An Active Functionality Service for E-Business Applications

M. Cilia and A. P. Buchmann
Databases and Distributed Systems Group, Department of Computer Science
Darmstadt University of Technology - Darmstadt, Germany
<lastname>@informatik.tu-darmstadt.de

Abstract

Service based architectures are a powerful approach to meet the fast evolution of business rules and the corresponding software. An active functionality service that detects events and involves the appropriate business rules is a critical component of such a service-based middleware architecture. In this paper we present an active functionality service that is capable of detecting events in heterogeneous environments, it uses an integral ontology-based approach for the semantic interpretation of heterogeneous events and data, and provides notifications through a publish/subscribe notification mechanism. The power of this approach is illustrated with the help of an auction application and through the personalization of car and driver portals in Internet-enabled vehicles.

1 Introduction

Two trends have become evident with the rise of e-commerce, a rapidly changing business environment and a tendency to satisfy these new requirements through a service-based architecture. Companies simply cannot avoid/ignore the fundamental problem that business requirements are changing faster than applications can be created and/or modified. Most of these requirements are related to business rules. Business rules are precise statements that describe, constrain and control the structure, operations and strategy of a business. They may be thought of as small pieces of knowledge about a business domain. They offer a way of encapsulating business semantics and promoting them to the surface in the same way that databases enable the separation of data from applications.

Traditionally, business rules have been scattered, hard-coded and replicated by different applications. Isolating the business rules from the application code enables developers to easily find and modify the pertinent rule(s) when a policy change is required. This provides the ability to quickly change rules without modifying the rest of the application code, thereby enhancing maintenance.

In recent years, one of the trends in database technology has focused on extending conventional database systems (DBMS) to enhance their functionality and to accommodate more advanced applications. One of these enhancements was extending database systems with powerful rule processing capabilities [10]. In their most general form, active database rules (also known as ECA-rules) consist of three parts: an **E***vent* causes the rule to be fired; a **C***ondition* is checked when the rule is fired; and an **A***ction* is executed when the rule is fired and its condition evaluates true. As it has been demonstrated, active databases are appropriate for business rule enforcement.

However, traditional active mechanisms have been designed for centralized systems and are monolithic, thus making it difficult to extend or adapt them to a new generation of distributed applications. New large-scale applications, such as EAI, e-commerce or Intranet applications impose new requirements. In this context, different organizations provide services, events and data are coming from diverse sources, and the execution of actions and evaluation of conditions may be performed on different (sub-)systems. Furthermore, events, conditions and actions may not be necessarily directly related to database operations. This leads to the question of why a full-fledged database system is required when only active functionality and some parts/services of a DBMS are used to coordinate the services.

Nowadays, the trend in the application space is moving away from tightly coupled systems and towards systems of loosely coupled, dynamically bound components. In such a context, it seems reasonable to move required active functionality outside of the active database system by offering a flexible service that runs decoupled from the database, and that can be combined in many different ways and used in a variety of environments. What seems to be appropriate for this is a service-based architecture in which an active functionality (ECA rule) service can be seen as a composition of other services, like complex event detection, condition evaluation, and action

execution. A similar approach, known as unbundling [7], places emphasis on unbundling active functionality into reusable components to later rebundle them according to the scenario in question. For distributed environments, the unbundling approach is inappropriate mainly because active databases were not conceived to take into account the inherent characteristics of distributed environments like, independent failures, message delays, the lack of a global time, and simultaneity of happenings/events. This has an impact not only on the complex event detector but also on the semantics of event operators [8].

Besides distribution, another important aspect to consider is the integration of events and data coming from heterogeneous sources. Combining data from different sources leads inevitably to problems if the meaning of the terms is not shared.

Ontologies play a fundamental role in this work. On the foundations of an ontology-based infrastructure, an active functionality service has been developed providing the following benefits:

- events from different sources are represented using common terms and additional contextual information,
- events are disseminated by means of a publish/subscribe mechanism which is adequate for distributed environments,
- services interact using an appropriate vocabulary at a semantic level,
- rule definition languages can be tailored for different domains using a conceptual representation, and
- the conceptual rule representation enables the use of a common active mechanism.

For the new generation of applications it is necessary to provide an active functionality service as an integral part of the middleware. This will help to develop applications that can be easily adapted to new business requirements. The approach is illustrated with examples based on online auctions and personalized car and driver portals.

The rest of this paper is organized as follows. In Section 2 the conceptual foundation of our flexible and extensible active functionality service is presented. Section 3 describes how business rules are defined. Section 4 presents scenarios where rules are used in an online auction environment and for personalized driver and car portals. Section 5 touches on the requirements of service based middleware platforms. Finally we present conclusions, address open issues and discuss future work.

2 Foundations

Three main pillars are the foundation of this work: an ontology-based infrastructure, event notifications, and a service-based architecture, which are presented in the following subsections. More details about this approach can be found in [5].

2.1 Ontology-based Infrastructure

Active functionality mechanisms in our context are fed with events coming from heterogeneous sources. These events encapsulate data that can only be properly interpreted when sufficient context information about its intended meaning is known.

This work is based on the use of shared concepts (ontologies) expressed through common vocabularies as a basis for interpretation of data and metadata. We represent events, or event content to be precise, using a self-describing data model, called MIX [3]. In the rest of the article we refer to events represented based on MIX, i.e. based on concepts from the common ontology, as *semantic events*. MIX refers to concepts from a domain-specific ontology to enable a semantically correct interpretation of events, and supports an explicit description of the underlying interpretation context. Semantic events from different sources can be integrated by converting them to a common semantic context using conversion functions defined in the ontology.

Ontologies in the scope of this work are extensible and they are organized at three different levels: a) the basic level, where elementary ontology functionality and physical representation is defined; b) the infrastructure level, where concepts of the active functionality domain and other aspects related to the infrastructure (i.e. notifications) are specified; and c) the domain-specific level, where concepts of the subject domain (e.g. online auctions) are defined [5].

2.2 Events and Notifications

An *event* is understood here as a happening of interest. Events coming from different applications are integrated by event *adapters*. They convert source-specific events into semantic events (represented by ontology-based concepts enriched with semantic contexts). A *notification* is a message reporting an event to interested consumers. A notification carries not only an event instance but also important operational data, such as reception time, detection time, event source, time to live, priority, etc.

A *notification service* based on a publish/subscribe paradigm is responsible for delivering events to interested consumers. Here a notification flows from an event producer to possibly a set of consumers. Subscribers (consumers) place a standing request for events by subscribing. On the other hand, a publisher makes information available for its subscribers. A publish/subscribe mechanism provides asynchronous communications, it natu-

rally decouples producers and consumers, it makes them anonymous to each other, it allows a dynamic number of publishers and subscribers, and provides location transparency without requiring a name service. By means of this notification service, subscriptions are made based on the concepts of the underlying ontology (concept-based addressing) and can include contextual information of an event of interest. The notification service is used to implement the communication among services that are involved in the ECA-rule processing.

2.3 Service-based ECA-rule Processing

In this work, traditional ECA processing is decomposed into its elementary and autonomous parts. These are responsible for complex event detection, condition evaluation, and action execution. Elementary services expose two kinds of generic and very simple interfaces: a) a *service interface* with a single method that receives an event notification as a parameter; b) a *configuration interface* is used for administration purposes, such as register, activate, deactivate, delete, etc. This simple service interface provides flexibility, allowing to configure the flow of service execution easily. ECA-rule processing is then realized as a combination/composition of these elementary services according to the rule definition. Interactions among services involved in the processing of a rule are based on the notification service.

But before processing rules, services must be configured for this purpose. The active functionality service offers the operations needed to define, remove, activate/deactivate, and search/browse ECA-rules. An ECA-rule Manager plays the role of a representative of the active functionality service. This means that activities related to registration, activation, deactivation, and deletion of rules are executed through this representative. The most complex process is the registration of a rule, which involves the composition of elementary services that participate in its processing. This composition consists of finding, contacting, and configuring them.

Once services are configured, the run-time phase follows, where rules are processed. When a triggering event is detected, the complex event detector publishes this happening. This means that all rules that were defined using this triggering event are automatically "fired" by means of a notification arrival. In this situation, no conflict resolution policy is needed because all rules are executed concurrently (other execution models are possible). When condition services that were configured are notified, they evaluate their predicate and if true, automatically notify the corresponding action services using the same notification service.

Figure 1 (middle and bottom) depicts the ECA-rule Manager when configuring services that were composed according to a rule specification.

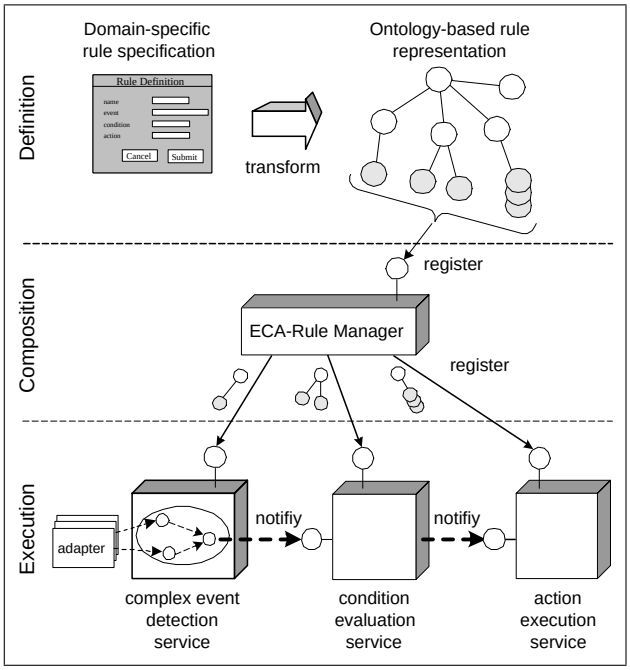

Figure 1: Putting it all together

3 Defining Rules

Rules must be clearly distinguished from two perspectives: how business rules are expressed by the users and how they are represented inside the system. Taking this into account, in this work the rule representation is organized in three layers. The *external* layer allows the possibility to tailor a rule definition language for each specific domain (or group of end-users) making convenient the specification of rules without the complications or levels of detail imposed by a generic rule definition language. The *conceptual* layer provides independence between the implementation of the underlying active mechanism and an end-user's rule definition. This layer uses an ontology-based representation of rules. Finally the *internal* layer enables the use of a "generic" active functionality service where components or services that are involved can be implemented using different optimization criteria or different programming languages, but they all "understand" the conceptual layer and they use an internal representation to process rules.

The conceptual representation enables the use of a "generic" active functionality service for different domains, making the underlying service independent from the rule specification.

Figure 1 illustrates the organization of rule representations presenting a web form for specifying a rule (external) and how they are transformed into a conceptual representation. This intermediary representation is then used for registration with the ECA Manager, which in turn decomposes it and registers its parts with the corresponding services. These take the high-level representation, configure themselves, and represent it using an internal data structure.

By means of an integral use of ontologies as part of the infrastructure, the definition of rules can benefit from the use of a context. Contexts can be associated to conditions and actions in order to evaluate them under the defined contextual information. For instance, a condition predicate that verifies distances can define ”metric system” as context. This way, incoming events from heterogeneous sources are first converted to the metric system (if necessary) before they are used for evaluation. Consequently, conditions and actions are always specified at a domain-specific level, and are independent from source-specific representations. This provides a very useful and powerful mechanism for combining events from heterogeneous sources.

4 Scenarios

4.1 Auctions

Auctions are a popular trading mechanism when multiple buyers compete for scarce resources. The advent of auction sites on the Internet, such as eBay or Yahoo has popularized the auction paradigm and has made it accessible to a broad public that can trade practically anything in a consumer to consumer interaction. The mechanism has become so popular that many e-businesses are using auction mechanisms to handle prices.

Tracking the objects that are auctioned is time consuming. Therefore, some form of notification mechanism is needed to alert a potential buyer when an item of interest comes on the market. Serious art collectors have used similar services for centuries. Agents or gallery owners notify a potential buyer whenever an article that might interest a customer becomes available. In the more mundane world of Internet-auctions collectors would like to enjoy a similar service. In addition, a collector might prefer to deal with one common auction portal instead of registering her interests with multiple auction sites. Therefore, we introduced the notion of a meta-auction [4]. A meta-auction allows a potential buyer to roam automatically and seamlessly across auction sites for auctions and items of interest.

To realize the meta-auction, several problems must be solved. We argue that today’s systems that are based primarily on user-initiated communication are not adequate and will not scale properly. The large number of interconnected users and systems, as well as their wide-area distribution imposes particular restrictions with respect to response time and network bandwidth. Internet-scale information systems therefore must leverage proactive information dissemination and caching techniques. However, typical client/server and n-tier system architectures are merely based on a request/response interaction and do not take into account the asymmetric nature of such systems [1], where the significant data flow is from a backend-tier to the application-tier which then provides access for end-users through a Web gateway.

Furthermore, the query metaphor from the database domain is currently the primary means for information acquisition, which results in the user polling for changes and happenings of interest. We argue that notifications about events, such as the placement of a highest bid, and their timely delivery to the user represent valuable information. Therefore, publish-subscribe as an additional interaction paradigm is needed to make the efficient dissemination of process-related information possible.

Each site participating in the meta-auction system provides information about items and the auction process but does not share a global data schema nor may we assume a global schema for notifications. Still, all participants come from the same application domain and at least conceptually, share a common vocabulary. While in most of today’s systems the vocabulary is left implicit, we propose an ontology-based infrastructure for explicit metadata-management on top of which the meta-auction service can be realized. The suggested ontology-based infrastructure provides common vocabularies for semantically meaningful exchange of data and notifications, and supports incremental integration of participating information systems as needed.

Consider the case of a collector. With the current auction sites, she has to manually search for the item of interest, possibly visiting more than one auction site. If successful, she might end up being engaged in different auctions at multiple auction sites. There are two obvious shortcomings to this approach: first, the user must poll for new information and might miss the window of opportunity, and second, the user must handle different auction sites with different category setups and different handlings. This motivates the need for the meta-auction broker, which.provides a unified view of different auction sites and services for category browsing, item search, auction participation and auction tracking.

Events that arise in the context of an auction process should be treated as first-class information and propagated

as notifications to the users who subscribed to the event. Propagation of events leads to a useful and efficient non-polling realization of an auction tracking service.

In this context, it is mandatory to cope with heterogeneity. Today, the exact meaning of terms, entities and notifications used by different auction sites is still left implicit. To enable the brokering between different participating auction sites, the precise understanding of the terms used by each site is needed and should be made explicit through a domain-specific common vocabulary. This is a prerequisite for semantically meaningful information exchange between a frequently changing set of independent participants in a large-scale business scenario.

As mentioned before, adapters resolve heterogeneities with regard to organization and structure of the data, and the use of different terms referring to the same real-world aspects. In addition, metadata is added to the available data to make implicit modeling assumptions concerning organization and meaning explicit. Based on this representation, heterogeneities in the semantics of the data, e.g. use of different units of measure, scale factors, derivation formulas, coding, or naming schema, can be resolved by the adapters at "integration" time.

Events related to the auction process are disseminated using the notification service described in Section 2.2. This way, publishers and subscribers use a semantic level of subscription which is common to all of them.

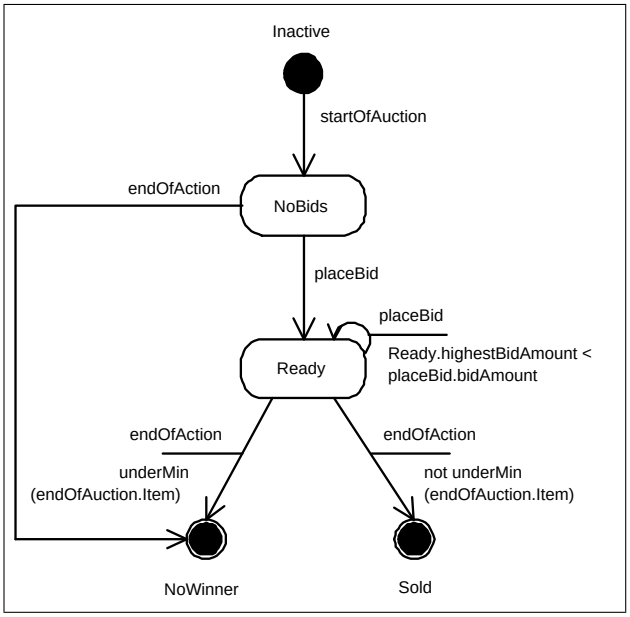

Figure 2: Statechart of a simple ascending auction process

The auction process itself can be defined using statecharts [2] and because they are event-driven, they can be easily implemented with ECA-rules. In this way, different sets of rules can describe different types of auction processes (ascending, reverse, dutch, etc.). Figure 2 shows a graphical representation of a simple ascending auction process. From this graphical definition ontology-based ECA-rules are generated (like the one depicted in Figure 3). The ECA-Rule Manager receives this rule definition and brakes it down into elementary elements (e.g. event, condition, action), searches for the corresponding services passing to them these elements for configuration. In particular, the rule shown in Figure 3 corresponds to the transition that reacts to the placement of a bid of a participant (placeBid) moving from the Ready state and back to the same state. Under these circumstances, and as a collateral effect, transitions from one state to another produce notifications, making the auction process available (for example, new highest bid, end of auction, etc.) to interested participants. As shown in figure 3, a context is associated to this rule, to which data are converted (if necessary) before any evaluation/comparison.

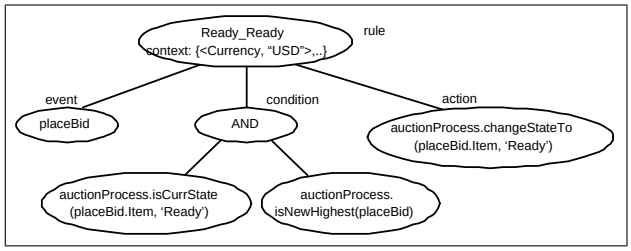

Figure 3: Graphical representation of an ontology-based rule

To track an item of interest during an auction process, for example to ascertain that another bidder has reached a highest bid, or that the deadline of an auction is approaching, an agent can be used. Here bidders can benefit from a rule service to program their own agents. In contrast to current agent bidders, where they are owned, controlled and implemented by the auction house, these agents can react to happenings of the auction process according to the bidders' strategy.

4.2 Personalized Car and Driver Portal

A service-based architecture is ideally suited for providing configurable applications. Personalized portals are a kind of application in which services can be added or deleted. In the case of vehicles and drivers, it is possible to provide location-based recommendations for repairs or fuel, personalized car settings, entertainment, navigation aids and scheduling information to name just a few. The recommendations can take into account the driver's pref-

erences and other requirements, such as company policies. We are experimenting with scenarios in which vehicles are equipped with Internet-access and connections for handheld devices, i.e. cell phones and PDA's. Such a scenario would be particularly attractive for "road warriors" who rent every day a car in a different city and want to emigrate their familiar environment but also used to carry part of it to a customer's location or interact with their home base.

5 Service-based Middleware and Pilot Implementation

Service based architectures combine a light-weight, encapsulated application code. To make service based architectures work, a middleware layer is needed so that it coordinates the invocation of services and reflects the business rules and policies [9]. Such a middleware layer must include monitoring and notification mechanisms. The active functionality service presented here is based on the foundation of a (loosely-coupled) service-oriented architecture, ontology-based infrastructure and the use of notifications to disseminate events.

Ontologies are used in this work as a common interpretation basis to enable semantically correct interpretation of events and notifications in open heterogeneous environments. Our ontology-based infrastructure applies homogeneously the ontology approach not only to integrate events from different sources but also to support the interaction among elementary services. Moreover, a conceptual representation of rules makes a high-level and domain-specific rule definition language possible providing independence between specification of rules and the active functionality mechanism. ECA-rule processing in our architecture is decomposed into elementary services. These services provide a very simple and generic interface, where parameters of methods are represented using the common ontology. Therefore, the flow of work through services can be easily configured – omission or inclusion of services like condition evaluation, event filtering or complex event detection is made easy. Notifications are used to carry events from their source to interested consumers (services), i.e. notifications are the means for services to interact. For this purpose, a notification service, based on a publish/subscribe mechanism using concept-based addressing, is employed. The use of this kind of mechanism is appropriate for loosely-coupled distributed systems. Because of this conceptual foundation, our architecture promotes flexibility, extensibility and integration for large-scale Internet-based applications.

A prototype of the active functionality service and its elementary services were developed using Java and they run on top of HP's Core Service Framework (CSF). Java was selected mainly because of code portability reasons since services may be necessary to run at different tiers and at different run-time configurations. Java is also used to specify ontology concepts and their relationships. Ontology support is completely implemented and the necessary ontology concepts of the infrastructure and the vehicle scenario are already defined. Our implementation also includes a running notification service that supports transactions and coupling modes, and an XML event adapter and an alarm service.

6 Conclusions and Future Work

The scenarios we have worked with have clearly shown the power of a service-oriented architecture. Among the critical mechanisms for a working service-oriented middleware platform, we identified the active functionality service, a publish/subscribe notification mechanism and the infrastructure for semantic interoperability.

In this article, a meta-auction scenario was discussed and it was shown how it benefits from using the active functionality service, in particular, the integration of data and events from heterogeneous sources by means of ontologies and the use of metadata. It must be noticed that in this scenario, notifications about the state of the auction process are considered first-class information, and its efficient delivery is considered required. Here, the auction process itself is described using ECA-rules and process-related happenings are automatically notified. In this context, information is disseminated by means of a publish/subscribe mechanism based on a vocabulary common to all participants. Additionally, bidders benefit from the use of such a service capturing their strategy using (ECA-)rules.

The same active functionality service was used in a different context and with the purpose of personalizing the user's experience in a vehicle scenario [6].

Current research involves profile organization and management and their relation with privacy issues. Additionally, we are investigating in more detail the complex event detection in open distributed environments. In particular certain issues are being investigated. First, consumption modes which should take into account partial order of events and how to cope with uncertainty of event order. Second, we are looking for a minimalistic set of (low-level) event operators, that let us define domain-specific powerful (high-level) event operators. Third, the extension of the timestamp ontology is required in order to correctly interpret and compare timestamps coming from dis-

tributed sources. Therefore, different time synchronization dimensions and event observation mechanisms must be studied and properly organized and represented. Last but not least, we are working together with other members of the department on scalability aspects of large-scale event dissemination and also on performance and capacity planning issues.

Acknowledgements

The authors would like to thank Christof Bornhövd, Fabio Casati, Umesh Dayal, Li-Jie Jin, Christoph Liebig, Memhet Sayal, and Ming-Chien Shan for their collaboration with this project. This work has been partially supported by Hewlett-Packard Laboratories and Hewlett-Packard German Innovation Center.

References

[1] S. Acharya, R. Alonso, M. Franklin, S. Zdonik. Broadcast Disks: Data Management for Asymetric Communications Environments. Proc. of SIGMOD, 1995.

[2] M. Benyoucef, R. Keller. An Evaluation of Formalisims for Negotiations in E-Commerce. Proc. Workshop on Distributed Communications on the Web, LNCS 1830, 2000.

[3] C. Bornhövd, A. Buchmann. A Prototype for Metadata-Based Integration of Internet Sources. Proc. CAiSE, LNCS 1626, 1999.

[4] C. Bornhövd, M. Cilia, C. Liebig, A. Buchmann. An Infrastructure for Meta-Auctions. Proc. WECWIS, 2000.

[5] M. Cilia, C. Bornhövd, A. Buchmann. Moving Active functionality from Centralized to Open Distributed Heterogeneous Environments. Proc. CoopIS, 2001.

[6] M. Cilia, A. Buchmann. Profiling and Internet Connectivity in Automotive Environments. Submitted for evaluation, SIGMOD'02 Conference.

[7] S. Gatziu, A. Koschel, G. von Bültzingsloewen, H. Fritschi. Unbundling active functionality. SIGMOD Record, 27(1):35-40, 1998.

[8] C. Liebig, M. Cilia, A. Buchmann. Event Composition in Time-dependent Distributed Systems. Proc. CoopIS, 1999.

[9] C. Liebig, M. Malva, A. Buchman. Integrating Notifications and Transactions: Concepts and X2TS Prototype. Proc. Engineering Distributed Objects, 2000.

[10] N. Paton (editor). Active Rules in Database Systems. Springer, 1999.