The Network as a Global Database: Challenges of Interoperability, Proactivity, Interactiveness, Legacy

Peter C. Lockemann, Ulrike Kölsch, Arne Koschel,
Ralf Kramer, Ralf Nikolai, Mechtild Wallrath, Hans-Dirk Walter
{lockemann|koelsch|koschel|kramer|nikolai|wallrath}@fzi.de

Fakultät für Informatik
Universität Karlsruhe
D-76128 Karlsruhe, Germany

Forschungszentrum Informatik (FZI)
Haid-und-Neu-Str. 10-14
D-76131 Karlsruhe, Germany

Abstract

The current integrated developments in network and computing give rise to a technical infrastructure for the information society which one may variously circumscribe by terms such as *ubiquitous computing*, *telepresence* and *the network as one giant global database*. The paper applies to the network the metaphor of global database, and subsumes the aspects of ubiquity and telepresence under it. It should then be possible to preserve many of the existing database techniques and to concentrate on adjusting these to the network information infrastructure.

The paper explores four challenges for adjustment: interoperability due to heterogeneous data repositories, proactivity due to autonomy of data sources, interactiveness due to the need of short-term and task-specific interaction and cooperation, and legacy due to the fitting of old systems to the networked environment. Based on several application projects and exemplary solutions, the paper claims as its experiences that object-orientation provides a natural framework for meeting the challenges, but must also draw on the combined resources of databases, data communications, and software engineering.

Proceedings of the 23rd VLDB Conference
Athens, Greece, 1997

1 Introduction

Modern information and communication technology is the driving force behind the future highly interconnected society. This raises high challenges to those that are entrusted with the technical infrastructure for society, and it places high responsibilities on them. To meet these, one should start from a global view on this infrastructure. To develop a consensus on such views the use of metaphors is commonplace. For the networked infrastructure three metaphors are usually mentioned, each corresponding to a different angle on it. One is ubiquitous computing which refers to the utilization of computing power of whatever kind whenever and wherever the need arises. Another is telepresence (or in the extreme, omnipresence) - the ability to be present anyplace without physically moving there, and to be at several places simultaneously. Telepresence is based on telecommunications that is no longer point-to-point and person-to-person, but may involve person-to-group or group-to-group relationships which open up many new avenues of creative collaboration and problem-solving. A third angle is the view of the network as one giant global database indicating that information becomes a shared commodity with equal access by many or all, instant reading and posting of items from anywhere by anyone.

All three angles appear equally proper and, hence, seem to indicate that only an interdisciplinary approach will ultimately be able to meet the challenges of the network information infrastructure. On the other hand, in any given situation one will have to emphasize one of them which will then determine the predominant technology under which one plans to strike out at the technical and organizational problems for the network.

This paper chooses the metaphor of global database as its primary perspective. For one, this seems a natural choice because databases pervade the entire fabric of an organization, as they do in industry, commerce, and public administration. For another, it seems a technically sound choice because databases can be considered a mature and reliable technology, though not necssarily in a network of thousands of loosely coupled, heterogeneous nodes. The purpose of the paper is to explore, on the basis of a number of case studies for practical applications, whether the choice is indeed a productive and constructive one, and if so, what adjustments are needed to the traditional database techniques, with which well-understood techniques from other disciplines they should be amalgamated and how this is to be done.

Such an exploration requires a framework on which all the disciplines involved can agree. We claim that object-orientation is such a framework (Section 2). We then examine a number of challenges to database and other technologies. One has to do with the technical, semantic and pragmatic heterogeneity of data repositories which must be overcome by suitable means for interoperability (Section 3). Another has to do with the autonomy of data sources which suggests that databases no longer just respond when a particular service is requested from them but become active on their own, requesting services from other network components. The paper illustrates how proactivity can be added to a system of distributed databases (Section 4). A third challenge is how to exploit the potential for collaborative behavior by providing the means for concerted action of database nodes in order to solve common tasks (Section 5). Section 6 addresses the challenge of how to preserve earlier investments and nonetheless fit legacy systems to the networked environment.

2 Object orientation

Object-orientation is a technical framework that gives cohesion to a wide range of information technological disciplines and, hence, appears as an ideal framework to meet the interdisciplinary challenges of the infrastructure for the networked society. The reasons are manifold. Objects allow any measure of compartmentalization and, together with encapsulation, the concentration of functionality along a service-oriented rationale and of information along a need-to-know principle. Objects interact by exchanging messages. Objects may adapt their responsiveness without changing their interface to the surrounding world. Objects may be replaced by "better" objects without disturbing their surroundings as long as their interface remains the same.

A more recent rationale for object-orientation is its closeness to the technical metaphor of society of agents which originated from distributed Artificial Intelligence but has another more technical root in telecommunications. Agents are software components that seem to assume a life of their own, respond to requests in some autonomous fashion that includes decision-making capabilities, become proactive by initiating actions seemingly without outside intervention, adapt to their environment by exhibiting learning capabilities, enter into contracts with one another if there is a need to interact, and may even move through the network to where the action is [CCWB94, Pog95, WJ95]. All these are properties that we must expect from the nodes in the network. There are indications that objects are a promising implementation concept for agents [BGK+95, GKT95]. In particular, if one encapsulates within each object its own process one obtains a basis for the desired autonomous, proactive and interactive capabilities. Hence, whereas in the traditional approaches to object orientation objects and processes are treated as orthogonal notions, the two become tightly interconnected in a networked world. Since independent processes may run concurrently, cooperation between objects in a network implies cooperation among processes.

In summary, object-orientation seems a particularly useful technical framework for the network infrastructure for modern society. At the same time, since object-orientation is also a concept underlying many modern developments in datatabases, particularly in the form of object-oriented database systems, it provides a suitable foundation for the view of the network as a global database.

3 Interoperability

Interoperability refers to the capability of independently developed objects to productively interact across a network of heterogeneous platforms and services. Interoperability has three aspects:

- Platform interoperability overcomes the heterogeneity of the hardware and system software in the nodes of the network, and of the services that transport messages between the nodes.

- Notational interoperability surmounts disagreements among the objects (application programs, tools, databases) on the structure, representation or interpretation of the data (denotation) they wish to exchange, which often reflect differences in universes of discourse, perceptions, attitudes, and goals (connotation). Typical effects are different or even incompatible functional interfaces, data models, data types, database schemas, terminologies, and data formats.

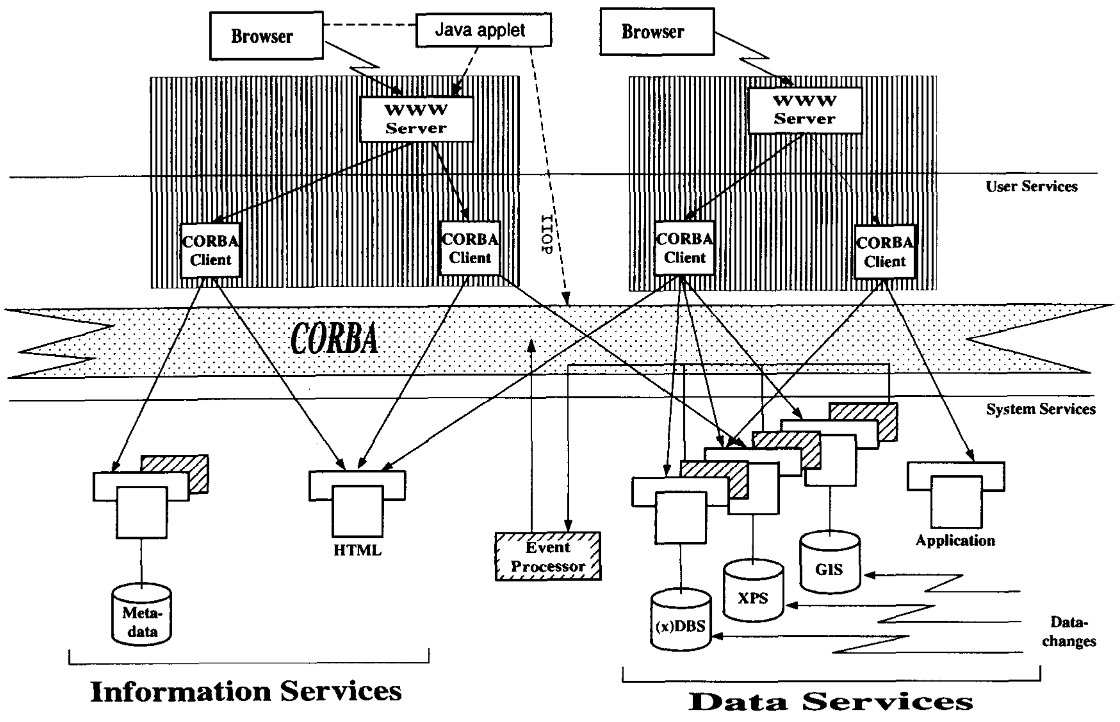

Figure 1: Architecture of a CORBA-integrated federated information system

- Coordinational interoperability imposes discipline on the interaction between objects, and is usually expressed by common policies, contracts and protocols to which the objects subject their activities.

All these characteristics strongly suggest a central role for database technology, particularly its extensions to distributed and federated databases. Other disciplines that should be drawn in are network operating systems, high-level communications or even distributed workflows.

We give an example how approaches to all three aspects of interoperability are combined into a single coherent solution for the information infrastructure. A federated environmental information system (EIS) draws on a number of heterogeneous, geographically dispersed data sources which are mostly online accessible and include geographic information systems, measurement databases, environmental reports and regulations, limit value databases, expert systems and the like. These database systems are based on a variety of database models, e.g., relational, pre-relational, and object-oriented. Users are environmental specialists in industry and public administration who interrogate the data for regulations and facts in order to determine how to meet environmental protection concerns, or who have to take spontaneous actions in case exceptional situations have arisen. For them the distribution of data across systems and locations is mostly irrelevant, they prefer to view the network as a single database. Consequently, the user interface should be uniform and hide all aspects of distribution.

Figure 1 depicts a high-level view of the system architecture [KKN+96]. For platform interoperability it uses today's standard solution of an integration layer for heterogeneous distributed systems, so-called middleware [OHE96]. Under our premise of Section 2, the natural choice is a layer which provides a homogeneous object-oriented view of the sources. Consequently, we utilize the Object Management Group's (OMG) industrial standard of the Object Management Architecture (OMA), with the Object Request Broker (ORB) for exchanging requests and responses as its backbone. The different EIS services are integrated as objects via wrappers (stubs and adapters for, e.g., DBMS access). The CORBA Services provide the basic functions to realize all the mechanisms on behalf of a client such as locating the source object implementation, preparing it to receive the request and communicating the data [Obj95, Sig96].

Existing approaches to notational interoperability are in the form of data exchange standards, wrappers for adjusting object interfaces to some global data model, or mediators for bridging differences in structure, terminology and interpretation of schemas. We deal with it by the information services of Figure 1. Connotational interoperability together with the needed selectivity is achieved with the help of one or more environmental data catalogues UDK. Each UDK contains metadata which identify the data sources by availability, contents, purpose, circumstances of acquisition, reliability, statistical significance, preprocessing, and the like. Differing terminologies are the responsibility of a terminological, multilingual thesaurus as known from information retrieval. Since different domains may require different thesauri, we extended our architecture to integrate multiple, multi- and monolingual, distributed and heterogeneous thesaurus databases [KN96]. Denotational interoperability is provided in the form of a library of mediators, most of which support visualization and digital image processing of maps and diagrams in connection with GIS or provide for conversions between standard raster image formats.

The user-level services of Figure 1 are supposed to give the user the visual appearance of a homogeneous database. Because a user request will often require the combination of several system-level services into a single visualization service, user-level services must include some form of coordinational interoperability. For visualization uniformity across a network a solution based on World-Wide Web suggests itself. Although almost each metadata and data source provides WWW tools, using these directly would not hide the distribution from the user who would face a separate WWW page for each source. Coordination requires that a user is provided with a single HTML page per request. This is done by combining contributions from different distributed sources. The collection of the contributions follows some script which dynamically prepares the corresponding HTML pages, in some general or user-specific fashion, using the inputs from the various data sources [KKN+96, KNK+97]. We foresee several CORBA clients, one each for encapsulating a script for a specific type of request. An end user, then, submits his request via a browser to a WWW server which calls, via the common gateway interface (CGI), the appropriate CORBA client, which in turn accesses several system-level services according to the script.

Because the HTT protocol is stateless, coordination lacks the reliability of a transaction service. Consequently, we have more recently substituted a driver based on the Java Database Connectivity (JDBC) standard for it. Java applets assume the role of the CORBA clients and use the Internet Interorb Protocol (IIOP) to communicate with their server counterparts.

4 Proactivity

In a network of autonomous nodes each object may take an initiative. Consequently, from the perspective of each node the global database is an active database. Take our environmental information system. There are many situations, such as pollution monitoring, where the EIS should on its own become aware of what is happening around it, and react properly and spontaneously, e.g., by automatically notifying a public officer. Active database management systems (ADBMS) are a response by the database community to the need for proactivity [Con96]. ADBMS technically base their behaviour on the event-condition-(re-)action paradigm which reduces all interactions with the system environment to the notion of an observed event that causes a condition to be satisfied which then gives rise to an action being taken. Event-condition-action (ECA) rules appear also an ideal means to support a wide a variety of coordination protocols and, hence, coordinational interoperability.

The challenge is to take ADBMS from a monolithic systems approach to a networked approach which facilitates orderly interaction and collaboration between widely dispersed, loosely coupled, autonomous data-intensive systems. For example, event handling must be able to observe and deal with spatially and temporally distributed events; to combine them into complex events despite temporal uncertainties due to lack of global time; to overcome unreliabilities due to loss of events or connections; and to prevail over unpredictabilities due to autonomous decisions of event providers and consumers, such as non-acceptance or non-responsiveness. Condition checking and action execution may also be distributed.

All these challenges are compounded by the heterogeneity of the data and event sources and the action providers. Consequently, not only must ADBMS technology draw on techniques drawn from communications, distributed systems, and system reliability, but it faces all three aspects of interoperability. Not surprisingly, then, we solve platform interoperability by means of CORBA.

So far CORBA provides only limited support in the form of a basic event transmission service via event channels across a distributed environment. Notational and coordinational interoperability, for example in the form of a rule management facility, must be provided as add-on services. Starting from standard centralized ADBMS techniques we have been able to identify a number of smaller event service units which are candidates for distributed event processing [BKK96]. Figure 1 indicates the integration of the ECA rule service into the overall architecture, Figure 2 the detailed, still partly centralized solution. The wrapper of each distributed source of events, e.g., a data source, must be capable of signalling primitive events which the central event processor receives either by polling or notification from event detectors. Events are first collected into a persistent event storage in order to capture them reliably and to derive complex events from them. Global ordering of events is achieved within

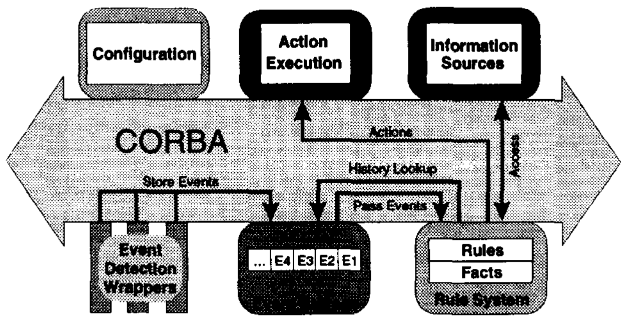

Figure 2: Distributed ECA rule service using CORBA

the limits of well-known techniques, e.g., those of the CORBA Time Service. Notational interoperability is achieved for events by categorization of event sources.

The event storage passes the detected events as an "output stream" to an expert system shell providing production rules. The events are mapped to facts for the rule system. If such a fact together with a condition defined over it gives rise to the application of a rule, the associated action is initiated at one of the distributed objects attached to the ORB - for example a data source or WWW client - by a CORBA method call. Both synchronous and asynchronous decoupled processing of events and action execution are possible. Likewise one may choose among different strategies for event consumption, rule orderings or parallelism of rule execution.

5 Interactiveness

Interactiveness refers to the need for, and capability of autonomous and active objects to enter into a temporary and task-specific collaboration. Technically speaking, interactiveness deals with the cooperation of independent processes subject to certain constraints or norms. Consequently, in a world of active objects we must deal with constraints that span several objects.

Transaction processing in database technology emphasizes coordinational norms. Only more recent developments such as cooperating transactions [Elm92] or scripts superimposed on transactions [WR92] deal with collaboration. Classical object orientation regulates the behavior of individual objects but does little to support multi-object constraints that are spread across several object implementations. Again, a few recent extensions try to rectify the situation [AGP95, HHG90, LM92, LW95].

Constraints allow a nondeterministic evolution of the message exchanges among objects. In other words, they tolerate a wide range of unexpected situations as long as they fall within a given solution space. In a network of autonomous databases, even though these have a well-defined functionality, the evolution is influenced by factors such as choice among several options on whether and how to react to incoming messages, reordering of responses with respect to the messages, erroneous and malicious responses, unexpected initiatives. Further, the interaction is exposed to the vagaries of the underlying communication network.

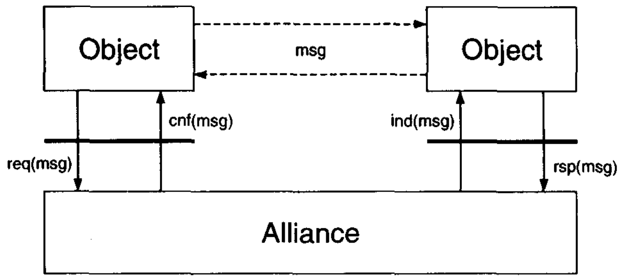

Figure 3: Alliances as a protocol metaphor

A declarative specification of norms is commonly referred to as a protocol. Since we take a network view, the communication channel as the mechanism for encapsulating a protocol in data communication suggests an approach to object technology whereby an explicit construct, the alliance, is added for encapsulating multi-object constraints [LW95]. Figure 3 conveys the idea.

A (un-)acknowledged message exchange between two objects raises corresponding events with the alliance. The cooperating objects remain unaware of the alliance, they still have the impression of message exchange directly with other objects. On the other hand, no message exchange escapes the attention of the alliance. It can now control the communication in many ways, e.g., simply transport messages, accumulate messages before delivery, ignore unexpected messages for the protection of objects, schedule deliveries, employ timeout mechanisms to cope with situations where expected messages do not arrive, provide certain guarantees such as fault tolerance by guaranteed delivery of possibly incomplete responses, temporal ordering of messages, error compensation.

As is common for data communication protocols, the specification of an alliance takes the form of a (sequential) automaton. Consequently, the specification consists of a description of states and their initializations, and of the transitions between states. The latter is in terms of ECA rules. Each rule identifies the event it reacts to, an optional condition which guards the action, and an action. Events may be external requests in the form of a message (perhaps parameterized) and a role name to denote the originator, a response, an internal event for spontaneous transitions, or a clock alarm to realize timeout mechanisms. Alliances extend traditional automata in an important way, though, by drawing on role models for object systems [RTL91, KM94]. A role states the part an object plays in the cooperation, and is declared in terms of the messages such an object must be able to receive and to send. Hence, roles constrain the types of objects that can be bound to it. An object may dynamically be bound to a role any time on explicit request by it, and disconnected on request by either, object or alliance.

Alliances match beautifully with the ECA mechanism of Section 4. Consequently, alliances are services to be placed above the ORB in Figure 1 where they assume a role similar to the CORBA clients and could use the CORBA services including the event mechanism of Section 4. They could also be considered as a value added to the event mechanism to deal with the uncertainties and unreliabilities of that mechanism. Alliances have also been used for controlling the distribution and migration of participating objects within a network [CKW96].

The close kinship to data communication protocols suggests implementation options that range from a centralized solution within an single and special alliance object that runs in its own process, all the way to fully distributed algorithms where a local alliance algorithm is part of the protocol tower of each participating object that is itself connected to the ORB[LW95]. Take Figure 1 where most of the data and information services already show a local connection to the event mechanisms. These connections could be augmented by the alliance algorithms.

Alliances may then play a useful role in the environmental information system. Just consider that emissions of a power plant are measured and, when these exceed a predefined level, data sources with limit values, legal regulations, air dispersion models, and contingency plans collaborate to determine which actions to recommend to an official in charge who may himself interact by submitting various scenarios. So far alliances have been used in support of interleaved engineering and manufacturing processes.

6 Legacy

One of the hopes for object-orientation is that it results in the design of systems whose components have well-defined, well-focussed and, hence, well-circumscribed functionalities. In our previous example the component objects are complete systems such as as database systems, geographic information systems, catalogues, thesauri. This is still better than the old monolithic and gigantic information systems. However, since these provide essential information processing services which rely on huge information repositories, they cannot simply be discarded. On the other hand, if we wish to incorporate them into the global network and treat them as part of the global database, with properties of interoperability, proactivity and interactiveness, they must be reengineered into societies of more focussed objects [IJ93]. Consequently, reengineering of legacy information systems is an important issue for the global database.

Traditional software reengineering methods tend to place priority on legacy programs, database design methods on database structure. Modern object-oriented analysis and design methods such as OMT [RBP$^+$91] take an equal and integral view of object structure, object functionality and dynamics within and between objects. Hence by choosing OMT, reengineering and database design methods can be fused, and one would still entirely stay within an object-oriented framework. Equally important, because of the integral view one should be able to describe the entire transition from the old functional to the new object world within the same framework.

Our basic philosophy as database people is to start from a data-oriented perspective. We motivate this by the observation that data form the backbone of every information system, with their structure undergoing only slow changes whereas system functionality is constantly adapted to new requirements. Our goal is to identify semantic units in the sense of complex application or business objects. This philosophy results in a reengineering strategy with three clearly identifiable steps [UK97]. 1. In a first step we design an object-oriented model of the intended (target) system. The completeness of the model depends on the information available at the time. In the best case, an enterprise model exists from prior business process reengineering. In the worst case, one can only draw on the engineers' applications knowledge. In general, the target model will describe the system at a high level of abstraction, e.g., in terms of business objects, processes and functions. 2. In the next step, a description of the existing system is developed, mainly by examining the inner structures of the information system. Although highly detailed, this implementation-oriented description (the state model) can neither be expected to be complete nor correct because of deficits in the existing documentation and inadequacy of the analysis tools. 3. By now two descriptions at different levels of abstraction - target and state - have been created. In the last step, the state data sources and the application programs are analyzed in order to find one or an aggregation of equivalen target data structures that represent the target objects. If the engineer rejects the analysis, the target objects can be changed or the step abandoned altogether. Otherwise the target equivalent in the state model is inspected in more detail, and a variety of software engineering methods and tools are used to delineate the corresponding part of the legacy system and, in particular, to identify candidates for methods of the object [SN95]. On this basis converters are constructed that substitute object-oriented code for the old one.

The three steps do not form a linear process. System size is too large to be manageable in one piece and must be divided into smaller units. Past experiences will affect the state and target models. Converters may fail to be found or to work. Hence, the overall reengineering process is broken down into a number of smaller processes called microprocesses. They are managed by a large process called the macroprocess which controls and guides the microprocesses and reacts flexibly to all changes and re-planning requirements. The macroprocess will iterate until the state and target models converge sufficiently well.

Even within a microprocess the steps are not necessarily followed in a linear fashion. Indeed, it is the strength of OMT that it allows to analyze and transform the business objects, the business processes, and the functionality of applications in parallel and to exploit the interdependencies between these submodels, and to create, automatically or on the engineer's suggestions, new ones on the next higher level of abstraction or reorganize, by appropriate changes, the current level [PH95].

Our reengineering methodology bears a strong semblance to the software engineering methodology of the spiral model of Boehm [Boe88a, Boe88b]. The combination of this methodology with a uniform OMT-based framework, and a well-organized, data-centered process model appear rather novel. It can easily be extended to a collection of macroprocesses when the system is large but has clearly identifiable subsystems. A workflow environment supports the reengineer by mediating between him, the knowledge about the legacy system, the reengineering process and the applicable engineering methods. Also an integral part of the methodology is a repository which contains a meta model of the reengineering process and follows an object-oriented data model as well.

7 Conclusions

The thesis of this paper was that the view of network information systems as one giant global database is not only a natural one to users but also suggests a framework within which much of existing database technology could be carried forward to networking. It identified four challenges - interoperability, proactivity, interactiveness and legacy -, and explored in the context of several application scenarios how database technology had to be extended, and how it could be combined with other information technologies to provide viable solutions. We could demonstrate that the view of global database seems indeed a productive strategy for attacking a number of challenges, and that object-orientation seems an ideal framework for gluing various technologies together. We could further show that the necessary extensions to database technology do not so much affect the local data sources in the individual nodes than suggest separate facilities which could be considered add-on values to be placed within the network infrastructure. By relying on object-orientation one should also be able to incorporate other promising developments such as higher protocol levels in wirebound and wireless, mobile data communication [Kot95], or data security techniques [Gro95, Sig96].

The practical application scenarios were taken from a widely used public environmental information system (interoperability and proactivity), an integrated platform for controlling the entire engineering and manufacturing process chain for mechanical parts (proactivity and interactiveness), and the reengineering of information systems of a public utility (legacy).

References

[AGP95] J.-M. Andreoli, H. Gallaire, and R. Pareschi. Rule-Based Object Coordination. In *Object-Based Models and Languages for Concurrent Systems*, number 924 in LNCS, pp. 1–13, 1995.

[BGK$^+$95] J. Bailey, M. Georgeff, D.B. Kemp, D. Kinny, and K. Ramamohanarao. Active Databases and Agent Systems - A Comparison. In T. Sellis, editor, *Rules in Database Systems*, number 985 in LNCS, pp. 342–356. Springer, 1995.

[BKK96] G.v. Bültzingsloewen, A. Koschel, and R. Kramer. Active Information Delivery in a CORBA-based Distributed Information System. In *Proc. 1st Int. Conf. on Cooperative Information Systems (CoopIS96)*, pp. 218–227, 1996.

[Boe88a] B.W. Boehm. A Spiral Model of Software Development and Enhancement. *IEEE Computer 21*, (5):61–72, 1988.

[Boe88b] B.W. Boehm. Anchoring the Software Process. *IEEE Software 13*, (4):73–82, 1988.

[CCWB94] P.R. Cohen, A. Cheyer, M. Wang, and S.C. Baeg. An open agent architecture. In *Proc. AAAI Spring Symp. on Software Agents*, pp. 1–8, March 1994.

[CKW96] O. Ciupke, D. Kottmann, and H.-D. Walter. Object Migration in Non-Monolithic Distributed Applications. In *Proc. 16th Int. Conf. on Distributed Computing*, pp. 529–536, 1996.

[Con96] The ACT-NET Consortium. The Active Database Management System Manifesto: A Rulebase of ADBMS Features. *ACM SIGMOD Record*, pp. 40–49, September 1996.

[Elm92] A. Elmagarmid, editor. *Database Transaction Models.* Morgan Kaufmann, 1992.

[GKT95] A. Geppert, M. Kradolfer, and D. Tombros. Realization of Cooperative Agents Using an Active Object-Oriented Database Management System. In T. Sellis, editor, *Rules in Database Systems*, number 985 in LNCS, pp. 327-341. Springer, 1995.

[Gro95] Object Management Group. CORBA Security. Technical report, Object Management Group, Inc, 1995.

[HHG90] R. Helm, I.M. Holland, and D. Gangopadhyay. Contracts: Specifying Behavioral Compositions in Object-Oriented Systems. In *Proc. ECOOP/OOPSLA*, pp. 169–180, 1990.

[IJ93] F.Lindström, and I.Jacobson. Re-engineering of Old Systems to an Object-Oriented Architecture. In R.S. Arnold, editor, *Software Reengineering*, IEEE Comp. Soc. Press, pp. 564–581, 1993.

[KKB+97] A. Koschel, R. Kramer, G.v. Bültzingsloewen, T. Bleibel, P. Krumlinde, S. Schmuck, and C. Weinand. Configurable Active Functionality for CORBA. In *ECOOP'97 Workshop - CORBA: Implementation, Use and Evaluation*, Jyvaskula, Finnland, June 1997.

[KKN+96] A. Koschel, R. Kramer, R. Nikolai, W. Haag, J. Wiesel, and H. Jacobs. A Federation Architecture for an Environmental Information System incorporating GIS, the World Wide Web, and CORBA. In *3rd Int. Conf./Workshop Integrating GIS and Environmental Modeling (NCGIA'96), 1996.*

[KM94] A. Kemper and G. Moerkotte. *Object-Oriented Database Management.* Prentice-Hall, 1994.

[KN96] R. Kramer and R. Nikolai. Accessing Multilingual, Heterogeneous Data Sources in Wide Area Networks - Requirements and Approach. In *Proc. Workshop on Flexible Query-Answering Systems*, pp. 255–264. Roskilde Universitetscenter, Denmark, 1996.

[KNK+97] R. Kramer, R. Nikolai, A. Koschel, C. Rolker, P. Lockemann, A. Keitel, R. Legat, and K. Zirm. WWW-UDK: A Web-based Environmental Metainformation System. *ACM SIGMOD Record 26*, (1):16–21, 1997.

[Kot95] D.A. Kottmann. Serializing Operations into the Past and Future: A Paradigm for Disconnected Operations on Replicated Objects. In *Proc. ECOOP'95 Workshop on Mobility and Replication*, 1995.

[LM92] L. Liu and R. Meersman. Activity Model: Declarative Approach for Capturing Communication Behavior in Object-Oriented Databases. In *Proc. 18th Int. Conf. on Very Large Data Bases*, pp. 481–493, 1992.

[LW95] P.C. Lockemann and H.-D. Walter. Object-Oriented Protocol Hierarchies for Distributed Workflow-Systems. In *Theory and Practice of Object Systems (TAPOS) 1:4*, pp. 281–300, 1995.

[Obj95] Object Management Group. The Common Object Request Broker: Architecture and Specification. Object Management Group, Inc. (OMG), 1995. Version 2.0.

[OHE96] R. Orfali, D. Harkey, and J. Edwards. *The Essential Distributed Objects Survival Guide.* John Wiley & Sons, 1996.

[PH95] G. Pernul and H. Hasenauer. Combining Reverse with Forward Database Engineering. A Step Forward to Solve the Legacy System Dilemma. In *Proc. Int. Conf. on Database and Expert Systems Appl. (DEXA)*, LNCS. Springer, 1995.

[Pog95] A. Poggi. DAISY: an object-oriented system for distributed artificial intelligence. In *Intelligent Agents - Theories, Architectures and Languages*, LNAI. Springer, 1995.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design.* Prentice-Hall, 1991.

[RTL91] R.K. Raj, E. Tempero, and H.K. Levy. Emerald: A General-Purpose Programming Language. In *Software - Practice and Experience 21:1*, pp. 91–118, 1991.

[Sig96] J. Sigel. *CORBA Fundamentals and Programming.* John Wiley & Sons, 1996.

[SN95] H.M. Sneed and E. Nyary. Extracting Object-Oriented Specification from Procedurally Oriented Programs. In *Proc. IEEE-TCSE ACM-SIGSOFT Working Conf. on Reverse Engineering.* IEEE Comp. Soc. Press, 1995.

[UK97] M. Wallrath U. Kölsch. A Process Model for Controlling and Performing Reengineering Tasks. In *Proc. First Euromicro Working Conf. on Software Maintenance and Reengineering*, pp. 20–23. IEEE Comp. Soc. Press, 1997.

[WJ95] M. Wooldridge and N. Jennings. Agent Theories, Languages and Architectures. In *Agent Theories, Languages and Architectures*, LNAI, pp. 1–39. Springer, 1995.

[WR92] H. Wächter and A. Reuter. The ConTract Model. In Elmagarmid [Elm92], pp. 219–264.