Z E S Z Y T Y N A U K O W E P O L I T E C H N I K I P O Z N A Ń S K I E J
Nr 63 Organizacja i Zarządzanie 2014

Cezary ORŁOWSKI*

SYSTEMY *SMART CITIES* – STUDIUM PRZYPADKU

W artykule przedstawiono architekturę szyny integracyjnej wykorzystywanej w budowie systemu informatycznego przetwarzającego potężne zasoby danych na potrzeby podejmowania decyzji w Urzędzie Miejskim w Gdańsku. Na wstępie omówiono kluczowe z punktu widzenia wytwarzania szyny procesy: instalacji środowiska wytwarzania, podłączenia bazy danych, opracowania mechanizmów przepływu oraz prezentacji danych. Procesy prezentacji wsparto modelami KPI (ang. *key processes identifier*) oraz SOP (ang. *simple operating procedures*) (także podłączonymi do szyny). W podsumowaniu wskazano na problemy budowy szyn integracyjnych, a zwłaszcza procesów: trasowania, konwersji i obsługi zdarzeń.

Słowa kluczowe: systemy *Smart Cities*, systemy oparte na wiedzy, *Service Oriented Architecture*, *Enterprise Service Bus*, *Software as a Service*

1. WPROWADZENIE

W związku z 70. rocznicą urodzin Pana Profesora Leszka Pacholskiego, planując referat do specjalnego numeru „Zeszytów Naukowych Politechniki Poznańskiej”, zastanawiałem się nad jego tematyką. Mogłem poświęcić go ergonomii systemów informatycznych lub tematyce z pogranicza informatyki i zarządzania, ale po rozmowach z Profesorem zdecydowałem się poświęcić ten artykuł tematyce, która go intryguje – systemom opartym na wiedzy (ang. *knowledge based systems*) [1, 2], przetwarzającym potężne zasoby informatyczne (ang. *big data*) [6]. Pragnąłem także, aby zakres tej pracy obejmował procesy zarządzania, stąd artykuł poświęcony jest projektowaniu systemów opartych na wiedzy dla systemów *Smart*

* Gdansk IBM Center for Advanced Studies on Campus.

Cities. Przedstawiono w nim przykład systemu jako usługi (ang. *Software as a Service*), jak też architektury systemu integracji usług SOA (ang. *Software Oriented Architecture*).

Budowa systemów informatycznych przetwarzających znaczne dane wymaga architektury, która z jednej strony integruje te dane, ale przede wszystkim stwarza warunki do integracji aplikacji [3, 4, 5]. Do połowy poprzedniej dekady dominowało podejście, w ramach którego integracja zasobów następowała z wykorzystaniem hurtowni danych aplikacji przez (1) tworzenie warstw *middleware*, (2) zastosowanie rozwiązań *Enterprise Application Integration*, (3) tworzenie protokołów API (ang. *Applictaion Protocol Interface*) bądź (4) projektowanie indywidualnych interfejsów GUI (ang. *Graphical User Interface*). W przypadku wielu problemów związanych z integracyjnymi systemami informatycznymi przedsiębiorstw wdrożenie tych rozwiązań było czasochłonne, a one same nie stwarzały możliwości rozwoju.

Dlatego też coraz większa grupa aplikacji tworzona jest z wykorzystaniem architektury szyny integracyjnej ESB (ang. *Enterprise Service Bus*). ESB to zorientowana na usługi platforma łącząca aplikacje oparte na zróżnicowanych technologiach, niekompatybilnych formatach i zasobach danych oraz protokołach komunikacyjnych [7, 8]. Zaletą tego rozwiązania jest przede wszystkim dynamiczna konwersja i transformacja danych (ang. *dynamic data transformation and conversion*), rozproszona komunikacja (ang. *distributed communication*) oraz inteligentne trasowanie usług (ang. *inteligent service routing*) [13].

Z tego też powodu przed przystąpieniem do budowy systemu na potrzeby Urzędu Miejskiego podjęto decyzję o wyborze architektury systemu opartego na SOA i wspartego ESB. Kolejno określono wymagania wobec systemu, zaprojektowano architekturę danych z wykorzystaniem środowiska MS oraz aplikacji opartej na RAS (*ang. Rational Software Architect*). Szynę integracyjną zbudowano, wykorzystując Brocket Toolkit. System informatyczny zobrazowano z zastosowaniem IOC (ang. *Intelligent Oprating Center*). Podsumowaniem artykułu są wnioski i spostrzeżenia autora skoncentrowane zarówno na problemach budowy modelu szyny i jej implementacji, jak i na implementacji systemu IOC.

2. WYMAGANIA WOBEC SYSTEMU IOC

Proponowany system IOC to inteligentny system zarządzania miastem, który (z racji swoich funkcji) wykorzystuje dane pobierane z wielu źródeł (z sieci monitoringu środowiska, z zasobów sieci przemysłowych, z repozytorium zarządzania kryzysowego (kamer, systemów bezpieczeństwa, systemów wczesnego ostrzegania i innych)). W przypadku Gdańska analiza wymagań dotyczących zanieczyszczeń, koncepcji projektu oraz budowy systemu wspomagania decyzji wymagała precy-

zyjnego opisu stosownych baz danych. W procesie analizy wymagań brano pod uwagę zarówno wymagania, jak i dostępne dane, stosując dwa podejścia, odmienne z punktu widzenia procesów budowy baz danych i ich zasilania. W pierwszym podejściu zakładano bezpośrednie zasilanie systemu IOC danymi od partnerów projektów. Drugie podejście polegało na budowie hurtowni danych zasilanej z poziomu zewnętrznych baz danych partnerów projektowych. Przed wyborem rozwiązania przeprowadzono testy obu podejść.

Przypadek pierwszy wymagał z jednej strony analizy różnych standardów baz danych, ale z drugiej strony możliwości zasilania budowanego systemu danymi o zróżnicowanym standardzie. Dlatego też przeprowadzono dwa eksperymenty. W ramach pierwszego wyspecyfikowano wymagania dotyczące danych i ich standardu (Armaag – dane dotyczące zanieczyszczeń, Urząd Miejski – dane dotyczące hałasu, Politechnika Gdańska – zanieczyszczenia i warunki atmosferyczne) i próbowano równolegle zasilać bazę danych IOC [9, 10].

Stosunkowo szybko okazało się, że zasilanie równoległe jest bardzo trudne do uzyskania i dlatego zdecydowano się na wstępne zasilanie danymi wsadowymi (podejście II), aby ocenić przydatność danych, a następnie mechanizmy ich pozyskiwania. W przypadku przetwarzania wsadowego nie stwierdzono problemów z uzyskiwaniem danych, ale pojawiał się problem powtarzalności takiego wsadowego zasilania. Opracowano także procesy zasilania zarówno hurtowni, jak i IOC.

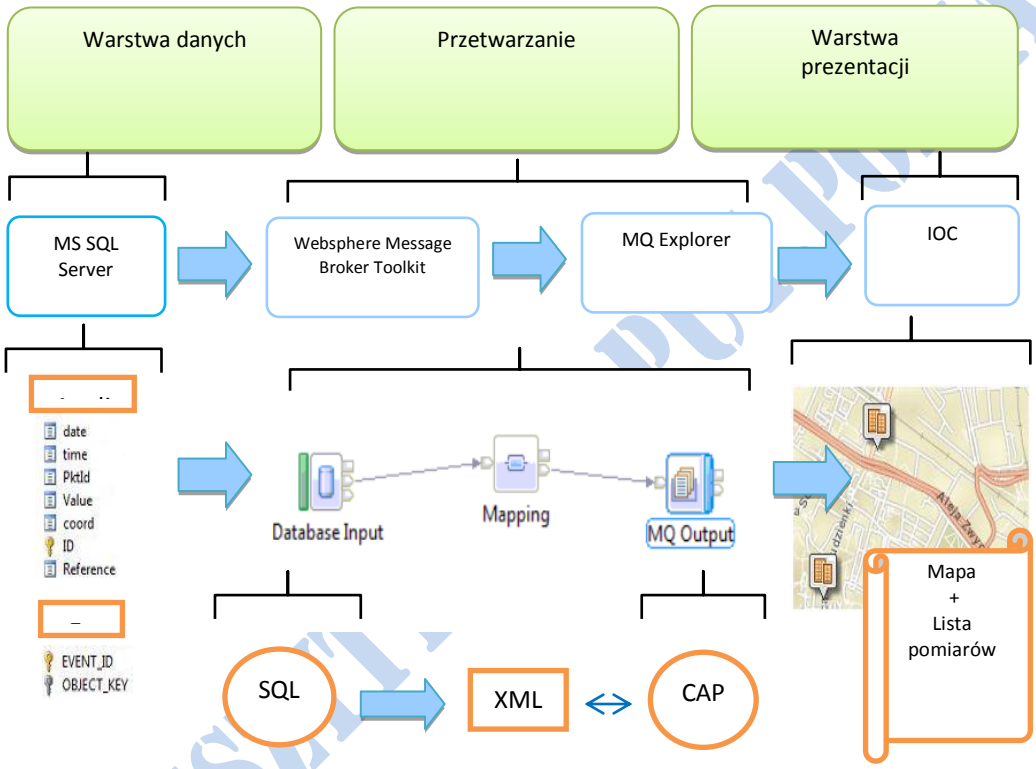
Kolejnym ważnym pytaniem było usytuowanie systemu baz danych. Na wstępie sugerowano, aby system baz danych był ulokowany na tym samym serwerze, na którym postawiono IOC. Okazało się jednak, że zarówno z punktu widzenia procesów testowania, jak i późniejszego wytwarzania, zdecydowanie lepszym rozwiązaniem (z punktu widzenia bezpieczeństwa) jest usytuowanie systemu baz danych na serwerze zewnętrznym w stosunku do IOC.

Ostatnim problemem była kwestia standardu hurtowni baz danych. Brano pod uwagę dwa standardy, *. SQL oraz *. DB2. Z punktu widzenia IOC lepszym (spełniającym wymagania IOC) rozwiązaniem był standard baz danych DB2. Brano jednak pod uwagę doświadczenie członków zespołu, którym standard DB2 nie był znany. Dlatego też zdecydowano się na standard SQL, mimo że bardziej rozwojowy wydawał się DB2. Decyzja o wyborze standardu bardziej znanego, a nie bardziej rozwojowego, wynikała z konieczności ograniczenia ryzyka projektowego.

3. ARCHITEKTURA SZYNY INTEGRACYJNEJ

Architekturę systemu przedstawiono na rys. 1. Składa się ona z trzech warstw, tworzących szynę integracyjną zapewniającą przepływ danych. Warstwa danych (węzeł *database input node*) umożliwia podłączenie do szyny integracyjnej bazy danych (w rozpatrywanym przypadku – hurtownie danych *akwilon2*). Warstwa

przetwarzania (węzeł *mapping node*) konwertuje dane z hurtowni danych do protokołu szyny integracyjnej CAP (ang. *common alerting protocol*). Z kolei węzeł *MQ qutput node* (warstwa prezentacji) ma za zadanie umieścić zdarzenia w formacie CAP w menedżerze kolejek szyny integracyjnej. Środowiskiem implementacji architektury systemu była aplikacja firmy IBM *WebSphere Message Broker Toolkit* (konstrukcja szyny) oraz *Netcool Impact* (umieszczanie zdarzeń na mapie systemu IOC) [12].



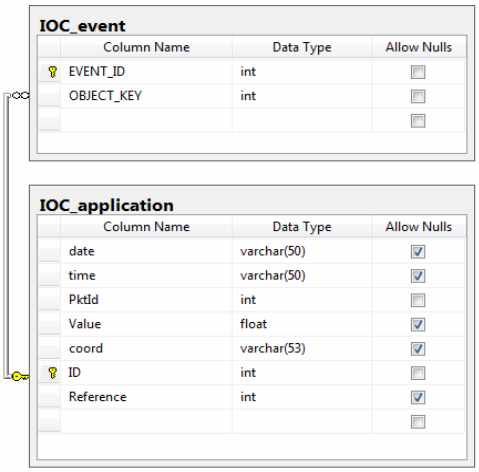
Rys. 1. Szyna integracyjna systemu zapewniająca przepływ danych na potrzeby systemu IOC

Złożony proces przepływu danych został zobrazowany na przykładzie wykorzystania zasobów (warstwa danych). Na poziomie tej warstwy sprawdza się, czy w tabeli *IOC_appliaction* w bazie danych *akwilon2* znajdują się wiersze zawierające dane z 85 stacji pomiaru hałasu. Jeżeli tak, to uruchamiany jest wyzwalacz *IOCAPPL* (rys. 2) nowych rekordów (polecenie insert), który jednocześnie tworzy rekordy z tym samym ID w tabeli *IOC_Event*. Tabele *IOC_event* i *IOC_application* połączone są relacją poprzez klucz główny ID [11]. Na rysunku 2 przedstawiono fragment kodu wyzwalacza umożliwiającego operacje na wierszach obu tabel.

```
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
```

Rys. 2. Fragment kodu wyzwalacza umożliwiającego operacje na wierszach obu tabel

Na rysunku 3 przedstawiono obie tabele bazy danych *akwilon2*.



Rys. 3. Tabele bazy danych systemu *akwilon2* (*ioc_event* oraz *IOC aplliaction*)

4. WNIOSKI

W artykule przedstawiono architekturę oraz implementację szyny integracyjnej ESB na potrzeby systemu IOC. W trakcie budowy szyny ESB napotkano dwie grupy problemów: uruchomienia i utrzymania przepływów danych oraz wykorzystania środowiska deweloperskiego do implementacji i modyfikacji architektury szyny.

W pierwszym przypadku uruchomienie i utrzymanie przepływów wiążą się głównie z problemami z konwersją danych i ze stale pojawiającym się pytaniem – czy szynę zasilać niezależnie, czy też poprzez hurtownie danych. Kolejnym problemem okazało się zastosowanie procesów: *move*, *assign* i *concept* w procesach mapowania danych. O ile proste procesy *move* umożliwiały konwersję danych do protokołu CAP, o tyle proces *assign*, a w szczególności *concept*, nie zawsze wspierał proces konwersji. Podobne problemy pojawiały się w procesach kolejkowania zdarzeń. Wielokrotne próby odświeżania zdarzeń zapełniały bazę i nie pozwalały na poprawne działanie wyzwalacza wierszy bazy danych. Dlatego też wielokrotnie sięgano po dokumentację środowiska deweloperskiego, aby usunąć pojawiające się problemy.

W drugim przypadku zastosowanie środowiska deweloperskiego *Websphere Message Broker Toolkit* umożliwiło konstruowanie szyny, ale nie wszystkich warstw proponowanej architektury. O ile architektura *high level* szyny ESB była stosunkowo prosta, o tyle włączanie kolejnych aplikacji deweloperskich nie zawsze wspierało implementację przepływu. Na przykład stosunkowo często występowały problemy z budową modeli KPI lub SOP. Ponieważ oba modele można było tworzyć na dwa sposoby – z wykorzystaniem *Websphere Message Broker Toolkit* lub *Websphere Business Monitor Development Toolkit*, nie zawsze wybór narzędzia gwarantował poprawność modelu. Podobnie jak w pierwszym przypadku, wielokrotnie sięgano do dokumentacji. Ciągłe zatrzymywanie projektu zmniejszało tempo jego realizacji.

Należy jednak podkreślić, że o ile proces budowy systemu IOC przetwarzającego potężne dane na potrzeby decydentów Urzędu Miejskiego w Gdańsku okazał się stosunkowo złożony, o tyle proces zmian systemu jest stosunkowo prosty. Środowisko deweloperskie jest trudne w instalacji, zapewnia jednak wielowątkową realizację procesu zmian. Dlatego też nakład pracy przeznaczony na proces instalacji środowiska szybko przekłada się na wydajność tego środowiska w procesach zmian budowanego systemu IOC. Dlatego też przedstawiony przykład budowy szyny integracyjnej ESB na potrzeby Urzędu Miejskiego należy uznać za pozytywny.

LITERATURA

[1] 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.

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

[3] 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, s. 7-28.
[4] Czarnecki A., Orłowski C., Ontology as a tool for the IT management standards support Agent and Multi-Agent Systems: Technologies and Applications, 2009, s. 330-339.
[5] IBM Intelligent operations center Information Center, http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp (dostęp: 2013).
[6] IBM WebSpher application server information center, http://publip.boulder.ibm.com/infocenter/cities/v1r5m0/index.jsp (dostęp: 2013).
[7] IBM WebSphere Broker Message Broker Information Center, http://publib.boulder.ibm.com/infocenter/wmbhelp/v7r0m0/index.jsp (dostęp: 2013).
[8] Kortas K., Data integration using ESB – IBM Websphere Message Broker, Diploma dissertation, Gdańsk 2013.
[9] 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.
[10] Orłowski C., Ziółkowski A., Czarnecki A., Validation of an agent and ontology-based information technology assessment system, Cybernetics and Systems: An International Journal, 2011, Vol. 41 (1), s. 62-74.
[11] Pastuszak J., Stolarek M., Orłowski C., Concept of generic IT organization evolution Model, Faculty of ETI Annals, Information Technologies, 2008, Vol. 18, s. 235-240.
[12] Snadach K., Graphical data presentatation in IBM Intelligent Operations Center, Diploma Dissertation, Gdańsk, 2013.
[13] Smith A.D., IBM Intelligent Operations Center KPI Implementers Guide for Websphere Software, Document version 1.0.

***SMART CITIES* SYSTEMS – A CASE STUDY**

Summary

This paper presents the architecture of an enterprise service bus used in the construction of information systems processing large amounts of data for decision-making needs at the City Hall in Gdansk. The first part presents the key processes of bus development: installation of developing environment, database connection, flow mechanisms and data presentation. Developing processes were supported by models such as KPI (*Key Processes Identifier*) and SOP (*Simple Operating Procedures*) (also connected to the bus). The summary indicates problems in bus construction, especially processes such as routing, conversion, and handling of events.