EXTENDING WEB INFORMATION SYSTEMS WITH WEB NOTIFICATION SERVICE

A. Stanimirović, D. Stojanović, L. Stoimenov

Faculty of Electronic Engineering, University of Niš

Beogradska 14, 18000 Niš, Serbian and Montenegro

{alex, dragans, leni}@elfak.ni.ac.yu

Abstract: In this paper present a new architecture of Web Notification Service. Web notification service is a component that monitors events and after occurrence of specified event sends notification. Proposed architecture is based on Active Databases and Web Services technologies. Also a new model for notifications was developed. Basic idea of this model was to provide generalized infrastructure that can easily integrated in different applications. At the same time notification service must provide extensible architecture in order to meet specific needs of different applications. Using this model we developed Web Notification service as extension of existing GeoJP application.

Keywords: event, notification service, active systems, Web service, publish/subscribe, notifications

1 Introduction

Modern Web based information systems generally are based on request/response Web mechanism where user have to explicitly specify request for information/service and system send response based on such request. Large amount of information available on Web has dynamic nature. This causes that user have to permanently issue request in order to get up-to-date information. An emerging building block for such systems is an infrastructure called **publish/subscribe event notification services** where user sends request once and is notified whenever an event occurs that changes information of interest.

Until recently, most event notification systems were intended to be used as part of specific applications or in localized settings, such as a single machine, a building, or a organization. Recently the need for global event notification system, that can dynamically connect changing sets of clients and services, is emerged.

Notification Service is a component that monitors events in the system on behalf of user (man or machine), and after detecting occurrence of a specific event, service sends notification to user or another component by e-mail, SMS, IM or WAP push according to user preferences and profile.

In this paper we focus on our approach to implementation of Notification service. The rest of paper is structured as follows. In section 2 we discuss related technologies that are base for implementation of Notification service. Section 3 explains model that we used for our implementation of Notification service. Section 4 describes architecture of our Notification service. Section 5 describes extension of existing component GIS application with Notification service.

2 Related work

For the past few years Web services technology has dramatically changed the way that Internet works. Before Web services Internet was dominated by program-to-users or business-to-consumer (B2C) communications. Nowadays, thanks to web services technology, program-to-program business-to-business (B2B) becomes predominate way of communicating over Internet [Gottschalk, 2002]. This new model is built on top of existing and emerged standards such as Hyper Text Transfer Protocol (HTTP), Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language and others.

Web services can be defined as loose coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols [Gottschalk, 2002]. Web service is interface that describes a collection of operations that are network-accessible through standardize XML messaging. A Web service performs specific task or a set of tasks. A Web service is described using standard, formal XML notation that provides all detail necessary for interaction with service.

Web services are not implemented in monolithic way but rather represent a set of related technologies. As a minimum any Web service must support a connection between two applications for exchanging queries and responses in XML over HTTP. Figure 1, bellow, represent Web services technology stack. Core layers, that define basic Web services communications have been widely accepted. High-level layers are remaining open question.

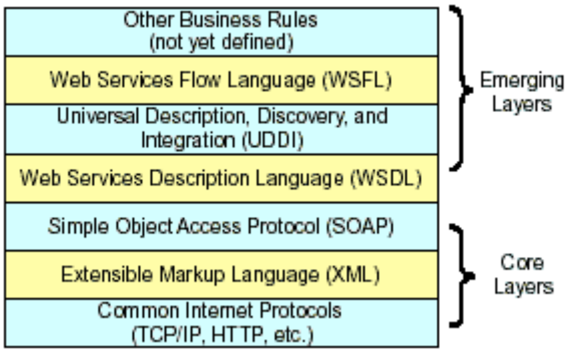
Other Business Rules
(not yet defined)
Web Services Flow Language (WSFL)
Universal Description, Discovery, and Integration (UDDI)
Web Services Description Language (WSDL)
Simple Object Access Protocol (SOAP)
Extensible Markup Language (XML)
Common Internet Protocols
(TCP/IP, HTTP, etc.)
Emerging Layers
Core Layers

Figure 1: Web services technology stack

Traditional database management systems (DBMSs) are passive i.e. queries, updates and transactions are executed only when explicitly requested. For many applications, however, it is important to monitor conditions defined over the database state and to take actions (possibly subject to timing constraint) when the state of underlying database changes. A database system with such capabilities is an *active database system* i.e. it is able to react to events [Bertino, 2001].

The desired behavior of active database system is expressed in form of production rules (also called event-condition-action rules or ECA rules), which are defined and stored in database along with the data of interest [Dayal, 1994]. This has the benefit that rules can be shared among many applications, and that the database system can optimize their implementation.

These rules have the following form:

```
ON event
IF condition
THEN action
```

A rule in ECA model has three components [Elmasri]:

- The event (or events) that trigger the rule. These events are usually database update operations but in general model they can also be temporal events or some kind of external events.
- The condition that determines whether the rule action should be executed. If condition is evaluated to true the action is executed.
- The action to be taken. The action is usually sequence of SQL statements but according to general model it can be any kind of database operations or some kind of an external program that will be automatically executed.

Rules that specify actions that are automatically triggered by certain database events are considered as very important enhancements of commercial database systems. The concept of triggers, technique for specifying some types of active rules, has been part of the earliest specifications of the SQL for relational databases. Commercial relational database solutions have various versions of triggers available.

Microsoft SQL Server 2000 defines triggers as special type of stored procedure that automatically takes effect when data in specified table is modified [Microsoft, 2000]. A trigger is invoked in response to INSERT, UPDATE and DELETE statements. A trigger can query other tables and in his action part can include very complex SQL statements. The trigger and the statement that fires it are treated as single transaction, which can be rolled back from within the trigger. If a severe error is detected the transaction is automatically rolled back.

3 Notification service model

Notification service is an event-based system that supports detection of important events and the delivery of notifications from one source to number of interested parties. The event notification service supports asynchronous, one-to-many, communication between numbers of entities [Gupta, 1998].

The event paradigm is based on three entities:

- Subscribers – entities that uses interfaces provided by Notification service to subscribe for receiving notifications about interesting events,
- Producers –entities that produce events and use interfaces provided by Notification service to publish events they are producing,
- Notification service – service that detects published events and sends notification to subscribed clients. This service hides implementation details from entities involved in communication and also hides the identity of entities.

Basic goal of solution for event notification service was to provide generalized infrastructure that can easily integrated in different applications [Tarkoma, 2002]. In order to achieve this goal only solution was to design event notification service as small and autonomous component that can be easily adopted by existing component systems. At the same time our event notification service must provide extensible architecture in order to meet specific needs of different applications.

Our solution for event notification service was designed using Web services architecture. Notification service is web service that describes interface that can be used by subscribers or producers. Subscribers use methods of this interface to subscribe for receiving notifications for events of interest. Producers use methods, defined by this interface, in order to publish their events and to send published events to Notification service. In order to achieve this functionality Notification service uses XML messaging and standard HTTP protocols for communication with subscribers and producers. Figure 2 shows basics of these communications.

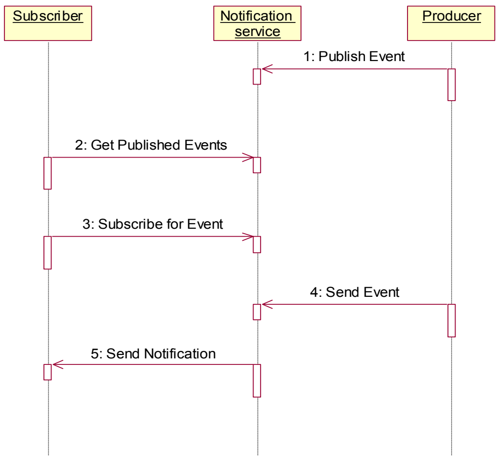

Figure 2: Notification service basic interface

Producer can publish all kinds of different events. The most common events are updates of database data (SQL INSERT, UPDATE and DELETE operations over data in some RDBMS). In order to publish events producer sends appropriate message to Notification service. This XML message is formatted according to defined XML schema. XML schema for publishing XML message is shown on Figure 3.

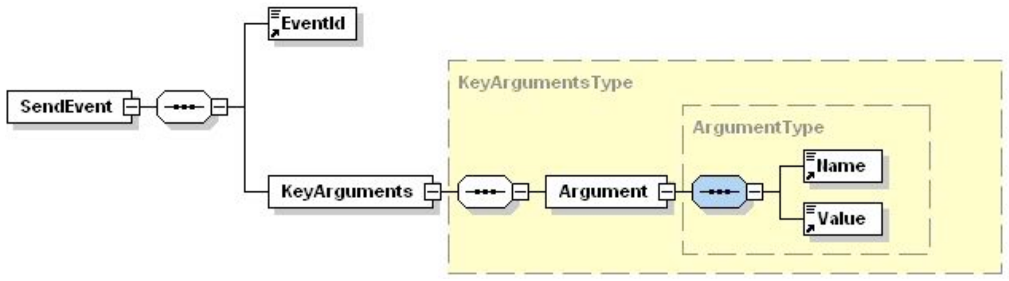

Figure 3: XML schema for Publish event XML message

For published event producer receives an event id that is generated by Notification service and that producer will use later to send event to Notification service. Schema for SendEvent XML message is shown on Figure 4.

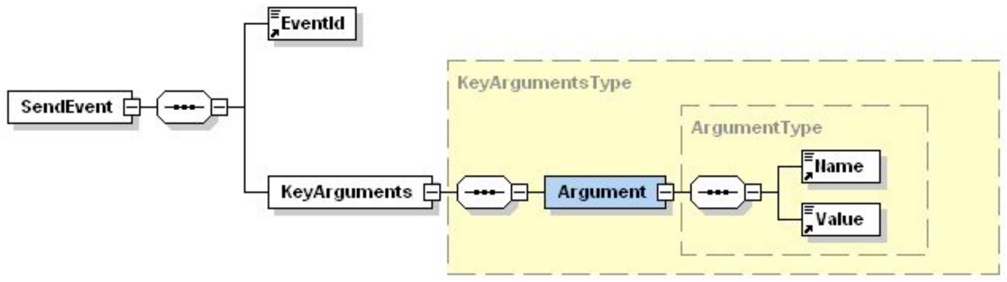

Figure 4: XML schema for SendEvent XML message

Subscriber can retrieve list of published events. Based on this list of published events subscriber can choose events that he is interested in. To subscribe for event subscriber must send to Notification service appropriate XML message. Appropriate XML schema for SubscribeEvent XML message is shown on Figure 5.

4 Notification service architecture

The high level architecture of the system is presented in Figure 6. This architecture consists of three modules:

- Monitoring events module,
- Notification delivery module and
- User interface module.

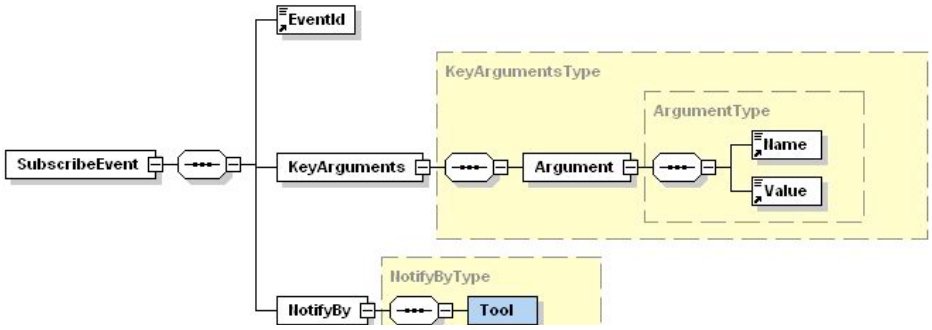

Figure 5: XML schema for SubscribeEvent XML message

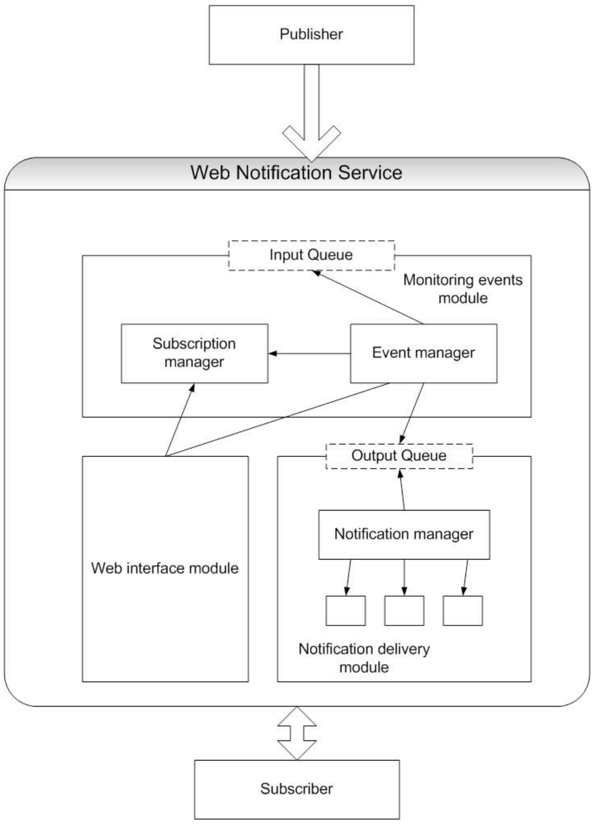

Figure 6: High-level architecture of Web Notification Service

Monitoring events module is a central component of notification service. This module implements the core functionality of the Notification service. It consists of two sub modules:

- Subscription manager. This sub module is responsible for dealing with subscription requests. Subscriptions are defined as XML messages according to XML schema explained earlier. This message defines event of interest and event notification options. Subscription manager validates this message and extract subscription info that is contained in the message. Subscription manager also keeps tracks of all subscriptions that Notification service received.

- Event manager. This sub module is responsible for dealing with received Publish requests. Event manager parses Publish XML message and extract all available event info. It also keeps records of all published events. Event manager sub module also process all received events from Input Que. For every received event this sub module checks if it is published earlier. If that is the case Event manager checks with Subscription manager if there is any subscription for that event. If subscription exists Event manager sends notification request to delivery queue.

Notification delivery module is module responsible for delivering notifications. This module is implemented using plug-in architecture. For every notification device (mobile phone, e-mail client, IM client, WAP enabled device) a plug-in will be developed. This kind of architecture allows new plug-ins easily to be added later. This module processes notification request from delivery queue and formats messages that will be send according to subscription info. Notification delivery module sends message using appropriate plug-in.

Web based user interface module allows human users to subscribe for receiving notification of events they are interested in. It has two basic functionalities:

- to provide to user information of all published events
- to collect from user all info necessary for subscription.

5 Implementation

The notification service is currently under implementation in the GeoJP geographic information system for Niš Municipality. GeoJP information system is developing as a part of project IT.1.23.0249A supported by Ministry of Science, Technology and Development of Republic of Serbia. The basic purpose of this information system is to keep records about public property on the territory of Niš Municipality.

Notification service extends the functionality of GeoJP providing possibility to notify citizens about events they are interested in (e.g. request is processed, time to pay taxes, time to make new bids, time to sign new contract, new regulations, new taxes etc.). This functionality is also in accordance with e-government solution adopted by Niš Municipality which basic idea is to provide service to citizens “any time - any place”.

In this case there is only one events producer in the system. It is MS SQL Server instance that manages GeoJP database. Active facilities of MS SQL server are used for sending events to Notification service. Set of events is predefined and triggers that will send these events are implemented. Example trigger is shown on Figure 7. Also a special extended procedure xp_SendEvent is implemented in order to send events to Notification service.

```
CREATE TRIGGER TAKSA_SEND_NOTIFICATION ON [dbo].[Takse]
FOR Update
AS
EXEC xp_SendEvent( 12 )
```

Figure 7: Trigger for sending events

6 Conclusion

In this work we presented solution for Web Notification Service. In order to do that new model for notifications was developed. This model provides generalized architecture that can be easily integrated in existing Web Information Systems. This model also provides architecture that can be easily extended in order to meet specific needs of different Information systems. Using this model a Web Notification Service component was implemented in order to extend existing GeoJP application with notifications.

Successful implementation and deployment of Notification Service will enable its integration in other Web based information systems such as: location based services for tourist guide, e-Commerce solutions for stock market monitoring, military exercises, etc.

7 Acknowledgements

This research was partially supported by the project “Geographic Information System for Local Authorities based on Internet/WWW Technologies”, funded by Ministry of Science, Technology and Development, Republic of Serbia, Contract No. IT.1.23.0249A.

8 References

1. E. Bertino, B. Catania, G. P. Zarri, Intelligent Database Systems, Addison Wesley, London, 2001
2. U. Dayal, E. N. Hanson, J. Widdom, Active Database Systems, Interoperability and Beyond, Addison Wesley, Reading Massachusetts, Sep. 1994
3. R. Elmasri, S. Navathe, Fundamentals of Database systems, 3rd 2000
4. S. Gupta, J. Hartkopf, S. Ramaswamy, Event Notifier, a Pattern for Event Notifications, Java Report, July 1998
5. K. Gottschalk, S. Graham, H. Kreger, J. Snell, Introduction to Web Services Architecture, IBM System Journal, July 2002
6. Microsoft Corporation, Microsoft SQL Server 2000 Books Online, Microsoft Press, 2000
7. S. Tarkoma, Scalable Internet Event Notification Architecture (Siena), Research Seminar on Middleware for Mobile Computing, 2002