Dealing with Heterogeneous Data in Pub/Sub Systems: The Concept-Based Approach

M. Cilia*, M. Antollini†, C. Bornhövd‡, A. Buchmann

Databases and Distributed Systems, Dept. of Computer Science,
Technische Universität Darmstadt, Darmstadt, Germany
E-mail: $< lastname >$@informatik.tu-darmstadt.de

Abstract

The event-based approach is well suited for integrating autonomous components or applications into complex systems by means of exchanging events. Event-based systems need an event dissemination mechanism to deliver relevant events to interested consumers. The publish/subscribe interaction paradigm has been gaining relevance in this context. One of the main characteristics of publish/subscribe systems is that they decouple producers and consumers so that they can remain unknown to each other. Therefore, the consideration of data heterogeneity issues is fundamental. However, most pub/sub notification services do not support the interaction among heterogeneous event producers and consumers.

In this paper we describe the concept-based approach *as a high-level dissemination mechanism for distributed and heterogeneous event-based applications. It enhances the notification service by enabling it to pass semantic information across component, institutional or cultural boundaries. It provides a high level abstraction for a meaningful exchange of data within the pub/sub interaction paradigm.*

1. Introduction

The event-based approach is well suited for integrating autonomous components or applications into complex systems by means of exchanging events. Because event-based systems do not require a-priori knowledge about the consumers of events they evolve and scale easily.

The exchanged events encapsulate data about a given happening of interest, which can only be properly interpreted and used when sufficient context information is known. In traditional systems, this context information is typically known by users and developers and left implicit. It is normally lost when data and events are exchanged across component or institutional boundaries. To process events in a semantically meaningful way, explicit information about the semantics of events and data is required.

Event-based systems need an event dissemination mechanism (or notification service) to deliver relevant events to interested consumers. The publish/subscribe interaction paradigm has been gaining relevance for this purpose. It basically consists of a set of clients that asynchronously exchange events, decoupled by a notification service that is interposed between them. Clients can be characterized as producers or consumers. Producers publish notifications[1], and consumers subscribe to notifications by issuing subscriptions, which essentially are stateless message filters. Consumers can have multiple active subscriptions, and after a client has issued a subscription the notification service delivers all future matching notifications that are published by any producer until the client cancels the corresponding subscription.

Since publish/subscribe mechanisms decouple producers and consumers, they should share a common understanding in order to express their mutual interests. In other words, events must be understandable beyond the closed confines of a single component or application. That includes applications that interact across traditional borders regardless of economic, cultural or linguistic differences (e.g. in its simplest form system of units, currency or date/time format). Because the source of an event cannot anticipate who is interested in a given event and where and when it must be delivered, a higher-level publish/subscribe infrastructure is needed.

To the best of our knowledge (see Section 2 for details), existing publish/subscribe mechanisms only expose the *data structure* of events but not the explicit semantics. This reflects a low level support for event consumers that

*Also Faculty of Sciences, UNICEN, Tandil, Argentina
†Faculty of Sciences, UNICEN, Tandil, Argentina
‡SAP Corporate Research, Palo Alto. Christof.Bornhoevd@sap.com

[1] In the context of this paper the terms notification, message and event are used interchangeable to mean the same thing.

based on this scarce information must express their interest without an explicit description of the intended meaning of events, i.e., the implicit assumptions made by event/data producers. Without this kind of information event producers and consumers are expected to comply with implicit assumptions made by participating software components or applications. Even in the case of a very small set of applications within an enterprise this approach is questionable.

For publish/subscribe mechanisms to be effective in open environments they must

- provide for the usage of a common vocabulary for defining interests, and
- support the correct interpretation of information independently of its origin and place of consumption.

We proposed in [2] the use of the *concept-based approach* to support meta-auctions. We have generalized concept-based pub/sub into a high level dissemination mechanism for distributed and heterogeneous event-based applications. Its goal is to provide a higher level of abstraction to describe the interests of event producers and consumers. This is achieved by supporting from the ground up ontologies which provide the base for correct data and event interpretation. Rather than requiring every producer or consumer application to use the same homogeneous namespace (as is common in other pub/sub systems) we provide metadata and conversion functions to map from one context to another. This last feature allows event consumers to simply specify the context to which events need to be converted before they are delivered for client processing.

The rest of this paper is organized as follows. In section 2 related work is presented. Section 3 analyzes and characterizes the support of pub/sub systems for heterogeneous data exchange. The concept-based approach is described in Section 4 including the main characteristics of the data model used and the role adapters play in this context. Finally, conclusions and ongoing work are presented in Section 5.

2. Related Work

In recent years, academia and industry have concentrated on publish/subscribe mechanisms because they allow loosely coupled exchange of asynchronous notifications, facilitating extensibility and flexibility. The channel model has evolved to a more flexible subscription mechanism, known as subject-based addressing, where a subject is attached to each notification [15]. Subject-based addressing features a set of rules that define a uniform and static name space for messages and their destinations. This approach is inflexible if changes to the subject organization are required, implying fixes in all participating applications.

To improve expressiveness of the subscription model the content-based approach was proposed where predicates on the content of a notification can be used for subscriptions. This approach is more flexible but requires a more complex infrastructure [4, 14]. Many projects in this category concentrate on scalability issues in wide-area networks and on efficient algorithms and techniques for matching and routing notifications to reduce network traffic [16, 12, 9, 18]. Most of these approaches use simple boolean expressions as subscription patterns since more powerful expressions cannot be efficiently treated.

In [17] some of the problems mentioned in the introduction are pointed out. The proposed solution consists of the application of two steps: 1) relationship "resolution" (e.g. synonyms and generalization/specialization) and 2) application of mapping functions (if necessary). The first step is aided by a domain-specific ontology so that subscriptions issued by subscribers are then re-written by the centralized notification service according to the default vocabulary. Notice that subscribers are not aware of synonym resolution. The second step involves functions in the sense of the resolution of relationships which otherwise cannot be specified using the relationship solver (e.g. derivation of data).

This approach could work well when human beings are the consumers of notifications but not if subscriber applications act as consumers. This problem arises because when matching notifications are delivered to subscribers they would not expect some fields (specifically those renamed by the synonym solver or resolved via data derivation) in the message content.

As mentioned before, only the data structure of events is exposed to all participants. This puts in evidence the low level support regarding data exchange which directly impacts the broad usability of such an important piece of the communication infrastructure. The resolution of synonyms is a relevant issue but alone does not solve the problem of applications that publish and consume events messages in heterogeneous contexts. Even when a common vocabulary is used the data interpretation problem is still present.

3. Heterogeneous Interaction Panorama

In a pub/sub environment, where new participants can join and leave dynamically, where producers and consumers are unknown to each other and thus fomenting loosely-coupling interactions, the consideration of heterogeneity issues is fundamental. However, most pub/sub notification services do not support the interaction among heterogeneous event producers and consumers.

When trying to tackle the problem of data/event heterogeneity, two main issues need to be considered. The first one concentrates on the vocabulary shared by all participants. The second one relates to the contextual information of applications that produce and consume events. In the following subsections these issues are treated in more detail.

3.1. Sharing a Common Vocabulary

Algorithms used to match published messages and subscriptions are (mostly) based on a string comparison. Therefore, these algorithms require the notifications coming from the publisher side as well as subscriptions to be in the same vocabulary. Because many applications can participate, a richer vocabulary is needed that includes all specific (sub)vocabularies used. Ontologies [10], for instance, are good candidates for this purpose because they support various relationships among terms (e.g. synonyms, specialization/generalization). But notice that within the ontology a set of terms must be identified as the terms used for matching purposes. From now on these terms are referred here as "matching terms" or "matching vocabulary".

Possible differences among participants at the vocabulary level are solved by a vocabulary agreement module. This is responsible for applying resolution strategies based on the navigation of ontology/vocabulary relationships to find the appropriate matching terms.

This module is always located between a publisher or a subscriber application and the pub/sub matching algorithm. From an abstract perspective, it is the delimitation border that distinguishes between the vocabularies used by participating applications and the matching vocabulary used within the pub/sub notification service. But remember that subscriber applications do only know their own vocabulary so that the vocabulary agreement module needs to maintain the applied resolution strategies for each subscription. This is required since matching notifications need to be transformed back from the matching vocabulary to the one known by the consumer application.

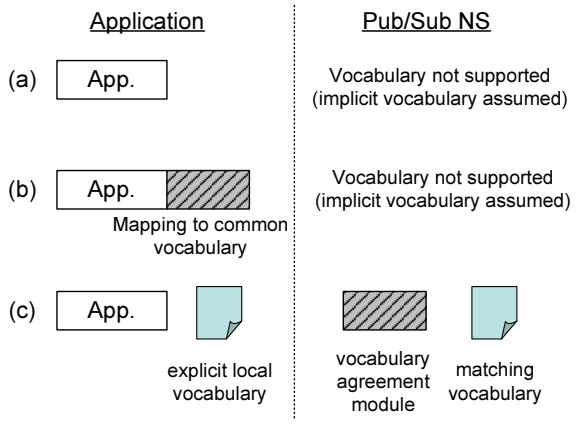

Figure 1. Common vocabulary treatment

Figure 1 shows three approaches how applications can deal with the vocabulary problem when interacting by means of a pub/sub notification service. Under the first approach (a) in the figure, applications implicitly assume per default a common vocabulary so that neither the pub/sub infrastructure nor applications explicitly care about this problem. Under the second approach (b), applications themselves internally resolve vocabulary issues while the pub/sub mechanism is unaware of them. The third approach (c) requires that applications can explicitly specify their vocabularies so that the pub/sub infrastructure uses this definition to feed the vocabulary agreement module. Notice that the notification service possesses also an explicit definition of the matching vocabulary that is used within the pub/sub boundaries for mapping purposes.

3.2. Considering the Context of Applications

Sharing a common vocabulary in pub/sub systems is a prerequisite but not the whole solution. The problem of dealing with heterogeneous data is still present even when using a common vocabulary. Relevant contextual information (i.e. the set of assumptions about data) is essential to allow the correct interpretation o the data in question. For instance, consider the common vocabulary terms DateOfIssue, FinalPrice, NetWeight, and Distance. Data associated to these terms require contextual information, like the date format, the currency and the units of measure, respectively. Human beings can often distinguish these cases based on the situation context while computers cannot.

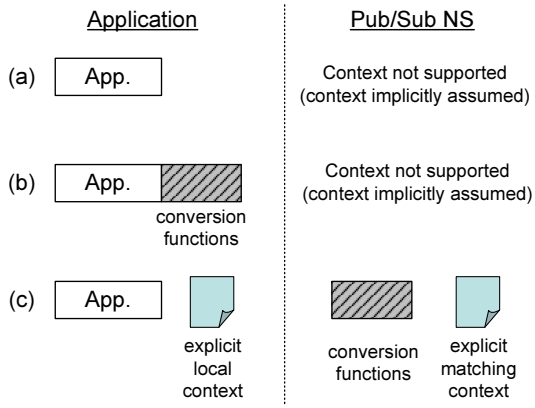

Figure 2. Consideration of application context

According to the handling of context, applications can be organized in three groups as depicted in Figure 2 (which by the way has a striking similarity to Figure 1). In the first group (a), participating applications do not consider contextual information of exchanged data and neither does the pub/sub notification service. Thus, an implicit context is assumed among all participants including the pub/sub infrastructure itself. The range of misinterpretation in this category is wide. In the second group (b), applications are context-aware and they explicitly convert data to and from a

common implicit context (assumed by the pub/sub system) when publishing and subscribing respectively. In this case, conversion functions are scattered in many applications. If at least one data producer incorporates terms implying not yet known conversion functions, then all consumers need to be modified to incorporate them. Finally, in the last group (c), the context of applications and the context assumed by the pub/sub system (henceforth *matching context*) are explicitly specified. With this, transformations from and to the common context can be automatically performed. These transformations can now be applied under the responsibility of the pub/sub infrastructure.

4. The Concept-based Approach

Originally introduced in [2] for a specific application and with an initial implementation [5] on top of a commercial pub/sub product, the concept-based approach was proposed to tackle the problem of event-based interaction among heterogeneous applications. On one hand it concentrates on resolving data interpretation problems. On the other hand it provides a pub/sub abstraction that can run on top of (various) notification services independent of the addressing model used underneath.

Towards these objectives it allows subscribers to express their interest considering their local context thus delivering to them as a result *ready-to-process* data which does not require further conversions. It also supports the enrichment of messages produced by publishers by adding their corresponding contextual information.

Figure 3 sketches this approach. The underlying notification service is used as a data delivery mechanism where the concept-based layer is responsible for providing a higher-level interaction among heterogeneous applications. A *vocabulary/ontology manager* is associated with this layer with the purpose of handling domain-specific vocabularies used by the participant applications. Additionally, this layer is in charge of mapping concept-based data into the data structures of the delivery mechanism underneath. *Adapters* are the intermediaries between pub/sub clients and the concept-based layer. At the publisher side they are responsible for resolving vocabulary issues and for enriching message content. At the subscriber side they are in charge of transforming/converting message content according to the subscriber preferences.

In order to incorporate a solution to the problem of data heterogeneity (or data integration) into a pub/sub system several crucial building blocks are needed: i) a shared vocabulary which is the consolidation of vocabularies used by all participating applications, ii) a matching vocabulary which is a subset of the shared one used internally by the pub/sub system for matching purposes, iii) a vocabulary agreement module that maps from the shared to the matching vocabulary, iv) explicit definition of contextual information of all participants including the pub/sub system, and v) conversion functions to map from one context to another.

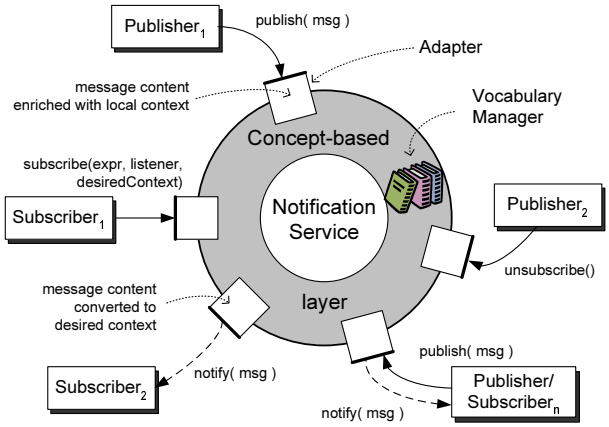

Figure 3. High-level view of the concept-based approach

From the previous section can be seen that context- and vocabulary-aware applications follow a similar pattern with respect to the infrastructure they need to support. The concept-based approach combines both into a model where vocabulary issues, the representation of context information and the conversion functions are under a uniform umbrella. Popular web standards such as RDF, DAML+OIL, and OWL provide adequate support for representing relationships among terms but do not support conversion function mechanisms as an integral part.

4.1. A Model for Data Exchange

To exchange data among loosely coupled systems, it must be taken into account that the structure and semantics of individual data items may vary, even if they describe objects of the same class of real world phenomena. Therefore, *context information* concerning the organization and meaning of data has to be given on an extensional level, i.e., on the level of data values. For this reason, we need description models that allow a flexible association of metadata with the available data items. In the following we describe MIX, the model used to represent event content.

Metadata based Integration model for data X-change [3, 1], or MIX for short, can be understood as a self-describing data model. This is because information about the structure and semantics of the data is not provided as a separately specified data schema, but is given as part of the available data itself. Thus, MIX allows a flexible association of context information in the form of metadata.

This model is based on the concept of a `SemanticOb-`

`ject`. It represents a data item together with its underlying `SemanticContext` which consists of a flexible set of meta-attributes that explicitly describe the implicit assumptions about the meaning of the data item. However, because we cannot explicitly describe all modelling assumptions the semantic context always has to be understood as a partial representation.

In addition, each semantic object has a concept *label* associated with it that specifies the relationship between the object and the real world aspects it describes. These labels have to be taken from a commonly known vocabulary, or `Ontology`. In the MIX model, an ontology is a finite set of concepts and their relationships.

It must be noticed that the `SemanticContext` is also specified by referring to the ontology/vocabulary. The association of context information with a given data value serves as an explicit specification of the implicit meaning of the data. This allows the determination of semantically equivalent semantic objects even if they are represented differently, i.e., relative to different contexts. For example

<NetWeight, 1234, { *<Unit, "kilogram(s)" >*} *>* and

<NetWeight, 2720.50, { *<Unit,"pound(s)" >*} *>*

are semantically equivalent, because they represent the same information and we can specify a *conversion function* by which one representation can be transformed into the other. Such conversion functions are a prerequisite for the integration of semantic objects coming from different sources, by converting these objects, as far as possible, to a common context. In detail, two kinds of conversion functions can be distinguished. The first are those attached to the concept definition. The second are those defined in a function conversion manager (because, for instance, they may change over time, e.g. currency conversion).

This approach provides the central management of conversion functions and the flexibility to extend them if necessary. This avoids scattering conversion functions that in other cases need to be embedded in participating applications.

4.2. Adapters

Adapters are the contact/mediator of applications with the notification service. They maintain the different views of client applications by enriching the ontology with: i) the corresponding contextual information, and ii) the mapping from their local vocabularies to the matching one.

From the view point of client applications, the first task is the vocabulary agreement. Here, synonyms within the ontology are resolved. For consumer applications in particular, subscriptions are rewritten considering other relationships, like generalization/specialization. The applied resolution strategies need to be preserved since they are fundamental to map back the content of incoming notifications. Remember that consumer applications can only process messages expressed in their local vocabulary.

The vocabulary agreement module can also be statically defined if the kind of message content applications publish and/or subscribe to is stable. Adapters are not necessarily located at the client side. In this sense, configuration information is passed (at bootstrap- or connection-time) to the notification service for setting up vocabulary agreements for a particular client application and as a consequence the ontology is enhanced with this information.

The second task relates to the context information. In the case of data producers, adapters are responsible for enriching message content with meta information. This is done based on the explicit specification of clients about their contextual information. In this case, the ontology is enriched with it for the use of this particular application. Thus, this information is used when publishing events (by simply looking up in the ontology the required contextual information that needs to be added to this particular attribute of the message). When incoming notifications/messages arrive they are converted according to the context specified for each term as defined in the enriched ontology.

Adapters can also be configured to minimize the context information attached to each message. For this purpose, an addressable context can be defined so that participant applications explicitly state their contexts. Messages can now simply refer to such a context instead of putting all relevant context information as part of the payload of messages. This is referred to as *context sharing*. Conversion functions are aware of this feature and can resolve this kind of references.

4.3. Building on Top of Delivery Mechanisms

The concept-based approach provides an abstraction where participating applications do not care about details of the underlying delivery mechanism. In this sense, applications involved in such interaction can replace this mechanism by relying on the concept-based approach causing minimal changes on applications if any. This allows, for instance, the migration from a topic-based dissemination (à la JMS) to a content-based without major effort.

From the publisher's point of view, the message content (represented with MIX) is mapped into the corresponding data structures of the underlying dissemination mechanism. The concept-based software layer is responsible for dealing with the "features" that the underlying delivery mechanism does not support. For details about the mapping into different addressing models see [6] for subject-based addressing and [7] for content-based addressing.

In many cases during the mapping to the underlying data structures data is converted according to the matching context defined by the notification service. The Rebeca framework [13] was specialized in order to delay the conversion

as much as possible by enabling message routers to automatically apply conversion functions on demand to appropriately evaluate expressions.

5. Conclusions and Ongoing Work

The concept-based approach enhances the scope of use of notification services by enabling it to cross component, institutional or cultural boundaries. It provides a higher-level of abstraction for a meaningful exchange of data within the pub/sub interaction paradigm. The concept-based software layer is responsible for resolving data heterogeneity problems by minimizing misinterpretation of exchanged data. This is achieved by integrating the use of ontologies, meta-data information and conversion functions provided by MIX. A notification service based on the concept-based approach delivers *ready-to-process* notification to subscriber applications so that no further data conversions are needed. This layer runs on top of various (commercial) notification services (such as, Sun's JMS reference implementation, JMS/JBoss, WebSphere MQ, OpenJMS, Rebeca, TIB/Rendezvous).

This approach was successfully used in research projects like Internet-enabled vehicles [8] and the meta-auction approach [2].

The MIX implementation used in this prototype is based on a pure java development while an OWL-based implementation is almost ready-to-use [11].

A centralized administration user interface for the concept-based layer is under development. Through this interface a common vocabulary can be defined, the ontology can be edited, new conversion functions can be defined, the matching context can be specified, and the mapping strategies into the underlying addressing model can be configured.

Currently, a framework for composite events is under development with the purpose of supporting event correlation in a variety of environments. Additionally and in order to encompass the OWL implementation of MIX we are experimenting with different XML routing strategies within the Rebeca framework.

References

[1] C. Bornhövd and A. Buchmann. Semantically Meaningful Data Exchange in Loosely Coupled Environments. In *Proc. Intl Conf on Information Systems Analysis and Synthesis (ISAS'00)*, July 2000.

[2] C. Bornhövd, M. Cilia, C. Liebig, and A. Buchmann. An Infrastructure for Meta-Auctions. In *Proceedings of WECWIS*, pages 21–30. IEEE Computer Society, June 2000.

[3] C. Bornhövd. *Semantic Metadata for the Integration of Heterogeneous Internet Data (in German)*. Ph.D. Thesis, Department of Computer Science, Darmstadt University of Technology, Germany, July 2000.

[4] A. Carzaniga, D. R. Rosenblum, and A. L. Wolf. Challenges for Distributed Event Services: Scalability vs. Expressiveness. In *Engineering Distributed Objects (EDO'99)*, Los Angeles, CA, May 1999.

[5] M. Cilia, C. Bornhövd, and A. Buchmann. Moving Active functionality from Centralized to Open Distributed Heterogeneous Environments. In *Proceedings of CoopIS*, volume 2172 of *LNCS*, pages 195–210, Trento, Italy, Sept. 2001.

[6] M. Cilia, C. Bornhövd, and A. Buchmann. CREAM: An Infrastructure for Distributed, Heterogeneous Event-based Applications. In *Proceedings of CoopIS*, volume 2172 of *LNCS*, Italy, Nov. 2003. Springer.

[7] M. Cilia, C. Bornhövd, and A. Buchmann. Event Handling for the Universal Enterprise (to appear). *Information Technology and Management (ITM), Special Issue on Universal Global Integration*, Jan. 2005.

[8] M. Cilia, P. Hasselmeyer, and A. Buchmann. Profiling and Internet Connectivity in Automotive Environments. (demo paper). In *Proceedings of VLDB*, pages 1071–1074, 2002.

[9] F. Fabret, F. Llirbat, J. Pereira, A. Jacobsen, K. Ross, and D. Shasha. Filtering Algorithms and Implementation for Very Fast Publish/Subscribe. In *Proceedings of ACM SIGMOD*, pages 115–126, 2001.

[10] T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. *Int. Journal of Human-Computer Studies (IJHCS)*, 43(5/6):907–928, 1995.

[11] P. Kabus. Implementing semantic data integration for the internet (in german). MSc Thesis, Department of Computer Science, Darmstadt University of Technology, Germany, Oct. 2003.

[12] G. Mühl, L. Fiege, and A. Buchmann. Filter Similarities in Content-Based Publish/Subscribe Systems. In *International Conference on Architecture of Computing Systems (ARCS)*, volume 2299 of *LNCS*, pages 224–238. Springer, 2002.

[13] G. Mühl and L. Fiege. The REBECA Notification Service, 2001. `http://www.gkec.informatik.tu-darmstadt.de/rebeca/`.

[14] G. Mühl. *Large-Scale Content-Based Publish/Subscribe Systems*. PhD thesis, Darmstadt University of Technology, Germany, Sept. 2002.

[15] B. Oki, M. Pfluegl, A. Siegel, and D. Skeen. The Information Bus – An Architecture for Extensible Distributed Systems. In *Proceedings of SIGOPS*, pages 58–68, 1993.

[16] L. Opyrchal, M. Astley, J. Auerbach, G. Banavar, R. Strom, and D. Sturman. Exploiting IP Multicast in Content-based Publish-Subscribe Systems. In *Proceedings of Middleware*, volume 1795 of *LNCS*, pages 185–207. Springer, 2000.

[17] M. Petrovic, I. Burcea, and H.-A. Jacobsen. S-ToPSS: Semantic Toronto Publish/Subscribe System. (demo paper). In *Proceedings of VLDB*, pages 1101–1104, Sept. 2003.

[18] P. Pietzuch and J. Bacon. Hermes: A distributed event-based middleware architecture. In *In Proceedings of the 1st International Workshop on Distributed Event-Based Systems (DEBS'02)*, Vienna, Austria, July 2002.