JCSI 1 (2016) 30-33
Journal of Computer Sciences Institute

Wysłane: 2016-09-12
Przyjęte: 2016-09-19

Analiza możliwości wykorzystania architektury SOA do integracji systemów informatycznych

Paweł Waręciak*, Piotr Kopniak
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem niniejszej pracy jest pokazanie różnych metod integracyjnych systemów informatycznych oraz koncepji SOA. Porównanie ich wymagań w stosunku do SOA oraz przedstawienie czy SOA spełnia te wymogi i jak je realizuje.

Słowa kluczowe: SOA; systemy informatyczne; SOAP; REST; WebService; integracja systemów

*Autor do korespondencji.

Adres E-mail: pawel.wareciak@gmail.com

Analysis of the possibilities of using SOA to integrate IT systems

Paweł Waręciak*, Piotr Kopniak
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Abstract. The aim of this work is to show different integration methods of IT systems with SOA conception. Compare their requirements to SOA and present if SOA meets these requirements and how implement them.

Keywords: SOA; IT systems; SOAP; REST; WebService; system integration

*Corresponding author.

E-mail address: pawel.wareciak@gmail.com

1. Wstęp

Współczesne aplikacje generują duże ilości danych, które następnie trzeba przetworzyć. Niekiedy takie systemy stają się skomplikowane w utrzymaniu i niepodatne na nowe zmiany. Wraz z rozwojem takich aplikacji zachodzi potrzeba integracji pomiędzy nimi. Istnieją różne możliwości integracji aplikacji, między innymi Architektura Zorientowana na Serwisy - SOA (ang. Service Oriented Architecture).

Integracja systemów jest to proces, który pozwala na wykorzystanie danych jakie rejestruje i generuje system informatyczny, funkcji tego systemu oraz procesów biznesowych innym systemom informatycznym. Główne dziedziny integracyjne to [1]:

- integracja aplikacji,
- integracja danych,
- integracja procesów.

Integracje aplikacji można określić jako integracje oprogramowania w celu automatycznej wymiany informacji pomiędzy aplikacjami często stworzonymi przez różnych producentów i działającymi w jednym przedsiębiorstwie lub w przedsiębiorstwach partnerskich.

Drugim typem integracji jest integracja danych, która polega na eliminacji zbędnych powtórzeń z baz danych, czyli redundancji danych. Dzięki temu różne systemy informatyczne operują na takich samych danych, a aktualizacje danych propagowane są do wszystkich zainteresowanych aplikacji. Zapewnia to jednolitość danych we wszystkich integrowanych systemach.

Ostatnim typem integracji jest integracja procesów, polegająca na współdzieleniu procesów biznesowych pomiędzy kilkoma aplikacjami, w taki sposób, że elementy procesów mogą być wielokrotnie wykorzystywane przez różne aplikacje i możliwa jest budowa różnych procesów na ich podstawie. Procesy takie wykorzystują zwykle wiele systemów informatycznych i wykraczają poza granice jednego przedsiębiorstwa.

Rozwiązania stosowane przy integracji systemów możemy podzielić na kilka typów [2]:

- portale informacyjne,
- rozproszone procesy biznesowe,
- współdzielona logika biznesowa,
- architektura zorientowana na usługi (SOA),
- transfer i replikacja danych,
- integracja aplikacji (EAI),
- obiekty rozproszone - RPC i CORBA,
- systemy kolejkowe.

Zastosowanie integracyjne w **portalach informacyjnych** cechuje udostępnienie jednego interfejsu np. strony internetowej. Użytkownik może zobaczyć wszystkie interesujące go dane z wielu systemów informatycznych na jednej stronie internetowej (portalu). Informacje są agregowane z wielu systemów i pokazane np. w formie modułów - sekcji na stronie, np. portletów, które reprezentują fragmenty różnych systemów.

Rozproszone procesy biznesowe i współdzielona logika biznesowa opierają się na integracji procesów biznesowych wewnątrz jednego systemu. Zdarza się że procesy biznesowe pochodzą z innych systemów, wtedy mówimy że jest to integracja B2B.

Transfer lub replikacja danych występuje kiedy zachodzi potrzeba synchronizacji danych przy integracji z innym systemem. Zastosowanie takiego rozwiązania można spotkać np. w systemach CRM (Client Relationship

Management). Przy takim podejściu przechowujemy dane z kilku aplikacji w jednym miejscu lub synchronizujemy dane w przechowywane w wielu miejscach. Poza zaletami takiego podejścia, tzn. spójnymi danymi we wszystkich aplikacjach, mogą pojawić się problemy wpływające na działanie poszczególnych z nich. Może to generować problemy przy niekompatybilnych modelach danych w nowszych wersjach aplikacji. Do replikacji na bazie danych wykorzystuje się wbudowane rozwiązania. Transfer danych to wymiana danych pomiędzy systemami za pomocą plików lup połączeń sieciowych wykorzystujących Gniazda TCP.

EAI służy do integracji aplikacji oraz procesów wewnątrz organizacji. Głównym celem jest przekształcenie aplikacji w spójną strukturę.

Gniazda zapewniają dwukierunkowy kanał do komunikacji za pomocą jednego gniazda TCP.

RPC i CORBA są to mechanizmy zdalnego wywoływania procedur. Umożliwiają współdzielenie funkcjonalności pomiędzy systemami. Pierwszy z nich wymaga takiego samego języka programowania do stworzenia komunikujących się, rozproszonych w sieci obiektów. CORBA dzięki zastosowaniu komponentów pośrednich ORB nie ma takiego wymagania i jest rozwiązaniem bardziej uniwersalnym.

Kolejki służą do przesyłania komunikatów lub plików pomiędzy integrowanymi aplikacjami w sposób zapewniający dotarcie do odbiorcy w określonej kolejności nawet w przypadku awarii sieci i czasowego braku komunikacji. Dodają warstwę pośrednią dla innych komponentów. Przykładem takiego rozwiązania jest Java Message Services (JMS).

SOA jest architekturą do tworzenia złożonych systemów informatycznych, która wg specyfikacji posiada cechy wszystkich opisywanych metod integracji systemów. Niniejszy artykuł analizuje czy SOA rzeczywiście nadaje się do integracji systemów i w jaki sposób można to zrobić.

2. Możliwości integracyjne systemów informatycznych

2.1. JMS

Java Message Services (JMS) można zdefiniować jako standard przesyłania wiadomości. Jest to popularny sposób integrowania aplikacji, poprzez wykorzystanie luźnych powiązań oraz przesyłania komunikatów asynchronicznie [3]. Architektura JMS zawiera następujące elementy [4]:

- **dostawce JMS** - dostarcza narzędzia do administrowania oraz implementuje interfejsy
- **klienta JMS** - klienci operują na komunikatach, tworzą, wysyłają oraz je odbierają
- **narzędzia do zarządzania** - umożliwiają tworzenie oraz zarządzanie połączeniami

Wiadomości są przekazywane poprzez jedną z dwóch możliwości:

- **kolejkę (Queue)** – wiadomość kierowana jest do jednego adresata,
- **temat (Topic)** - wiadomości są subskrybowane przez wielu odbiorców.

Wykorzystanie kolejek zapewnia luźną integracje systemów informatycznych, co oznacza że komunikacja nie wymaga jednoczesnego połączenia klientów i nie musi być prowadzona w czasie rzeczywistym. Takie podejście jest głównie wykorzystywane do budowania warstw pośrednich w celu wykonywania usług dla innych komponentów.

JMS umożliwia integracje aplikacji oraz procesów. Poprzez zastosowanie asynchronicznego generowania zdarzeń można bezpośrednio przekazywać zdarzenia do punktu zbiorczego. Takie podejście pozwala na wyzwalanie kolejnych zdarzeń w procesie.

2.2. RPC i CORBA

Technologia zdalnego wywoływania procedur (RPC - ang. Remote Procedure Call) została opracowana i zaczęła być powszechnie stosowana w roku 1980 [5]. Podczas wywołania RPC parametry procedury są zestawione w żądaniu i przesłane na serwer. Zazwyczaj żądania są przesyłane za pomocą protokołu UDP albo TCP oraz zawierają informacje o nazwie procedury z listą jej parametrów [5]. Po dotarciu żądania do serwera jest ono konwertowane oraz procedura jest weryfikowana przez serwer. W odpowiedzi serwer aplikacji odsyła wartości zwrócone przez procedurę do klienta który oczekuje na żądanie. Komunikat RPC zawiera zarówno zbiór wartości zwróconych przez procedurę lub informacje o błędzie. Wywoływanie synchroniczne jest typowym trybem dla operacji na zdalnych procedurach, również w przypadku Common Object Request Broker Architecture (CORBA) [5]. CORBA jest rozszerzeniem RPC, która opisuje podstawową infrastrukturę do tworzenia zdalnych wywołań procedury poprzez obiekty. Systemy rozproszone mogą pracować na różnych systemach oraz wykorzystywać różne języki programowania.

RPC i CORBA są odpowiednim standardem dla rozproszonych obiektów tzw. fragmentów aplikacji działających na innych środowiskach, które współpracują ze sobą. Zaprojektowanie rozwiązania w taki sposób umożliwia integracje aplikacji.

2.3. Gniazda

Gniazda są protokołem sieciowym, który umożliwia zestawienie dwukierunkowego kanału komunikacji za pośrednictwem pojedynczego połączenia TCP [6]. Protokoły TCP oraz UDP które mogą zostać wykorzystane poprzez gniazda, cechuje zapewnienie połączenia zawsze pomiędzy dwoma procesami [7]. W przypadku WebSocket połączenie jest nawiązywane za pomocą protokołu HTTP. Aplikacja która inicjuje połączenie wysyła specjalne żądanie HTTP z adresem URL punktu końcowego z którym chce nawiązać połączenie. Jeśli serwer zaakceptuje połączenie, tworzona jest odpowiedz która zostaje przesłana do klienta. W takim wypadku połączenie TCP może zostać zestawione i będzie aktywne dopóki któraś ze stron nie zerwie połączenia albo nie zostanie przekroczony limit czasu z powodu bezczynności.

Wykorzystanie gniazd umożliwia integracje aplikacji, gdzie zachodzi potrzeba aktualizacji danych w czasie rzeczywistym np. w przypadku danych giełdowych.

2.4. Architektura SOA

Większość aplikacji jest stworzona w architekturze trójwarstwowej. W takim podejściu dużą wagę przywiązuje się do interfejsu użytkownika. Bardzo często spotykamy potrzebę rozszerzenia takiej aplikacji o możliwość wymiany

danych z innym systemem. Niestety nie zawsze taka możliwość została przewidziana podczas projektowania aplikacji. Dlatego przy tradycyjnym podejściu, kiedy zachodzi potrzeba integracji z innym systemem, rozwiązanie jest przygotowane na szybko bez przemyślanej koncepcji. Takie rozwiązania nierzadko były dostosowane do jednego systemu, co uniemożliwiało zastosowanie istniejącego rozwiązania do integracji z innymi systemami. Największym problemem takiego podejścia jest budowana nieelastycznej na zmiany aplikacji oraz duża pracochłonność i koszt nowych modyfikacji. Architektura zorientowana na usługi wprowadza koncepcję integracji różnych systemów przy zastosowaniu wydzielonych usług oraz podziału funkcjonalności aplikacji na usługi [8].

Architektura zorientowana na usługi określa koncepcję tworzenia usług. Głównymi cechami charakteryzującymi usługi są [9]:

- Kontrakt - interfejs usługi, który zawiera informacje o zasadach komunikacji, danych wejściowych, wyjściowych oraz metodach, które udostępnia dana usługa sieciowa.
- Luźne powiązania - usługi udostępniają pewien kontrakt danych. Ukrywając jak jest realizowana dana funkcjonalność.
- Abstrakcja - zawiera tylko podstawowe informacje na temat usługi, poza udostępnionym kontraktem. W całości ukrywa jak realizowana jest dana funkcjonalność.
- Ponowne wykorzystanie - usługi zawierają podzieloną logikę pomiędzy sobą, przy podziale logiki w łatwy sposób można wykorzystywać ponownie stworzoną funkcjonalność.
- Autonomiczność - każda usługa jest autonomiczna. Odpowiedzialność takiego serwisu ogranicza się do realizowanej funkcjonalności.
- Bezstanowość - minimalizuje użycie zasobów, nie przechowując informacji o kliencie. Brakuje informacji o wykonanych wcześniej operacjach.
- Wykrywalność - stworzone z myślą o wykorzystaniu w rozproszonych systemach. Usługi są uzupełniane metadanymi za pomocą, których mogą być skutecznie wykryte i zinterpretowane.
- Kompozycja - mogą składać się z kilku funkcjonalności realizowanych poprzez inne usługi, tzn. tworzyć złożone usługi, które rozdzielają zadanie pomiędzy nimi.

Poniżej przedstawione zostaną standardy, na których opiera się architektura SOA, tzn. usługa WebService, UDDI, SOAP i ESB. Przedstawiony zostanie także inny typ usługi - REST.

2.5. Usługą WebService

Termin WS (Web Services), czyli usługę sieciową możemy zdefiniować jako oprogramowanie stworzone do wymiany danych pomiędzy maszynami poprzez sieć, wykorzystując do tego zbiór technologii (WSDL, XML, SOAP, UDDI)[10]. Usługi sieciowe są jednym z rozwiązań stosowanych przy implementacji architektury zorientowanej na usługi. Bazując na standardach komunikacji poprzez protokół wymiany wiadomości kończąc na kontrakcie danych. Są reużywalnymi elementami budowy procesów biznesowych w systemach informatycznych.

2.6. UDDI

UDDI (Universal Description, Discovery and Integration)[11] - technologia pozwalająca na publikacje i wyszukiwanie informacji o usługach WebService. Jest to otwarty standard zarządzany przez organizacje OASIS [12]. Opiera się na mechanizmie publikacji - wyszukania - powiązania. Skorzystanie z usługi musi zostać poprzedzone opublikowaniem jej w rejestrze. Konsument wyszukuje usługi dostępne w rejestrze, po odnalezieniu konkretnej usługi następuje powiązanie jej z konsumentem, który otrzymuje możliwość korzystania z operacji danej usługi. Informacje wykorzystywane w UDDI są opisane za pomocą rejestru biznesowego, który można podzielić na trzy grupy:

- białe strony (white pages) - dane kontaktowe organizacji dostarczającej usługę,
- żółte strony (yellow pages) - klasyfikacja biznesu i usług,
- zielone strony (green pages) - informacje o udostępnionej usłudze np. z odnośnikiem do specyfikacji.

2.7. REST

Innym typem usługi sieciowej rzadko wykorzystywanym w rozwiązaniach SOA, ale opisanej dla uzupełnienia wiedzy o usługach sieciowych jest REST. Jest to koncepcja tworzenia usługi/protokołu, nieposiadająca opisu zestawu operacji danej usługi. Transmisja danych jest wykonywana w oparciu protokół HTTP. Usługa REST bazuje na następujących założeniach [13]:

- **Adresowalne zasoby** – wszystkie dostępne zasoby danej usługi powinny być adresowalne za pomocą unikalnego adresu URI
- **Ujednolicony interfejs** – usługa posiada prosty dobrze określony zestaw metod, za pomocą, których uzyskuje się dostęp do jej zasobów.
- **Wielopostaciowa reprezentacja** – cześć zasobów w zależności od potrzeb może przybierać różną postać tzn. jeden zasób dostępny pod tym samym adresem URL, może mieć format XML-owy lub JSON-owy lub też zupełnie w zależności od potrzeb i jego zastosowania.
- **Bezstanowość** – usługa nie przechowuje stanu pomiędzy kolejnymi jej wywołaniami, wszystkie żądania są zupełnie od siebie niezależne.

REST wykorzystując protokół HTTP udostępnia metody (POST, GET, PUT, DELETE).

- **GET** – stosuje się w celu pobrania danego zasobu.
- **PUT** – z jego pomocą tworzy się nowy zasób pod wskazanym adresem URI oraz aktualizuje istniejące.
- **POST** – może być stosowany zamiennie z PUT, ale przyjęło się, że z pomocą metody POST zapisuje się w usłudze dane mające charakter hierarchiczny.
- **DELETE** - służy do usunięcia zasobu z usługi.

2.8. SOAP

W celu wymiany wiadomości usługi sieciowe typu WebService wykorzystują protokół SOAP. Protokół ten umożliwia przesłanie danych w formacie tekstowym i binarnym. Opisuje sposób kodowania oraz wymiany wiadomości w formacie XML. Dokumenty SOAP najczęściej są przysyłane za pomocą protokołu HTTP/HTTPS, ale możliwe jest też przesłanie wiadomości za pomocą protokołu

SMTP, FTP, RMI. SOAP opakowuje wiadomości i składa się z kilku elementów [14]:

- **Envelope -** wszystkie wiadomości SOAP zawierają ten element razem z przestrzenią nazw http://schemas.xmlsoap.org/soap/envelope/.
- **Nagłówek** - opcjonalny parametr, który może zostać przesłany. Zazwyczaj wykorzystuje się go do przesłania dodatkowych informacji odnośnie uwierzytelnienia, podpisu cyfrowego itp. Jeśli nagłówek jest niezrozumiały dla usługi, zostanie on pominięty.
- **Treść** - jest to treść wiadomości w formacie XML
- **Załączniki** - przechowują dane binarne

2.9. ESB

Enterprise Service Bus (ESB) jest to baza strukturalna i komunikacyjna dla usług w architekturze SOA, zakłada tworzenie systemu opartego o usługi oraz sterowanego przy pomocy zdarzeń. ESB nie są konieczne, ale są często stosowane przy architekturze SOA. ESB ułatwia dynamiczne łączenie, zarządzanie oraz interakcje, udostępniając wspólną infrastrukturę. Wymiana danych odbywa się przez szynę, która ma za zadanie rozdzielać wiadomości w zależności od konfiguracji. Architektura ESB zakłada szynę jako centralny model danych w połączeniu pomiędzy serwisami. Szyna zawiera wszystkie serwisy, które są związane z systemem oraz wiadomości które są przekazywane z oraz do serwisów za jej pośrednictwem. Zaletą takiego rozwiązania jest mała ilość połączeń pomiędzy usługami. ESB zakłada jeden model danych, skutkuje to tym że teoretycznie każdy serwis integrujący się z szyną powinien posiadać ten sam model danych. Takie wymaganie uniemożliwiło by wręcz integrację z innymi systemami. Rozwiązaniem tego problemu są adaptery, które umieszcza się na styku pomiędzy serwisem, a szyną. Służą one to konwersji danych do poszczególnych modeli [15].

2.10. Ocena możliwości zastosowania SOA przy integracji systemów informatycznych

Zastosowanie SOA umożliwia wykorzystanie 3 typów integracji. Architektura zorientowana na usługi pozwala na integracje aplikacji (EAI) poprzez wykorzystanie wydzielonych usług do wymiany informacji z innymi aplikacjami. W SOA mamy WebService który przesyła asynchroniczne wiadomości, ale dzięki SOAP ma jeszcze kontrolę błędów i niepowodzeń i generalnie służy do integracji nie tylko danych ale też współdzielenia funkcji biznesowych. SOA umożliwia integracje procesów biznesowych, przy pomocy szyny ESB. W tym wypadku ESB występuje w roli pośrednika pomiędzy wykonaniem procesów biznesowych (BPEL) a usługą, zarządzając również przysyłanymi komunikatami.

Kolejną możliwości jest integracja danych, gdzie SOA ignoruje przetwarzanie danych masowych, operując na poziomie usług. Automatycznie zakłada się zatem, że zdalna procedura (usługa) zaprojektowana została w taki sposób, by obsłużyć wszystkie możliwe sytuacje biznesowe związane ze zdalnym dostępem do zasobów.

3. Podsumowanie

Zastosowanie architektury zorientowanej na usługi do integracji systemów informatycznych jest dobrym pomysłem, ponieważ wprowadza koncepcje wydzielenia funkcjonalności do niezależnych usług. Przy takim podejściu systemy stworzone w koncepcji SOA, mogą wymieniać dane oraz tworzyć procesy biznesowe złożone z kilku aplikacji.

Porównując podejście wykorzystywane w architekturze SOA, do innych możliwości integracyjnych, koncepcja architektury zorientowanej na usługi umożliwia integrowanie usług dla wielu klientów. Taką możliwość mają jeszcze kolejki. RPC oraz gniazda cechują się integracją typu punkt-punkt. Główną wadą RPC jest blokowanie klienta, który oczekuje na odpowiedz od serwera. Koncepcja SOA wymusza wytwarzanie oprogramowania w oparciu o luźno powiązane usługi, co przy innych typach integracji nie zawsze jest możliwe. Przy takim podejściu koszty późniejszych zmian są bardzo duże. Poprzez narzut złożoności testowanie SOA jest znacznie trudniejsze, niż w innych typach integracji. Stworzone aplikacje przedstawiają podział systemu na usługi, co zapewnia większą elastyczność w odróżnieniu do integracji punkt-punkt. Zastosowanie takiego podejścia umożliwia ponowne wykorzystanie usług przez inne systemy oraz współdzielenie danych. Kolejną cechą jest możliwość budowy procesów biznesowych oraz wykorzystanie danych z jednego systemu w kolejnych. SOA poprzez swoją koncepcje umożliwia integrowanie aplikacji, danych oraz procesów. Wprowadza to bardzo elastyczne podejście ponownego wykorzystania usług, stawiając SOA na lepszej pozycji podczas wyboru możliwości integracji systemów informatycznych.

Literatura

[1] Dr. Lui M., Gray M.,Chan A., Long J., Pro Spring Integration, Apress, Nowy Jork, 2011

[2] Kiermasz W., Integracja systemów informatycznych w architekturze zorientowanej na usług, Instytut Informatyki PW, 2009.

[3] Schaefer C., Ho C., Harrop R., Pro Spring, Fourth Edition, Apress, Nowy Jork, 2014

[4] JMS, http://docs.oracle.com/javaee/5/tutorial/doc/bncdx.html [16.04.2016]

[5] Bolton F., Pure CORBA, Sams, 2001

[6] Coward D., Java WebSocket Programming, Oracle Press, 2013

[7] Lombardi A., WebSocket, O'Reilly Media, Inc., Sebastopol, 2015

[8] Rotem-Gal-Oz A., SOA Patterns, Manning, Nowy Jork, 2012

[9] Plunkett T., Chou D., Balasubramanian R., Thomas P., Roy S., Tost A., Erl T., SOA with Java: Realizing Service-Orientation with Java Technologies, New Jersey Prentice Hall, 2014

[10] Dirksen J., SOA Governance in Action: REST and Web Service Architectures, Manning Publications, 2012

[11] UDDI, http://uddi.xml.org [16.04.2016]

[12] Organization for the Advancement of Structured, Information Standards. Strona domowa organizacji OASIS. http://www.oasis-open.org/home/index.php [16.04.2016]

[13] Shin S. SUN TECH DAYS, http://www.oracle.com [16.04.2016]

[14] W3C Organization. SOAP, https://www.w3.org/TR/soap12-part1 [16.04.2016]

[15] DiMaggio L., Conner K., Kumar M. B, Cunningham T.: JBoss ESB, Packt Publishing, Birmingham 2012