priority, SI has to be employed. The AI solution may include redundancy but ensures losslessness while the SI approach can result in a possible loss of data but ensures no-redundancy. Therefore depending on whether the reengineering activity is not considered as a closed job, both approach will be rated quite differently.

In practice, which approach fits best, varies for each fragment of the system and therefore a compromise which merges both strategies is often most valuable. Such a hybrid solution should exploit the advantages of both strategies for a system specific solution.

3 A PROCESS TO REENGINEER DISTRIBUTED INFORMATION SYSTEMS

The drawn conclusion in the previous section obliges us to consider SI as well as AI in parallel. Investigating all such possible hybrid integration scenarios would however be an impracticable task. For this reasons our process (depicted in Figure 4) starts with facile investigations.

In a first step, a (light) reverse engineering activity is applied to the application tier, i.e. we analyse the interfaces from the user view (step 1.1). The result is a simple directed call-graph from the basic user operations to the main parts of the schemas. In parallel, in step 1.2, a data structure reverse engineering step is done on the different schemas. This step is limited in a premier time in extracting entities and obvious relationships.

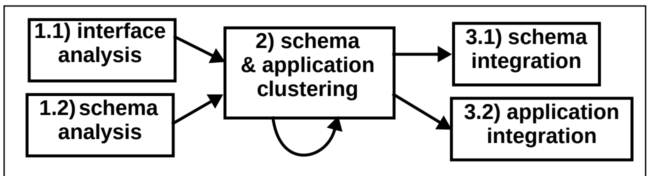

Figure 4 DISRE Process

Based on a comparison of the call-graphs and when the analysis of their offered functionality indicates a sharing between both applications, we check pairs of schemas on the degree of worth for their integration. This second process step provides us with a schema & application clustering. Here it might be necessary to have further infomation than the ones provided by the first steps. This leads on the one hand in step 2 to the need for iteration and on the other hand we have to provide techniques which allows more profound analyses. Those have to be program understanding as well as data structure recovering methods. The observed degree of schema compatibility permits to estimate the required integration effort. Thus, we can decide where SI is appropriate and where not.

To perform step 3.1, i.e. SI, we need the complete structure of the affected schemas. Two existing schemas reverse enginering approaches which adresses the problem of an entire structural schema recovery are *DB-Main* [EH99] and *Varlet* [Jah99]. Moreover, we have to recover relationships between the schemas. This inter-schema relationships have to be considered and can be use directly for the integration, i.e. the construction of a virtual schema. But for a complete and consistent integration we need information about data (schema) behaviour, which induces again more profound application understanding.

Alternatively, AI is done in step 3.2 of our process. The central problem is to identify all applications which are concerned by concrete user interactions. Furthermore, we have to cover all possible user interaction activities. This implies the need of full knowledge about the interfaces, because missing only one application for one user interaction leads automatically to partial inconsistencies. To ensure the completness of the reengineered information about the interfaces, we need, beside data structure (schema) and behaviour knowledge, information about the inter-schema relationships.

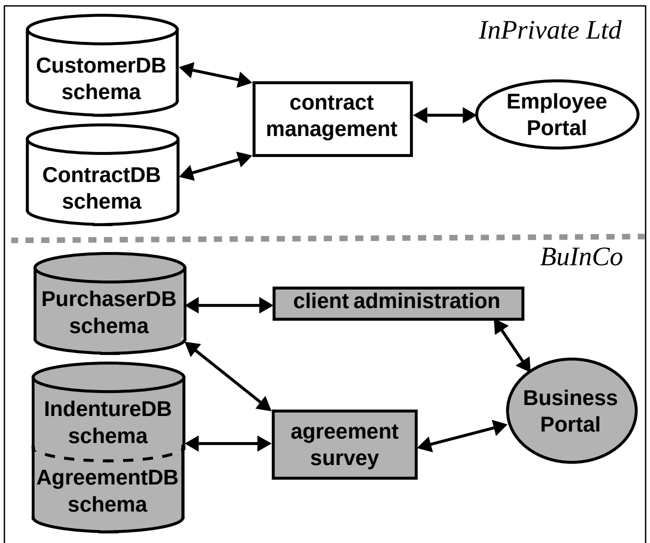

Figure 5 Example systems (simplified)

An early overview of reverse engineering techniques for program understanding is presented in [BMG+94]. Possibilities and requirements for tool interoprerability where recently discussed in Dagstuhl [EKM01].

According to the three layers of Figure 1, we propose an example to illustrate our approach. Assume, we have two insurance companies which want to fuse. The first is the *InPrivate Ltd* which only offers contracts for private insurances. Second, we have the *BuInCo* which only concludes agreements with business people and enterprises. Both companies operate DIS which should be integrated. Figure 5 depict a simplified exerpt of the two original DIS of the companies.

In the following, we apply our process to this insurance fusion example. This is of course a simplified view of the example due to lack of space. In Figure 6 we have two databases, CustomerDB and ContractDB, and an applications contract management for the *InPrivate Ltd*. *BuInCo* has also

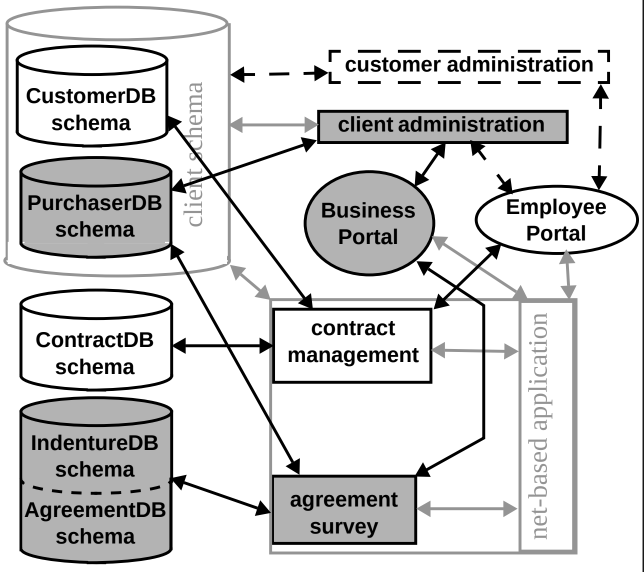

Figure 6 Hybrid integration (simplified)

two databases, PurchaserDB and IndentureDB / AgreementDB, and two applications the client administration and the agreement survey. The IndentureDB schema is for insurances for business people and the AgreementDB is to store information about company insurance agreements. In addition, we have two kind of portals, one for the company employees (EmployeePortal) and the other for business clients (BusinessPortal).

After performing step 1.1 and step 1.2 of our process, the comparison of the applications contract management and client administration indicates overlapping between databases CustomerDB and PurchaserDB. A more detailled analysis of the two schemas reveals large similarities in the manner of storing client information. This entails an integration of those two schemas to a client schema. Consequently, following Figure 2, the new client schema has to be adapted to the access from applications contract management and client administration.

Overlapping for private insurance contracts and agreements for buisness people is indicated by comparing applications contract management and agreement survey. Further investigations in the corresponding schemas reveal that an integration of the ContractDB schema and the IndentureDB schema could result in risks for the fusionned companies, e.g. a private person may benefits from buisness people advantages.

In contrast, the ContractDB schema and the company agreement schema parts of the AgreementDB schema are easily integrable, since they have no overlapping at all. But in this case integration makes little sense, because they would coexist and generate a superfluous management overhead. For the given example no further suitable schema pairs for SI can be identified.

AI makes sense for contract management and agreement survey because this two services are planned to be operated in a net-wide environment. A web access for client administration in contrast will be a considerable risks. For these reasons net-based application only encapsulates contract management and agreement survey.

Finally, for the case of an extension, we consider adding a customer administration application. The overlapping of functionality with the client administration indicates that SI employing the derived client schema is an appropriate solution. Note that the EmployeePortal is connected to the client administration application to ensure that the employees have the same access to the application as the business clients.

4 CONCLUSIONS

The proposed process to reengineer DIS and the guideline of their evolution should, beside the identified complexity for SI or AI, consider whether a later consolidation is planned. For the SI approach the later merging is rather straightforward based on the given combined virtual schema. For the AI strategy, however, such a merging requires that schema and application code have to be combined while no reuse for the integration efforts in form of the virtual application layer is guaranteed.

While the phenomena of distribution is more naturally covered by the AI approach, the SI strategy links the modules more tightly and therefore help to avoid serious problems with redundancy. Besides the discussed technical aspects, organizational structures and requirements are also relevant for an appropriate solution with respect to the degree of coupling. Therefore, whether SI or AI is reasonable, is not only a matter of technical feasibility.

REFERENCES

[BLN86] C. Batini, M. Lenzerini, and S. B. Navathe. *A Comparative Analysis of Methodologies for Database Schema Integration*. ACM Computing Surveys, 18(2):323–364, ACM Press, 1986.

[BMG+94] E. Buss, R. De Mori, W.M. Gentleman, J. Henshaw, H. Johnson, K. Kontogiannis, E. Merlo, H.A. Müller, J. Mylopoulos, S. Paul, A. Prakash, M. Stanley, S.R. Tilley, J. Troster, and K. Wong. *Investigating Reverse Engineering Technologies for the CAS Program Understanding Project*. IBM Systems Journal, 33(3):477–500, 1994.

[EH99] V. Englebert and J.-L. Hainaut. *DB-MAIN: A Next Generation Meta-CASE*. Journal of Information Systems - Special Issue on Meta-CASEs, 24(2):99–112, Elsevier Science Publishers B.V (North-Holland), 1999.

[EKM01] J. Ebert, K. Kontogiannis, and J. Mylopoulos, editors. *Interoperability of Reengineering Tools*, volume 296 of *Dagstuhl-Seminar-Report*. IBFI gem. GmbH, January 2001.

[Hal96] C. L. Hall. *Building Client/Server Applications Using TUXEDO*. John Wiley & Sons, Inc., 1996.

[Hou98] P. Houston. *Building Distributed Applications with Message Queuing Middleware*. Microsoft Cooperation, 1998.

[Hud94] E. S. Hudders. *CICS: A Guide to Internal Strucure*. John Wiley & Sons, Inc., 1994.

[Jah99] J.H. Jahnke. *Management of Uncertainty and Inconsistency in Database Reengineering Processes*. PhD thesis, University of Paderborn, Paderborn, Germany, September 1999.

[JTS99] JTS. *Java Transaction Service (JTS)*. Sun Microsystems Inc., December 1999. Version 1.0.

[Lew99] R. Lewis. *Advanced Messaging Applications with MSMQ and MQSeries*. Que, 1999.

[Lin99] D. Linthicum. *Enterprise Application Integration*. Addison-Wesley, 1999.

[OTS98] OTS. *Transaction Service Specification*. Object Management Group, February 1998.

[VW94] V.M.P. Vidal and M. Winslett. *Preserving update semantics in schema integration*. In Proc. of the 3rd International Conference on Information and Knowledge Management, Gaithersburg, Maryland, pages 263–271. ACM Press, November 1994.

[XA94] XA. *Distributed Transaction processing: The XA+ Specification, version 2*. X/Open Group, 1994. X/Open Company, ISBN 1-85912-046-6, Reading, UK.