Management and Production Engineering Review
Volume 5 • Number 1 • March 2014 • pp. 28–31
DOI: 10.2478/mper-2014-0004

ENTERPRISE SERVICE BUS ARCHITECTURE FOR THE BIG DATA SYSTEMS

Cezary Orłowski[1], Edward Szczerbicki[2], Jan Grabowski[1]

[1] *IBM Center of Advanced Studies in Campus*
[2] *University of Newcastle, Callaghan University Drive, Callaghan NSW 2308, Australia*

Corresponding author:
Cezary Orłowski
Gdańsk University of Technology
Narutowicza 11/12 80-823 Gdańsk, Poland
phone: +48 58 347 13 19
e-mail: cor@zie.pg.gda.pl

Received: 25 October 2013
Accepted: 5 January 2014

Abstract
This paper presents the construction of the enterprise service bus architecture in data processing resources for a big data decision-making system for the City Hall in Gdansk. The first part presents the key processes of bus developing: the installation of developing environment, the database connection, the flow mechanism and data presentation. Developing processes were supported by models: KPI (Key Processes Identifier) and SOP (Simple Operating Procedures) (also connected to the bus). The summary indicates the problems of the bus construction, especially processes of routing, conversion, and handling events.

Keywords
enterprise service bus, service oriented architecture, big data system, smart cities, transformation and conversion processes.

Introduction

Development of IT systems processing big data requires architecture design which integrates on the one hand, the data on the other- creates the conditions for application integration. Half of the last decade was dominated by the approach in which the integration of resources was supported by data warehouse.

In the era of systems based on big data agglomerations have the problems of the substantial resources integration. These resources stored in heterogeneous databases derived from measurement networks. For example, Geneva [1] collects data on hundreds TB size originating from the measurement noise, pollution, traffic and current data from monitoring networks.

In turn, the city in the United States as Boston or New York [2] in principle take the scattering data for later analysis for decision support systems. They accept that the data collection of the big data at the present stage of development of Smart Cities systems are not able to processing it.

Information technology capable of processing such significant data are interested in of building the efficient solutions for the needs of cities, both in collecting and processing the data. These solutions are based on multilayer architectures and applications, focused on advanced GIS (ang. *Geographical Information Systems*). Most of them are based on the IT integration buses. The scope, shape and mechanism of those buses depend on the size and range of the data.

The article was analyzed bus integration combined with the verification environment to check how the verification environment determines the efficiency of those buses.

The application integration took place through the creation of layers of middleware, *Enterprise Application Integration Platforms*, protocols, API (*Application Protocol Interface*) or individual design GUI (*Graphical User Interface*). These solutions were not generic, and in case of problems of integration of enterprise information systems, it was time-consuming to implement and impossible to develop [4, 5].

That is why a growing number of applications is built on the basis of architectural design integration bus ESB (*Enterprise Service Bus*). ESB bus is a service-oriented platform for connecting applications created basing on various technologies, incom-

patible formats, data resources, and communication protocols. The advantage of this solution is primarily its dynamic conversion and data transformation (*dynamic data transformation and conversion)*, distributed communication and intelligent routing services.

For this reason, before the construction of a system for the City Council, decisions about the selection of future system architecture had to be made. Given the advantages of ESB, a SOA (*Service Oriented Architecture*) supported by ESB was chosen. A subsequent specification of requirements for the new system and data architecture was designed basing on MS SQL environment and applications on the basis of RAS (*Rational Software Architect*). The bus integration was built on the *WebSphere Message Broker Toolkit*. The presentation layer was shown in IOC (*Intelligent operating system*).

The article ends with the authors conclusions and observations focusing on the problems of both the construction of bus model and its implementation as well as the implementation of the system in the IOC.

Installation of developing environment

The IOC system is delivered (the development version), as images of five logical servers on the *VMware vSphere* platform (operating system *Linux Red Hat 5*). This complex development environment within which, development tools are divided into logical servers. This division of development tools on the one hand facilitates the process of software development because it allows developers to focus on selected tools. On the other hand, requires knowledge of the full developer environment to find and use the tools needed for the development process [3–6].

First *ioc15install* logical server is used primarily by the installation of the IOC [7, 8]. It is later used in the work related to software development. The second server *ioc15db* facilities is determined as a database server infrastructure and has a database tool *.db2 and environmental processes for their preparation and monitoring. Additionally *ioc15db* server creates conditions for the collection and processing of any databases resources. The third *ioc15event* server is most important one from the point of developing view and IOC application design. Its purpose is to support the ESB and ensure the flow of events through the bus. *Ioc15event* server is also responsible for connection to external data sources (such as the ones used in the ESB design of the database *.SQL).

This server includes development tools supporting the flow and events through the ESB bus. It contains four essential developing tools: *WebSphere Message Broker* used to design ESB architecture bus, *WebSphere Message Queue* supporting the flow data (*messaging handling*) and two other tools for communication and event processing: *Tivoli Netcool Impact* – supporting event handling and *Tivoli Netcool Omnibus* - supporting the process (*event processing and enhancing*). The development environment is complemented by two others servers, four application server*ioc15app* offering access to the GUI for IOC system and the fifth *ioc15mgmt* supporting developing and configuration management environment, its start and stop, also monitoring the status of the work of individual IOC modules.

Requirements for the IOC system

The IOC system to be built is an intelligent city management system which (due to its functionality) uses data collected from multiple sources (from the environmental monitoring, industrial network resources, crisis management repository (cameras, security systems, early warning systems and others)). In case of Gdansk requirements analysis (focusing on the pollutants and their use) needs a precise description of the design concept of data acquisition and processing as well as the construction of a decision support system [1, 9]. In the process of requirement analysis the developers used two approaches different from the point of view of the processes of construction of databases and their acquisition. The first approach assumed direct power system IOC data from project partners. The second approach was based on building a data warehouse fed by data from an external database project partners. Before choosing the solution both approaches were tested.

The first approach required both the analysis of different standards of databases and possibilities of data acquiring. Therefore two experiments were carried out. The first, in which data requirements and their standards were specified (ARMAAG – information about pollution, the City Council-noise data, Gdansk University of Technology – pollution and weather conditions) and attempts were made to supply IOC database in a parallel manner.

It soon turned out that parallel supply is very difficult to achieve. Therefore, it was decided to supply sequential data (using the second approach), thus evaluating usefulness of the data and the mechanisms of their acquisition. In the case of sequential processing (second approach), obtaining data was possible but there was a problem of sequential repetition of such supply. To ensure repeatability of the supply, both wholesale and IOC processes were developed.

Another important task was to build the foundation of the system database server. As it was sug-

gested at the beginning, the database system could have been placed on the same server on which IOC was set. However, it turned out that both in terms of testing processes and the subsequent development of the system, a much better solution (also from the point of view of safety of the system) was putting the database system on an a server external for the IOC.

The standard data warehousing was analyzed as a last problem. Taken into account the two standards *.SQL and *.DB2. From the point of view of the IOC the better solution (supplying the requirements of the IOC) was the standard DB2 databases. Taken into account the experience of our team, for which the standard DB2 was not known. Therefore, it was decided to standard SQL, although more development seemed to be the DB2 [10, 11]. The decision on this standard of the more well-known than the more predictive stemmed from the need to reduce project risk.

Integration bus architecture

The integration bus architecture of a decision support system is shown in Fig. 1. It consists of three layers of data flow. The data base input node connects database integration bus (in our case, the data warehouse akwilon2) [12]. The Mapping node converts the data from the data warehouse into integration bus protocol CAP (*Common Alerting Protocol*). The MQ Output node (presentation layer) is to put the event (recorded in CAP) in the queue manager ESB integration bus. Applications of*IBM WebSphere Message Broker Toolkit* (bus construction) and *Netcool Impact* (posting events on the IOC system map) were the environmental implementation of the system architecture [2, 13, 14].

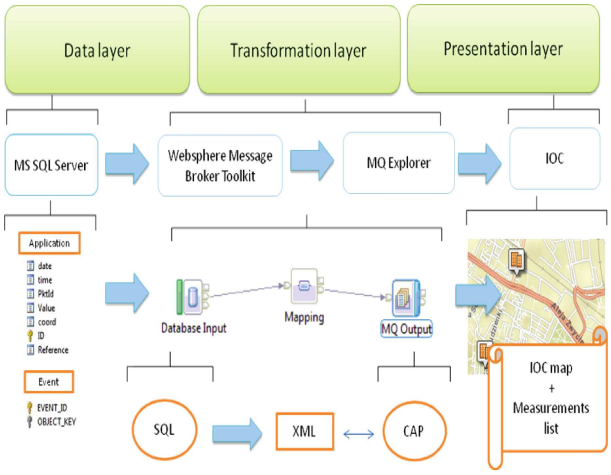

Fig. 1. The integration bus architecture for IOC system.

The data flow is illustrated by the example of the use of resources (data layer). At this layer it is checked whether the table in the database *IOC_appliaction* Akwilon2 finds rows that contain data from 85 stations of noise. If so, the trigger (data base element) of new records is activated *IOCAPPL* (Fig. 2) (insert command), which creates records with the same ID in table *IOC_Event*. Tables *IOC_event* and *IOC_application* are connected by the relationship with a primary key ID. Figure 2 shows a fragment of the code trigger which allows operations on rows of both tables.

```
USE [AkwilonData]
GO
/****** Object:  Trigger [dbo].[IOCAPPL]
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[IOCAPPL]
   ON  [dbo].[IOC_application]
   AFTER  INSERT
AS
BEGIN

    SET NOCOUNT ON;

 DELETE FROM dbo.IOC_event
    INSERT INTO [dbo].IOC_event
           ([OBJECT_KEY])
     select INSERTED.id from INSERTED

END
```

Fig. 2. A fragment of the code IOCAPPL trigger allowing operations on rows of both tables.

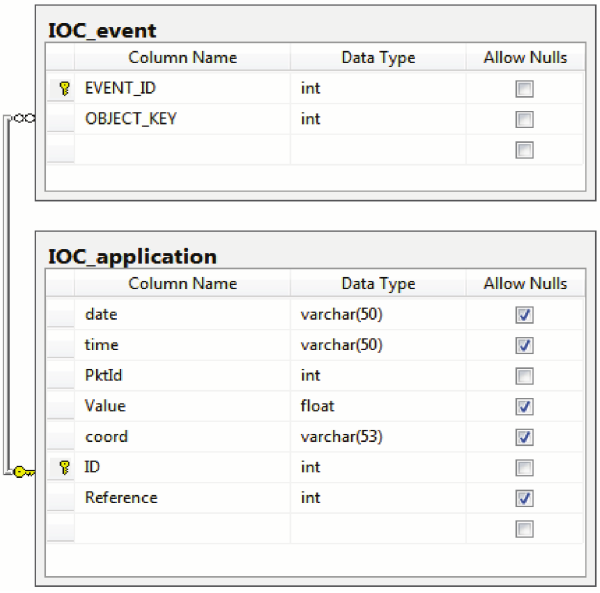

Fig. 3. The database tables akwilon2 (*ioc_event* and *IOC aplliaction*).

Conclusions

The paper presents the design of ESB integration bus architecture of IOC decision support system for the City of Gdansk. During the construction of the bus two groups of issues had to be dealt with: creation and maintenance of data flows and the use of development environment for the implementation and modification of bus architecture.

In the first case (start and maintaining the flow) the problems mainly focused on the conversion of data e.g. how to supply ESB. Another problem concerned the use of processes: *move*, *assign* and *concept* in data mapping processes. While the *move* allowed the simple processes to *convert* the data to the CAP protocol, process *assign*, especially concept not always supported the conversion process. Similar problems occurred in the processes of the event queue. Repeated attempts to refresh the event filled the base and did not allow correct action trigger database rows. Therefore, documentation development environment was often used to solve these problems.

In the second case the application of the development environment *WebSphere Message Broker Toolkit* aided the construction of the bus but not all in layers of the proposed architecture. While the high level architecture of ESB bus was relatively easy to build, including developers applications for the implementations of data flow processes was far more complicated e.g. For example, frequent problems with the construction of models, KPIs and SOPs. Since both models could be built in two ways: using *WebSphere Message Broker Toolkit* or *WebSphere Business Monitor* development toolkit, the selection of the tool did not always guarantee the correctness of the construction of the model. As in the first case, the documentation had often to be referred to. This continuous analysis of documentation slowed down the process of constructing decision support system in IOC project.

While the process of developing IOC system for processing big data for decision-makers: the City of Gdansk, was a relatively complex, the process of introducing changes to the system turned out to be relatively simple. Development environment which was difficult to install provided a multi-layered process of change. It has been shown (as is the case of other complex developing environments) that the work on the process of the installation of the development environment is quickly reflected by its performance. Therefore, the example of the construction of ESB for the City Council might be considered positive.

References

[1] Czarnecki A., Orłowski C., *Ontology as a tool for the IT management standards support Agent and Multi-Agent Systems*, Technologies and Applications, 2009, pp. 330–339.

[2] *IBM Intelligent operations center Information Center*, http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp, (2013).

[3] Orłowski C., Kowalczuk Z., *Knowledge management based on dynamic and self-adjusting fuzzy models*, in Knowledge-Based Intelligent Information and Engineering Systems, Springer Berlin Heidelberg, 2006.

[4] Orłowski C., Ziółkowski A., Czarnecki A., *Validation of an agent and ontology-based information technology assessment system*, Cybernetics and Systems: An International Journal, 41 (1), 62–74, 2011.

[5] Orłowski C., Rule-based model for selecting integration technologies for Smart Cities systems // Cybernetics and Systems, 2014 – in press.

[6] Pastuszak J., Stolarek M., Orłowski C., *Concept of generic IT organization evolution Model*, Faculty of ETI Annals, Information Technologies, 18, 235–40, 2008.

[7] Bhowmick A., *IBM Intelligent Operations Center for Smarter Cities Administration Guide* 5, Event flow diagnostic and validation tool for IBM WebSphere Business Monitor, International Business Machines Corporation 2009.

[8] Common Alerting Protocol Version 1.2 http://docs.osasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.pdf, (2013).

[9] Czarnecki A., Orłowski C., Sitek T., Ziółkowski A., *Information technology assessment using a functional prototype of the agent based system*, Foundations of Control and Management Sciences, 2009, pp. 7–28.

[10] Snadach K., *Graphical data presentatation in IBM Intelligent Operations Center*, Diploma Dissertation, Gdańsk, 2013.

[11] Smith A.D., *IBM Intelligent Operations Center KPI Implementers Guide for Websphere Software*, Document version 1.0.

[12] Kortas K. *Data integration using ESB – IBM Websphere Message Broker*, Diploma dissertation, Gdańsk, 2013.

[13] *IBM WebSpher application server information center*, http://publip.boulder.ibm.com/infocenter/cities/v1r5m0/index.jsp, (2013).

[14] *IBM WebSphere Broker Message Broker Information Center*, http://publib.boulder.ibm.com/infocenter/wmbhelp/v7r0m0/index.jsp, (2013).