REaltime ACtive Heterogeneous Systems - Where Did We Reach After REACH?

Thomas Kudraß

Fakultät Informatik, Mathematik, Naturwissenschaften
Hochschule für Technik, Wirtschaft und Kultur Leipzig
Postfach 30 11 66
04251 Leipzig
kudrass@imn.htwk-leipzig.de

Abstract: This paper gives a survey of the deployment of ideas from the area of real-time, active and heterogeneous database systems in the years from 1991 to 2010 as they have been embraced by IT industry. During that time the Database and Distributed Systems group (DVS) led by Alejandro Buchmann has made lots of contributions to the development of those ideas by many research projects. After 20 years it is time to conclude insights how far the ideas of the first project REACH are still valid for the development of commercial products and standards. In some cases, industry has taken another direction as it has been expected. In other cases, the DVS research prototypes were forerunners for commercial products that are now well-established.

1 The Early History before REACH

The early research topics of Alejandro Buchmann and his colleagues comprised architectural issues to be resolved when using a database system as an active object in a distributed environment with real-time capabilities. In the end of the 80s years, many of those ideas were discussed in the research community but were still far away from commercial use.

The DOM (Distributed Objects Management) project at GTE Labs addressed the integration of autonomous, heterogeneous database and non-database systems into a distributed computing environment. In [Bu90] the idea of an active object space was sketched to model heterogeneous cooperating information systems. An active object has been defined by its capability to react autonomously and asynchronously to incoming events. Object-oriented models were considered best as common model for federations of heterogeneous systems because of its encapsulation and data abstraction features.

The late 80s years were characterized by the expectations of a soon retirement of the then very popular relational DBMS. Therefore, object-oriented data models were the most favoured to express the behaviour of a system in a heterogeneous environment. There was a variety of ideas how the ideal object model looks like regarding its expressiveness and relationship types. But there was still some hope on an object model standard developed by consortia like the Object Management Group (OMG) or the Object Database Management Group (ODMG). The notion of active objects was defined with various forms that have appeared in active database systems that were another source of inspiration for DOM.

The research of active databases was mainly influenced by some pioneering projects. Among them HiPAC introduced the event-condition-action (ECA) rule abstraction. One frequently cited HiPAC paper was titled “Rules are Objects Too” [DBM88] that promoted the idea to treat rules as first-class objects. One of the HiPAC ideas was the definition of timing constraints and their assignment to the rules or to a part of it in order to support application scenarios that require reactions within a certain time period. In that way, the real-time feature became the third part of the first Darmstadt research project REACH (REaltime Active Heterogeneous System) that started in 1992. REACH pursued the idea to *really* implement lots of the ideas of HiPAC and DOM to study the real problems in an object-oriented database system that is active, works in a time-constrained manner when executing queries and can be the platform for a mediator in an active object space using rules to control overall consistency or global transactions spanning different components.

The paper is organized as follows: After a retrospective on the early history and the REACH project we look at the development of concepts of system integration, active capabilities and global consistency control driven by IT industry. The paper continues with lessons we have learned from experiences in applying REACH ideas in the development of large information systems. Finally a short outlook on future issues concludes the paper.

2 The REACH Project – REaltime ACtive and Heterogeneous

2.1 Real-Time Databases

The idea to constrain the execution of rules relates to the concept of real-time databases and to the incorporation of the time dimension to specify rules. The ability to process results in a timely and predictable manner will always be more important than fast processing. So real-time databases are able to handle time-sensitive queries, return only temporally valid data, and support priority scheduling. Deadlines are the constraints for soon-to-be replaced data accessed by the transaction [BB95]. Deadlines can be either observant or predictive. The latter approach is a more stable way of dealing with deadlines but requires the capability to predict the transaction behaviour. The response to a missed deadline depends on whether the deadline is hard, firm or soft. A hard deadline has to be met, otherwise it creates serious problems. Transactions passing the deadline must be aborted. Firm deadlines are similar but they measure how important it is to complete the transaction at some point after the transaction arrives. In real-time environments the data quality decreases as time progresses between the time data was acquired and the time it is consumed. This can be expressed by value functions that specify the value of the outcome of the transaction dependent on the elapsed time. Soft deadlines can be applied best, if meeting time constraints is desirable but missing deadlines do not cause serious trouble.

2.2 Active Databases

An active database system monitors situations of interest and triggers an appropriate response in a timely manner when they occur. The desired behaviour is expressed by ECA rules that can be used to specify static or dynamic constraints in a distributed environment. The monitoring component of an active database is responsible for the detection of events and their propagation to a rule engine. An event may trigger the execution of one or more rules. A rule is executed by evaluating the condition and possibly executing the action. We can define a coupling mode between event and condition (EC) and between condition and action (CA) as well. The different coupling modes as they have been introduced in [HLM88] specify the execution of the rule as a single transaction or even as independent transaction (detached) with consequences to the required transaction model for the active system beyond flat transactions. In REACH different rule subclasses that inherit their structure from a RULE superclass were introduced for different domains: access control, consistency enforcement, flow control and application-specific rules [BB+93]. So it was possible to process all rules in a uniform manner.

The event hierarchy of REACH comprised not only changes of the database state as events but also temporal and transaction events. Due to the underlying object paradigm events were related to (active) objects such that method calls and operations on attributes became part of the event hierarchy. By treating transactions as objects, transaction-specific events were subclasses of a method call event. Besides primitive events, composite events were defined using logical operators. Any relative temporal event was defined relative to another event that originated in a committed transaction – contrary to absolute time events that were classified as primitive events. A specific aspect was the event consumption semantics when processing lots of event occurrences together with their parameters. Although events can be considered instantaneous, the time difference between occurrence and detection of an event has to be taken into consideration when implementing an active system with event detection. Events are strictly distinguished from temporal constraints on rules that determine when a rule execution begins and when the execution of a rule must be complete.

2.3 Integrating Heterogeneous Databases

Many large companies use various semi-autonomous and heterogeneous database systems to serve the needs of various application systems. As it has been mentioned above, one important application domain for REACH was the consistency control in heterogeneous database that cooperate together in federations with different degrees of coupling. These federations can be designed in a layered architecture with different abstraction levels [SL90]. The term *interdependent data* was coined to imply that two or more data items stored in different databases are related through a global integrity constraint [SRK92].

The idea of active objects was to set up an object model that represents the local components to be controlled. That could be done on a coarse-grained level by viewing a local component and its interface as an object with methods that represent the behaviour of the whole system. Relational databases could well be integrated into such a system by object-relational mapping techniques that make it possible to define rules on objects that represent relations or single tuples in remote databases.

Many practical problems have to be tackled when dealing with heterogeneous databases. Since the data items to be managed may be distributed throughout a network events on them can also be distributed. In a rather conventional approach all events could be collected and processed by a mediator with active database features as they have been worked out in REACH [KLB96]. One of the unresolved issues was how an event service can deal with distributed events in an open environment without a global clock. Liebig et al. [LCB99] presented algorithms for event composition and consumption that used accuracy interval based timestamping and discussed the problems that result from inaccuracy and message transmission delays.

A global system that interacts with components that have been designed independently has to deal with one main obstacle, the local autonomy of the participant. In general, autonomy can be characterized as the freedom to make decisions. It comprises three different categories: structure, behaviour and communication [Ku97]. Structural autonomy covers all design aspects of a system, e.g. its schema and internal system architecture. The behaviour autonomy describes the capability of a local system to decide independently about the actions it executes. Actions can be executed at different local interfaces (e.g., SQL operation or a local operation call) and change the state of the local database. The behavioural autonomy can be restricted by proscribing local actions or, vice versa, by enforcing actions that are part of a global transaction spanning multiple systems. The communication autonomy describes the freedom of a local system to decide about the information it is willing to provide to the federation. Among them are: status information at run-time of the system, data and schema information, and occurred events.

Even if the schema is public there might by some remaining problems to understand the semantics of the schema elements that is a prerequisite for schema integration algorithms in tightly coupled integration approaches [RB01]. Alternatively, metadata about local systems can be used in a global knowledge base in addition to global integrity rules. In the early REACH prototypes the role of metadata was not completely analyzed but it was worth doing so, because lots of global knowledge could be transferred to a metadata base instead of rules.

3 After REACH: Concepts in Distributed Heterogeneous Systems

3.1 Integration Technologies

Object-Oriented Middleware

When the author left the DVS group at the Technical University of Darmstadt 1997 some new paradigms and trends entered the stage in the IT community. The dominance of relational databases continued, the object-oriented paradigm gained importance as model for middleware in interoperable systems. In such distributed computing infrastructures, DBMSs were considered one kind of component over which distributed applications are built.

As an example, the Common Object Request Broker Architecture (CORBA) was defined as a standard to enable software components written in multiple languages and running on multiple computers to work together [Obj04]. The interface definition language (IDL) of CORBA specifies interfaces that objects will present to the outside world. To use different implementation languages mappings from IDL to specific languages were defined. An Object Request Broker (ORB) is the platform for the cooperating applications to interact. In addition to providing users with a language and platform-neutral remote procedure call specification, CORBA defines a landscape of commonly needed services such as transactions, security, time and events. The CORBA Event Services support the push model, in which a supplier object initiates the transfer of event data to consumer objects, as well as the pull model, in which a consumer of events requests event data from an event supplier. Composite events are not explicitly defined in the standard. The CORBA specification as the brainchild of a committee appeared very complex, ambiguous and hard to implement entirely. Thus, existing ORB implementations were incomplete or inadequate [He08].

There are some other examples of object-oriented middleware that have gained some popularity. Among them is Java RMI (Remote Method Invocation), the Java programming interface that performs the object-oriented equivalent of remote procedure calls. Jini was the more advanced version of RMI with better searching capabilities and mechanism for distributed object applications [Sun99]. A major competitor of CORBA was DCOM (Distributed Component Object Model), a proprietary Microsoft technology for communication among software components distributed across a network. The difficulties of both CORBA and DCOM technologies to work over the internet and on unknown and insecure machines hindered their broad acceptance as middleware standards.

Integration Infrastructures

The integration of heterogeneous systems beyond the communication in a RPC style was addressed in EAI technologies (Enterprise Application Integration) as they become popular in the end of the 1990th years. EAI can be considered a framework to integrate applications within an organization to support business processes that run on different systems, such as supply chain management, or to enable business intelligence applications with complex analytical operations over lots of distributed heterogeneous data [CH+05]. The EAI system can provide a single uniform access interface to different local application (façade) and can ensure that data of different sources is kept consistent. This is also known as EII (Enterprise Information Integration) [HA+05]. An EAI approach avoids point-to-point communication by a centralized infrastructure that has either a hub-and-spoke or a bus topology.

The latter can be implemented using message-oriented-middleware. The advantage of a message-based middleware is the decoupling of the sending, receiving and processing of messages in an asynchronous way. Local applications can preserve more autonomy (cf. 2.3) and the overall system can be more flexible and failure-tolerant. The publish/subscribe model brings message publishers together with message consumers that subscribe messages with certain topics. One of the most widely used technologies providing publish/subscribe capabilities is the Java Message Service (JMS) [Sun02]. Since there are two main use cases for EAI, we can distinguish between mediation and federation scenarios. The mediation scenario resembles the active object style as it has been conceived in [Bu90, KLB96]. Whenever an interesting event occurs in an application (e.g. data change, end of a transaction) a component of the EAI broker is notified, necessary actions (e.g. data propagation) are fired. The federation scenario can be used to shield the user from local interfaces when business intelligence applications on multiple applications have to be executed. Message brokers and enterprise service bus systems (ESB) are typical implementations of the EAI approach.

Data Representation

The coexistence of different data models used by autonomous participants of a distributed information system made it necessary to think about a suitable format for data exchange regardless of the used communication protocol. A canonical data model is considered an independent data model based on a standard data structure. It must be stated that the idea of distributed objects that communicate via method calls has become obsolete due to the same reasons why object-oriented database systems failed to overtake the market. There was no single widely-agreed object model although there have been standards defined by the Object Management Group. The OMG Core Object Model with few basic concepts was just an abstraction usable to define interfaces but required language bindings for every implementation. To exchange information between cooperating systems a data serialization is necessary to represent data independently from their original site. For that purpose, XML (together with XML Schema) was an important milestone. It appears that XML and the use of XML stylesheets has become the standard for this universal business language that has been needed for many years.

Service-Oriented Architecture and Business Process Management

XML is commonly used for interfacing services in a service-oriented architecture (SOA). SOA defines how to integrate disparate applications in a web of autonomous systems with multiple implementation platforms. It can be considered as the further development of EAI technologies. A service in a SOA presents a simple interface in terms of protocols and functionality that abstracts away its underlying complexity. Users can access independent services without knowledge of the service implementation. Web Services can implement a service-oriented architecture. The Web Service Description Language (WSDL) describes the services themselves, while SOAP describes the communication protocol. Alternative light-weight technologies can be used to implement a SOA (e.g. REST).

One main purpose of SOA is to allow users to combine chunks of functionality as they are represented in services to form applications in an agile way (resembling SAP cross-application technologies). Ideally, services can be used to construct executable business processes (also known as workflows) that span different organisations. Standards such as BPMN and WS-BPEL [Obj09] can be used for the definition and execution of business processes with service calls as process steps. So a common language to describe long-running activities over heterogeneous systems is available. The travel reservation example of DOM [BÖ+92] with all dependencies of the activities can be specified in a convenient way using the BPMN notation. WS-BPEL supports business transaction by a compensation-based mechanism (long running transactions). Since we are working in an open environment using the web, closed nested transactions can not be employed. Instead, compensating activities can be executed to undo effects of already finished activities if the overall process has to be cancelled. Fault and compensation handler can be defined as equivalent to contingency transactions of the DOM transaction model [BÖ+92].

3.2 Active Capabilities

Business Rules

Looking back at REACH and the active objects, the question remains: Where are the ECA rules gone? The need for active features is generally accepted. Therefore, so-called business rules have been incorporated into business processes [Ro03]. The idea of business rules is similar to that of triggers, they describe invariants that specify constraints of business aspects, e.g., in a credit application workflow the maximum loan sum that can be given to a bank customer of a certain category. All business rules can be maintained in a central business rules repository that should interact with a workflow engine. Business rules are different from flow logic that is inherent in the business process. The latter can be found in the process specification as there are control flow elements available (such as flow objects in BPMN or structured activities in BPEL). It is a software engineering issue where to place business-related rules. Alternatively, special decision services that encapsulate a more complex logic could be integrated as activity into the business process. Database triggers remain an important part, as they are responsible for local consistency constraints on a (lower) data level. Business Rule

Management Systems (BRMS) have been evolved from rule engines, based on declarative definitions of business rules. The rule representation is mapped to a software system for executions. BRMS vendors have been acquired by big middleware companies because of the obvious need to integrate rules into business processes, for example: ILOG + IBM = IBM WebSphere ILOG BRMS [IBM10], Drools + JBoss = JBoss Rules [Jb10]. Thus the business rules approach gains more importance because it is a key to agile processes with flexible rules that are interpreted dynamically and may be changed at run-time of the process without adaptations of services.

Compared to ECA rules, the SQL standard imposes lots of restrictions on database triggers that consider only database operations as event or action. It is interesting to note that there is some more emphasis on the role of events in business process definitions, particularly in BPMN [Obj09]. An event in BPMN is something that happens (rather than an activity which is something that is done). The current BPMN standard provides a rich set of event types. We can classify throwing and catching events that support both scenarios of event-producing and event-consuming processes. An event can trigger a process as Start event, it can represent the result of a process as End event, or it is an Intermediate event that can catch or throw triggers. BPMN supports following event types: Message, Timer (i.e. absolute point in time or period), Error, Cancel (cancellation of a subprocess), Compensation (compensation of an activity), Conditional (triggered by a condition), Link (connections to other process parts), Signal (without specific targets), Terminate (immediate end of all activities without compensation). In BPMN there exists also a "Multiple" event element, which represents a choice between a set of events. An arbitrary number of other events can be connected to the "Multiple" event, which is in fact a complex event. For example, a message event can be combined with a time event to express the wait for a message with a timeout.

Event Processing

Complex Event Processing (CEP) is an approach [Lu02] that incorporates concepts of active databases, middleware and service-oriented architectures. Among them the issue of composite events and their detection is well-known from early active database research [CM94, BB+93]. Applications with lots of occurring events such as network management but also business process management (BPM) have triggered the development of CEP technologies. CEP must interact with BPM since BPM focuses on end-to-end business processes. A complex situation to be dealt with can mostly be considered as a combination of primitive events on a lower abstraction level. A simple example is calculating an average within a certain time frame based on data of the incoming events. A similar scenario is the processing of event streams to identify relevant events within those streams to enable in-time decision making. Typical applications are stock trading, RFID event processing or process monitoring. Composite events were just a part of the event hierarchy in REACH, whereas complex events with a complex detection or calculation algorithm can be considered more business-oriented in CEP. It is still disputed how far BPM is a natural fit for CEP which would motivate the integration of CEP technology into BPM.

A second trend emerged some years ago as a complement to the service-oriented architecture, the event-driven architecture where services can be triggered by incoming events [Ho06]. Sensing devices such as controllers or sensors can detect state changes of objects and create events which can then be processed by an active, i.e. event-driven, system. Such a system typically acts in an open environment characterized by an unpredictable and asynchronous behaviour. An event-driven architecture is characterized by applications and systems which transmit events among loosely coupled software components and services. The pattern recalls the concept of ECA rules with coupling modes to describe their execution. In an event-triggered architecture we can distinguish different components: event generator, event channel and event processing engine. The communication of events is based on the same principles as asynchronous messaging with queues to be processed later by an event processing engine.

3.3 Global Integrity Control in Heterogeneous Systems

Master Data Management

The problem of global data consistency in an organization operating a landscape of heterogeneous information systems has been addressed by the concept of Master Data Management (MDM). It comprises processes and tools to define and manage non-transactional data [WK06]. Among them are customer data or product data which are quite stable and also reference data such as calendars or geographical base data. The need for MDM is caused by mergers and acquisitions or the organizational autonomy of departments of a large corporation. The coarse design approach for MDM is to install a master data hub, a software component that stores the master data and keep it synchronized with the transactional systems [Wo07]. There are several basic styles of architecture used for MDM hubs: In the repository approach, the complete collection of master data is stored in a single database. The applications that use or produce master data have to be modified to use the master data in the hub instead of the local data. The registry approach is the opposite of the repository approach, because none of the master data is stored in the hub. The master data is maintained in the application databases, and the MDM hub contains lists of keys to find all related records in the local databases. The hybrid model, as the name implies, includes features of both the repository and the registry models, whereby the most significant attributes of master data are replicated on the hub so that certain MDM queries can be satisfied directly from the hub database.

Data Quality

The MDM approach is an important technology to implement data quality in a company. The term data quality considers data in an enterprise-wide context, because data is considered a production factor and an asset [WZL01]. The Total Data Quality Management Approach [Wa98] goes far beyond a global integrity control defined by some rules because the data properties that describe quality include not only “classical” database attributes such as completeness but also semantic and time-related aspects that have to be expressed in a knowledge base of a data quality management system.

4 Conclusions: Where Did We Reach?

When we discussed REACH 18 years ago we underestimated some trends that had impact on the next research directions. Although we already identified some problems in that time, their context was narrower and possible solutions for them were more restricted.

Consider viewpoints in architecture. Although process-oriented information systems (such as BPM) and data-oriented information systems (such as MDM or Data Warehouses) should be kept separate in operation, they can be enriched by active functionality as it has been discussed. Database triggers, business rules, complex events or rules in a master data hub base can be processed in a similar way but they are on different abstraction levels. So they have to be implemented in a non-redundant way avoiding cross-effects. The specification of an open distributed system in terms of viewpoints as the RM-ODP model provides, combined with system layers [Ku03] allows us to define the appropriate software architecture with active capabilities.

Rapid growth of events and data. The explosion of data and information driven by the digitalization and the development of the WWW raises new questions about the data quality of digital assets and (possibly event-driven) tools to control the quality. The internet enables lots of people to contribute information to the web as a global database. Web 2.0 media such as blogs or wikis reflect this trend of the active participation of users as information producers. The drawback of this development is a decline of the quality of the published information, it can be erroneous (i.e. with misspellings), incomplete or outdated. Actually the information quality can be maintained only for data that is mission-critical for an organization. Global data integrity has to be defined in different grades, each with a policy that implies the suitable implementation to ensure the necessary degree of consistency.

Unbundling. We can bring back into use the idea of unbundling that has been discussed for REACH [ZK96]. Active functionality, like other key features as access control or transaction management, can be considered a useful but not mandatory component of a database or information system. When required it must be possible to enrich the system by an active component, e.g. a business rule or a CEP engine in a BPM system. The question how far active functionality should be built-in is always a software engineering issue and depends on the application profile. In some cases it makes sense to use database triggers, in other cases log-based or message-based approaches may be superior, for example when managing master data in an environment of loosely-coupled systems.
The same applies to real-time features. In [BL99] some problems have been discussed when combining REACH technologies with different goals and requirements in one system. For example, real-time systems require predictability of resource consumption and execution time. On the other hand, active databases have to react on events and trigger rules dynamically, which makes predictability rather difficult. A second example is the detection delay for complex events in distributed environments, which may have an impact on the temporal consistency of the data. To solve certain trade-offs between conflicting system components, unbundling can be an approach for application-specific solutions.

Services as the new objects. The object paradigm has not gained the broad acceptance as it was expected when we worked out the REACH ideas. Should we replace “object” by “service” – and now define rules as services? First, we should understand why distributed objects and CORBA have failed. On one hand, they provided some abstraction with regard to the implementation platform and language. However, many ORB users did not understand the need to model some abstraction layers atop. Even some component models providing more complex artefacts were not the solution because they were often too specific and also too complex, e.g. Enterprise JavaBeans in the Java Enterprise Edition. Like distributed objects, services provide the same abstraction regarding platform and physical details. They can be design artefacts as well as executable units of work, typically as web services. They allow loose coupling between service provider and service user as it is required in most scenarios that connect multiple parties, even beyond organizational borders. In active systems, a service can be part of an ECA rule or even represent the whole rule.

Autonomy vs. Quality of Service (QoS). A service level agreement (SLA) is a useful concept to describe the necessary quality of service a provider has to guarantee. The requirements on the service agreed herein imply some restrictions on the local autonomy of the system of the service provider. If a service provider has to guarantee a certain response time he has to adapt his system, particularly in multi-tenant applications that are used by independent clients [GK+08]. Besides operational and availability requirements, data quality can be a key component of an SLA if the service deals with many data.

With services as generic concept, it is still necessary to solve the whole bunch of open software architecture issues, for example where to locate consistency constraints, event composition or execution of business rules. The differentiation of services into business services or lower-level infrastructure services allows to distinguish between low-level events (as they are raised by a sensing device) and business-level events (that may be the result of a computation). In [ASB10] QoS is discussed for event-based systems in terms of features they have to provide with an impact on their autonomy.

Convergence of analytics and processes. The separation of concerns in different abstraction layers allows us to decide independently on the best way to implement a multi-tier architecture with active functionality at some level. For example, it would be possible to define a business rule in a business process with a complex event that is interpreted as the result of the aggregation of many simple event occurrences stored in a database. Complex events mark the borderline between analytical and process-centric systems. There are no restrictions what to define as a service. Even CEP is a possible service candidate [AS+10]. In this way it recalls the “Rules are Objects Too” statement [DBM88]. Business process logs can represent the calculation base of complex events as well as the subject of further analytical, not necessarily event-driven, applications that measure the process quality.

5 Outlook

There will be some more progress in the development of hardware with major implications on research in distributed, active and real-time systems.

The growth of event data by enhanced hardware capabilities (e.g., RFID scanners, sensor networks) to monitor the environment in many scenarios, such as traffic control or healthcare systems, results in data streams that are processed in a way different from traditional DBMSs. The term “Internet of Things” refers to the networked interconnection of everyday objects producing billions of parallel and simultaneous events.
Data streams need not be stored persistently. Instead, standing queries or event patterns specify situations an active system has to cope with. As an alternative platform, main-memory databases are faster than disk-optimized databases, which make them attractive for applications where response time is critical. The emergence of solid state disk (SSD) technology that provides higher read performance over current hard disks will also have an impact on future DBMS architectures. Peripheral devices such as disk controllers or sensors can behave like a database becoming ubiquitous smart objects. Those small and mini databases (embedded databases) have to administrate themselves also known as self-managing, self-healing, always-up. They mark a trend that is also important to traditional DBMSs [Gr04].

Combining ideas of active and real-time databases applied in a heterogeneous world as we envisioned it in the REACH project remains a good approach for many today’s information systems. However, we have to deal with crucial design issues resulting from the complexity of those systems [BL99]. First it is necessary to understand the interaction between different base technologies before moving the boundaries between them towards the goal of a more generic distributed platform with active and real-time functionality. As we suggested in the paper the future development will offer lots of further research challenges in this area.

Acknowledgment

I am greatly indebted to my scientific advisor, Professor Alejandro Buchmann, for his support and for all his suggestions when I was doing my PhD from 1992-97. Many of the problems and the ideas we discussed in that time appeared in application scenarios and IT trends I encountered later. In many cases, the REACH ideas were a key to understand conceptual and architectural issues of system design.

References

[AS+10] Achakeyev, D.; Seeger, B.; Schäfer, D.; Schmiegelt, P.: Complex Event Processing as a Service. GI-Workshop Database as a Service, http://fgdb2010.imn.htwk-leipzig.de, HTWK Leipzig, 2010.

[ASB10] Appel, S.; Sachs, K.; Buchmann, A.: Quality of Service in Event-based Systems. 22nd GI-Workshop on Foundations of Databases (GvD), Bad Helmstedt, Germany, 2010.

[BB95] Branding, H.; Buchmann, A.: On Providing Soft and Hard Real-Time Capabilities in an Active DBMS. Internat. Workshop on Active and Real-Time Database Systems, Skovde, Sweden, 1995.

[BB+93] Branding, H.; Buchmann, A.; Kudrass, T.; Zimmermann, J.: Rules in an Open System: The REACH Rule System. Proc. of the 1st Internat. Workshop on Rules in Database Systems (RIDS), Edinburg, 1993, Springer-Verlag.

[BL99] Buchmann, A.; Liebig, C.: Distributed, Object-Oriented, Active, Real-Time DBMSs: We Want It All – Do We Need Them (At) All? Proc. of the joint 24th IFAC/IFIP Workshop on Real-Time Programming and 3rd Internat. Workshop on Active and Real-Time Database Systems, Saarland, Germany, 1999.

[BÖ+92] Buchmann, A.; Özsu, T.; Hornick, M.; Georgakopoulos; Manola, F.: A Transaction Model for Active Distributed Object Systems. in: Elmagarmid, A. (Ed.): Database Transaction Models for Advanced Applications. Morgan Kaufmann Publ., 1992.

[Bu90] Buchmann, A.: Modelling Heterogeneous Systems as a Space of Active Objects. Proc. of the 4th Internat. Workshop on Persistent Objects, Martha's Vinyard, 1990.

[CH+05] Conrad, S.; Hasselbring, W.; Koschel, A.; Tritsch, R.: Enterprise Application Integration – Grundlagen, Konzepte Entwurfsmuster, Praxisbeispiele. Spektrum Verlag, München, 2005.

[CM94] Chakravarthy, S.; Mishra, D.: Snoop: An Expressive Event Specification Language for Active Databases. Data and Knowledge Engineering 14(10), Oct. 1994.

[DBM88] Dayal, U.; Buchmann, A.; McCarthy, D.: Rules are Objects Too: A Knowledge Model for an Active Object-Oriented Database System. Proc. of the 2nd Internat. Workshop on Object-Oriented Database Systems, Bad Muenster, 1988.

[GK+08] Gmach, D.; Krompass, S.; Scholz, A.; Wimmer, M; Kemper, A.: Adaptive quality of service management for enterprise services. ACM Transactions on the Web 2(1), 2008.

[Gr04] Gray, J.: The Next Database Revolution. ACM SIGMOD Conference, Paris, 2004.

[HA+05] Halevy, A.; Ashish, N.; Bitton, D.; Carey, M.; Draper, D.; Pollock, J.; Rosenthal, A.; Sikka, V.: Enterprise information integration: successes, challenges and controversies. ACM SIGMOD Conference, Baltimore, 2005.

[He08] Henning, M.: The rise and fall of CORBA. Communication of the ACM 51(8), 2008.

[HLM88] Hsu, M.; Ladin, R.; McCarthy, D.: An Execution Model for Active Data Base Management Systems. Proc. of of the 3rd Internat. Conference on Data and Knowledge Bases, Jerusalem, 1988.

[Ho06] van Hoof, J.: How EDA extends SOA and why it is important. V6.0, 2006, http://soa-eda.blogspot.com.

[IBM10] IBM Websphere: What is a BRMS. http://www-01.ibm.com/software/ websphere/ products/business-rule-management/whatis/, retrieved on 24-08-2010.

[Jb10] JBoss Community: Drools 5 – The Business Logic Integration Platform, http://www.jboss.org/drools, retrieved on 24-08-2010.

[KLB96] Kudrass, T.; Loew, A.; Buchmann, A.: Active Object-Relational Mediators. Proc. of the 1st Internat. Conference on Cooperative Information Systems (CoopIS'96), Brussels, 1996.

[Ku97] Kudrass, T.: Aktive Mechanismen zur Konsistenzsicherung in Förderationen heterogener und autonomer Datenbanken (in German). Dissertation, infix Verlag, 1997.

[Ku03] Kudrass, T.: Describing Architectures Using RM-ODP. In: Kilov, H.; Baclawski, K. (Eds.): Practical Foundations of Business System Specifications, Kluwer Academic Publishers, 2003, p. 231-245.

[LCB99] Liebig, C.; Cilia, M.; Buchmann, A.: Event Composition in Time-dependent Distributed Systems. Proc. of the 4th Internat. Conference on Cooperative Information Systems (CoopIS'99), Edinburgh, Scotland, 1999.

[Lu02] Luckham, D.: The Power of Events: An Introduction to Complex Event Processing. In: Distributed Enterprise Systems, Addison-Wesley, 2002.

[Obj04] Object Management Group: CORBA Home Page, http://www.corba.org/, 2004.

[Obj09] Object Management Group BPMN 1.2 – Final Adopted Spccification, http://www.omg.org/spec/BPMN/1.2/PDF.

[RB01] Rahm, E.; Bernstein, P.: A survey of approaches to automatic schema matching. VLDB Journal 10(4), 2001.

[Ro03] Ross, R.: Principles of the Business Rule Approach. Addison Wesley, 2003.

[SL90] Sheth, A.; Larson, J.A.: Federated Database Systems for Managing Distributed Heterogeneous and Autonomous Databases. ACM Computing Surveys 22(1990) 3.

[SRK92] Sheth, A.; Rusinkiewicz, M.; Karabatis, G.: Using Polytransactions to Manage Interdependent Data. in: Elmagarmid, A. (Ed.): Database Transaction Models for Advanced Applications. Morgan Kaufmann Publ., 1992.

[Sun02] Sun Microsystems Inc. Java Message Service Specification Final Release 1.1, 2002.

[Sun99] Sun Microsystems Inc: Jini Architecture Specification – Revision 1.0, 1999.

[Wa98] Wang, R.: A Product Perspective on Total Data Quality Management. Communications of the ACM 41(2), 1998.

[WK06] Wolter, R.; Haselden, K: The What, Why, and How of Master Data Management. Microsoft Corp., http://msdn.microsoft.com/en-us/library/bb190163.aspx.

[Wo07] Wolter, R.: Master Data Management (MDM) Hub Architecture. Microsoft Corp., http://msdn.microsoft.com/en-us/library/bb410798.aspx.

[WZL01] Wang, R.; Ziad, M.; Lee, Y.: Data Quality. Kluwer, 2001.

[ZK96] Zimmermann, J.; Kudrass, T.: Advanced Database Systems: From Monoliths to Unbundled Components. 8th GI-Workshop on Foundations of Databases (GvD), Friedrichsbrunn, Germany, 1996.