# Supporting Autonomy for Information Systems in a Changing Environment

**J. Kusch** and **G. Saake**
Institut für Technische Informationssysteme, Abteilung Datenbanken,
Otto-von-Guericke-Universität Magdeburg, D–39106 Magdeburg, Germany,
E-Mail: {kusch|saake}@iti.cs.uni-magdeburg.de

## Abstract

Availability and scalability are important features of information systems. To gain this kind of requirements, we propose a distributed schema catalog in conjunction with appropriate development phases. The distributed schema catalog has to support distribution transparency including partitioning and replication to be flexible for changes within organization structures. Additionally, phases of autonomy design have to appoint the adequate usage of distribution transparency aspects to enable execution autonomy by a minimum of replication.

This paper only gives a brief overview concerning the development of distributed information systems based on object-oriented structures.

## 1 Motivation

In the development of information systems, object-oriented specification (e.g. TROLL [Jungclaus *et al.*, 1995]) is useful for conceptually modeling of the universe of discourse. Viewing an information system as a collection of communicating objects is close to the intuitive perception of such systems on a conceptual level. A uniform lifecycle model of objects (or agents) covers the description of structural and behavioral aspects. Nowadays complex information systems (e.g. knowledge bases) are an integral part of organizations [Fasnacht, 1993]. In order to be conducive for interconnecting departments, information systems have to be flexible to accommodate for changes in organization structures. Mandatory features are *distribution* for decentralization support and *scalability* for modular system increase [Simon, 1995]. Furtheron processing in parallel enables an important *speedup* [Gray, 1995]. Thus, the task is to map the global conceptual model to computational reality as a distributed infrastructure. However the advantages of distribution are only usable by an adequate support of software.

Object-oriented specification consists of a global abstract description as conceptual model without non-functional requirements (e.g. distribution, persistence, exception handling). A distributed infrastructure consists of a set of inter-connected loosely coupled nodes with own processors and disk-spaces. *On which 'level' distribution has to be combined with the conceptual model?* A complex information system structure, which is described as conceptual entirety, is transparently distributed over several nodes. Important is a partial use of the information system, although not all nodes are available or reachable due to a site failure. *How to achieve a maximum of node autonomy for a distributed system (C. J. Dates first (of 12) rule for distributed database systems [Date, 1990]) supported by the distributed schema catalog?*

Developing highly available information systems, we propose a *distributed schema catalog* in conjunction with *autonomy design phases*:

**Distributed schema catalog.** In general, distributed database systems have to deal with a lot of *tradeoffs* [Rahm, 1994; Bell and Grimson, 1992; Özsu and Valduriez, 1991], e.g. data replication versus data transfer, reuse versus autonomy, transparency versus efficiency. Our goals in gaining distribution for an object-oriented data model are:

- *Distribution transparency*: For scalability support transparency of location and migration enables objects to be used without knowledge of their location and movement of objects within a system without affecting the operations [Herbert, 1989]. In conjunction with horizontal and vertical class partitioning, the system could be expanded in scale without changing the specification or the references.
- *Execution autonomy*: For decentralization support, neither federation does interfere with local (or subsystem) operations nor any knowledge of the federation is needed for performing local (or subsystem) operations [Kalathil and Belford, 1994; Veijalainen and Popescu-Zeletin, 1988].

Existing approaches in the area of database systems mostly do not pay enough attention to the opportunities of autonomy and scalability, which are getting increasingly important by a new generation of *parallel hardware clusters* [Gray, 1995].

Additionally design phases are mandatory to control the transparency aspects of distribution with the view to execution autonomy.

**Autonomy design phases.** Multiple allocation of data within a distributed environment promises an increase of performance and availability and a decrease of communication. Contrary to this disadvantages are memory consumption and consistency maintenance in case of a site failure. Thus we aspire execution autonomy enabled by a minimum of replication, which has to be guaranteed by an extended development process:

- In *former development phases* autonomy modules have to be modeled conceptually, which are based on informal autonomy requirements.
- In *later development phases* the initial distribution structure has to be appointed, which is based on the modeled autonomy modules. Altogether this effects the transparency aspects of distribution, e.g. object class location and partitioning and object location and replication.

Evolution requirements of the distribution structure are mostly not equal to those of the conceptual model. Thus the additional autonomy design phases has to be performed independently from the remaining development phases.

To focus our approach within the area of object-oriented specification and distributed databases, this work is based on a homogeneous integrated schema and covers only *structural aspects* including integrity constraints.

## 2 Object-Oriented Structures

Conceptual modeling of information systems requires the description of the application domain, the so-called *universe of discourse*, on a high abstraction level. Looking at an information system and its environment as a collection of interacting *objects* seems to be a very natural way for conceptualizing information structures and processes. Objects have a local state, show a specific behavior, communicate with other objects and may be themselves composed from smaller objects. This observation is confirmed by the current success of object-oriented analysis and design frameworks, e.g. [Rumbaugh *et al.*, 1991].

This paper emphasizes only the *structural aspects* of the object model as base of data maintenance. Thus, execution autonomy refers to data model operations, e.g. creation, deletion, migration of objects and object classes. Behavioral features, e.g. processes, synchronization and transactions, are not regarded. Mandatory structural features of object-oriented database systems and information systems as pointed out in i.e. [Jungclaus *et al.*, 1995; Cattel, 1994; Ahmed *et al.*, 1991; Rumbaugh *et al.*, 1991; Atkinson *et al.*, 1990] are

- *class types*,
- *objects* with a global, immutable and system wide unique *object identity*,
- *object classes*,
- *specialized classes* supporting semantic inheritance [Saake, 1993],
- *component relations* to model complex objects and
- object preserving *views*.

These abstractions of the conceptual model are grouped into an object base. Thus, each object base contains a set of classes, which consist of a class type and a set of objects.

With the view to a later implementation, the chosen abstractions gain a "small is beautiful" object model.

## 3 Distributed Schema Catalog

Supporting autonomy for information systems in a changing environment, a schema catalog is introduced which enables *distribution transparency*, and in conjunction with internal structures an increase of availability. To characterize the distributed schema catalog, the aspects *data definition interface*, *data-logical architecture* and *meta schema* are briefly represented.

**Data definition interface.** Trends about information systems and knowledge bases point out the everlasting complexity increase for data processing [Hwang and Briggs, 1989]. Representing complex data in an adequate way, a variety of abstractions are needed. To take some of the load off the development phases, the data definition language of a distributed schema catalog has to support the abstractions of the conceptual model, i.e. object classes, specialized classes, component relations, and views.

This has an effect on the development phases, which are discussed in section 4 in detail. The development process gets simplified, since there is no transformation necessary between the conceptual model and the phase of implementation. Furtheron the phase of distribution design is superfluous in early stages of development. Considering changing environments, aspects of distribution, i.e. location, partitioning and replication, are not statically decidable in advance. Due to distribution transparency, the phase of distribution design could be displaced as a later *distribution tuning phase*, dependent of the data access profile. This strategy seems to be more adequate for the design of distributed information systems.

**Data-logical architecture.** Base of the distributed schema catalog is a distributed infrastructure, consisting of a graph of nodes and inter-node-connections. Each node contains a set of instance-buffers for persistently maintaining data. The inter-node-connections are established via *broadcast channels* to neglect network partitioning problems.

Due to the increase of hardware performance and complexity, the schema catalog has to perform the mapping from the conceptual specification structure to the distributed infrastructure. To support execution autonomy and scalability within a changing environment, location and partitioning of object classes and location and replication of objects have to be archived in an transparent way. Object classes

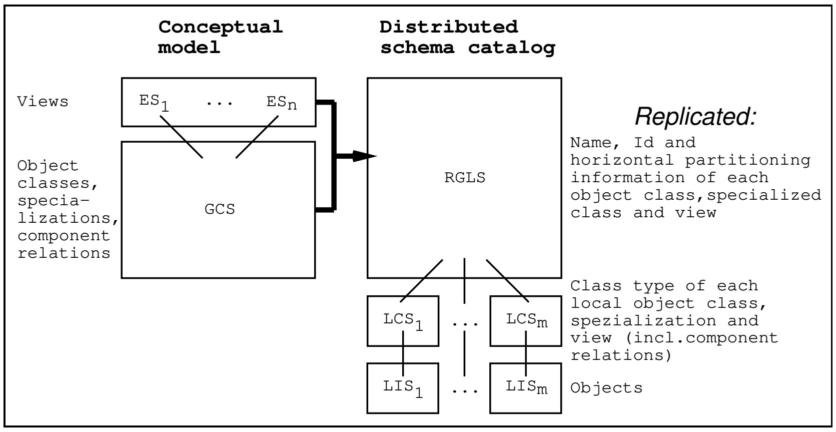


Figure 1: Data-logical Architecture of the Distributed Schema Catalog

should be located to a set of nodes (horizontal partitioning). Objects of specialized classes should be located to a set of instance-buffers of different nodes (vertical partitioning). The consideration of object encapsulation leads to the fact, that specialization is the only way of vertically partitioning objects. An object should be located to several nodes within a horizontally partitioned object class (replication).

As architecture of the distributed schema catalog (figure 1) our approach proposes a *replicated global location schema* (RGLS) with distributed local conceptual schemas (LCS's) and appropriate local internal schemas (LIS's). This depicts a specialization of the ANSI-4-level-architecture. The RGLS, which is replicated to each node, contains the name, identification and horizontal partitioning information of each object class. As a *virtual global schema*, the RGLS offers with respect to autonomy minimal global knowledge for the access of remote information. The LCS's contain class type information of each object class, which is replicated to those sites, where parts of their extension are located. Thus, objects are co-located with their types. At least the LIS's contain objects, which are maintained via a set of instance-buffers.

Further internal details, which are mandatory for scalability and execution autonomy are briefly depicted:

- Object identifiers are built as *compound object identifier* of a class Id and a class internal Id. Class internal Id's are maintained via areas of free Id's.
- Relations between and within object classes are maintained through object identifier references, which implicitly contain location information (via class information).
- Replication of key attributes has to be maintained implicitly.
- If a failed node gets online, several complex data updating operations have to be performed on the whole object base.

This allows to perform consistency preserving operations even if nodes in the context of this operations are offline.

Access to object properties is always directed to the RGLS, which determines a path over a hierarchical structure of LCS's and LIS's. In conjunction with compound object identifiers *location transparency* is enabled.

**Meta schema.** The conceptual model has to be free of non-functional requirements. For orthogonally influencing distribution, a meta schema is introduced, which is modeled as a reflexive system [Maes, 1988] exclusively with the selected abstractions of the object model. To maintain transparency aspects, the meta schema is modeled as *a set of vertically partitioned classes* for each node of the distributed infrastructure, specialized from *a totally horizontally partitioned class* with replicated objects. Replicated objects contain replicated schema information of horizontally partitioned classes, whereas the specialized parts are managing node related class information (i.e. only local objects of an object class and related instance-buffers). This meta schema architecture enables a sound maintenance of horizontally partitioned classes and replicated objects. Distributed administration is performed by the events of the meta schema.

## 4 Autonomy Design

Nowadays information systems are based on distributed infrastructures to manage the requirements of the users. To take advantage of a decentralizable platform several existing disadvantages [Rahm, 1994; Bell and Grimson, 1992; Özsu and Valduriez, 1991] of distributed database systems have to be regarded. Due to the possibility of node failures (e.g. power failure, hardware- or software failure, installation or maintenance tasks or user control failure) autonomy considerations have to be taken into account. This should save failure costs of the whole information system.

Developing highly available information systems, the development process has to be extended. Base of the development process is the presented distributed

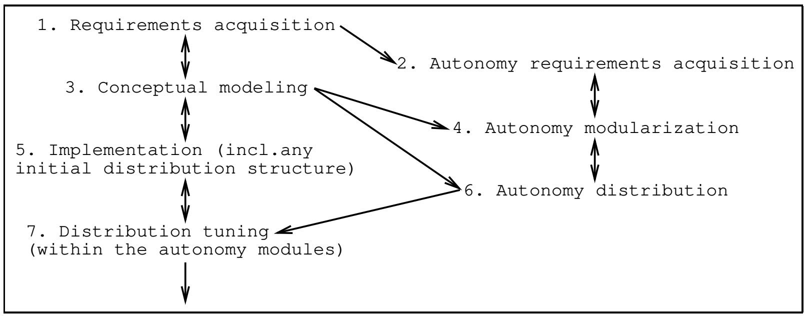


Figure 2: Phases of Autonomy Design

schema catalog which supports location, partitioning and replication transparency. Additional design phases for information systems are mandatory to control the transparency aspects of distribution, and thus serve the availability requirements by a minimum of data replication. Results of these additional design phases are the initial location including partitioning of object classes and the initial location and replication of objects.

We propose the following phases of design (figure 2) to develop highly available information systems:

1. *Requirements acquisition*: Informal description of the universe of discourse.
2. *Autonomy requirements acquisition* (based on requirements acquisition): Informal description of availability requirements within the organization structure.
3. *Conceptual modeling* (based on requirements acquisition): Formal specification of a global object model.
4. *Autonomy modularization* (based on autonomy requirements acquisition and conceptual modeling): Formal specification of a set of autonomy modules which represent autonomously maintainable areas of organization structures. Each autonomy module consists of a set of object classes, and a set of related nodes, on which this classes have to be at least located.
5. *Implementation* (based on conceptual modeling): Due to the abstraction level of the data definition interface, the conceptual model could be directly implemented. The replicated global location schema (RGLS) of the distributed schema catalog offers a "virtual global schema" to each node, independently from the initial location of the implementation.
6. *Autonomy distribution* (based on conceptual modeling and autonomy modularization): Autonomy conflicts arise through *autonomy module overlapping relations* between abstractions of the object model, i.e. specified by integrity constraints, component and specialization relations. Thus, an algorithm, which cannot be presented here in detail, automatically generates *a set of locations for each object class* (as horizontal partitioning), which is the base of mandatory object replication. Additionally *a set of birth events for objects classes* is generated to control location and replication of created objects. Altogether, a distribution structure is generated, which achieves execution autonomy by a minimum of object replication.
7. *Distribution tuning* (based on implementation and autonomy distribution): Dependent on later data access, the distribution structure within the autonomy modules could be optimized.

Phases 1, 3, 5 and 7 (figure 2) enable an evolutionary development strategy, likewise the phases 2, 4 and 6. Due to the different evolution requirements of the conceptual model and the distribution structure, the phases of autonomy design could be performed independently.

## 5 Outlook

The presented *specification language independent* work depicts fundamentals for the development of highly available and scalable information systems.

Object-oriented specification integrates structural and behavioral aspects of modeling. Thus one important enhancement of our approach is the consideration of *object behavior* with the view of autonomy and parallelism. Here, transactions and commit protocols within distributed systems have to be taken into account. To support implicit parallelism, we propose asynchronous communication with implicit synchronization. This could be performed by dataflow driven data evaluation [Lee and Hurson, 1994] via pipelining.

The disadvantages of optimization within our approach, which are evoked by transparent distribution, could be improved by dynamic query optimization. *Query objects* [Kusch, 1994; Jungclaus *et al.*, 1991] with internal knowledge of the state of the infrastructure are a first approach for this problem.

## References

[Ahmed *et al.*, 1991] S. Ahmed, A. Wong, D. Sriram, and R. Logcher. A Comparison of Object-Oriented Database Management Systems for Engineering Applications. Technical report, Massachusets Institute of Technology, Department of Civil Engineering, 1991.

[Atkinson *et al.*, 1990] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, and S. Zdonik. The Object-Oriented Database System Manifesto. In W. Kim, J.-M. Nicolas, and S. Nishio, editors, *Deductive and Object-Oriented Databases*, pages 223–240. North Holland, 1990.

[Bell and Grimson, 1992] D. Bell and J. Grimson. *Distributed Database Systems*. Addison-Wesley, 1992.

[Cattel, 1994] R. G. G. Cattel. *The Object Database Standard: ODMG-93*. Morgan Kaufmann, 1994.

[Date, 1990] C. J. Date. *An Introduction to Database Systems, 5th Edition*. Addison-Wesley, 1990.

[Fasnacht, 1993] D. Fasnacht. *Koordination verteilter und heterogener Datenbanksysteme*. Verlag Josef Eul, 1993.

[Gray, 1995] J. Gray. Super-Servers: Commodity Computer Clusters Pose a Software Challenge. In G. Lausen, editor, *Datenbanksysteme in Büro, Technik und Wissenschaft*, pages 30–47. Springer-Verlag, 1995.

[Herbert, 1989] A. Herbert. *The ANSA reference manual*. Cambridge, U.K.: Architecture Projects Management Limited, 1989.

[Hwang and Briggs, 1989] K. Hwang and F. A. Briggs. *Computer Architecture and Parallel Processing*. McGraw-Hill, 1989.

[Jungclaus *et al.*, 1991] R. Jungclaus, G. Saake, and C. Sernadas. Using Active Objects for Query Processing. In R. Meersman, W. Kent, and S. Khosla, editors, *Object-Oriented Databases: Analysis, Design and Construction (Proc. of the 4th IFIP WG 2.6 Working Conf. DS-4, Windermere (UK), 1990)*, pages 285–304. North-Holland, Amsterdam, 1991.

[Jungclaus *et al.*, 1995] R. Jungclaus, G. Saake, T. Hartmann, and C. Sernadas. TROLL – A Language for Object-Oriented Specification of Information Systems. *ACM Transactions on Information Systems*, 1995. *To appear.*

[Kalathil and Belford, 1994] B. J. Kalathil and G. G. Belford. Supporting Local Autonomy in a Distributed Object-Oriented Database. In T. Özsu, U. Dayal, and P. Valduriez, editors, *Distributed Object Management*, pages 347–352. Morgan Kaufmann Publishers, 1994.

[Kusch, 1994] J. Kusch. Ein Ansatz zur Operationalisierung deskriptiver Anfragen durch Anfrageobjekte. In S. Conrad, P. Löhr, and G. Saake, editors, *"Grundlagen von Datenbanken"*, pages 92–96. Otto-von-Guericke-Universität Magdeburg, Institut für Technische Informationssysteme, Bericht 94-01, 1994.

[Lee and Hurson, 1994] B. Lee and A. R. Hurson. Dataflow Architectures and Multithreading. *IEEE Computer*, 27(8):27–39, 8 1994.

[Maes, 1988] P. Maes. *Meta-Level Architectures and Reflection*. Elsevier Science Publishers B. V., 1988.

[Özsu and Valduriez, 1991] M. T. Özsu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, 1991.

[Rahm, 1994] E. Rahm. *Mehrrechner-Datenbanksysteme*. Addison-Wesley, 1994.

[Rumbaugh *et al.*, 1991] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-oriented modeling and design*. Prentice-Hall, 1991.

[Saake, 1993] G. Saake. *Objektorientierte Spezifikation von Informationssystemen*. Teubner, Stuttgart/Leipzig, 1993.

[Simon, 1995] A. R. Simon. *Strategic Database Technology: Management for the year 2000*. Morgan Kaufmann Publishers, Inc., 1995.

[Veijaleinen and Popescu-Zeletin, 1988] J. Veijaleinen and R. Popescu-Zeletin. Multidatabase systems in ISO/OSI environments. In N. Malagardis and T. Williams, editors, *Standards in Information Technology and Industrial Control*, pages 83–97. North-Holland, 1988.