Biuletyn WAT
Vol. LXVI, Nr 1, 2017

Badanie wydajności magistral usług ESB typu Open Source

TOMASZ GÓRSKI, KACPER PIETRASIK

Wojskowa Akademia Techniczna, Wydział Cybernetyki,
Instytut Systemów Informatycznych,
00-908 Warszawa, ul. gen. S. Kaliskiego 2,
tomasz.gorski@wat.edu.pl, pietrasikk@gmail.com

Streszczenie. Rosnące zainteresowanie firm integracją oraz interoperacyjnością systemów informatycznych spowodowało wzrost znaczenia architektury usługowej (ang. *Service-Oriented Architecture*), która zapewnia narzędzia umożliwiające integrację aplikacji korporacyjnych (ang. *Enterprise Application Integration*). W tym sensie magistrala usług (ang. *Enterprise Service Bus*) zapewnia techniczne możliwości komunikacji między systemami informatycznymi.
Celem artykułu jest przedstawienie wyników badań wydajności wybranych magistral usług typu Open Source. Podstawą przeprowadzonego badania wydajności rozwiązania integracyjnego był przypadek biznesowy realizacji zamówienia w sklepie internetowym sieci sprzedającej sprzęt elektroniczny. W celu realizacji zamówienia wymagana była współpraca trzech systemów informatycznych. Rozwiązanie integracyjne zostało zaimplementowane przy wykorzystaniu każdej z wybranych do badania magistral usług: WSO2, Mule oraz Talend. Określono scenariusze testów i przeprowadzono testy wydajnościowe dla każdej z trzech wybranych magistral usług. Na podstawie uzyskanych wyników można stwierdzić, że każda z porównywanych platform ma swoje mocne i słabe strony. Koncentrując się na zaletach, należy wskazać, że WSO2 uzyskuje najkrótsze czasy transmisji przy komunikatach niewielkich rozmiarów, małej liczbie użytkowników i dodatkowo nie obciąża znacząco procesora. Magistrala Talend uzyskuje najkrótsze czasy transmisji przy przesyłaniu komunikatów o zarówno małych, jak i dużych rozmiarach, przy dużej liczbie użytkowników, jednak najmocniej obciąża procesor. Z badań wynika, że przy doborze magistrali usług do budowy rozwiązania integracyjnego warto wcześniej przeanalizować parametry przesyłanych komunikatów. Wyniki pokazują, że magistrala Talend może być dobrym wyborem przy budowie rozwiązania integracyjnego w środowisku biznesowym z dużą liczbą użytkowników oraz zróżnicowanymi komunikatami.
Słowa kluczowe: *Enterprise Service Bus*, ESB, wydajność, integracja, architektura usługowa, SOA
DOI: 10.5604/01.3001.0009.9487

1. Wprowadzenie

Naturalną sytuacją w dużych firmach jest istnienie wielu systemów informatycznych. Były one najczęściej budowane na przestrzeni lat funkcjonowania firm. Systemy te muszą współpracować, aby wspomagać realizację procesów biznesowych firmy. Architektura usługowa (SOA) dostarcza mechanizmy do współpracy (integracji) systemów informatycznych w przedsiębiorstwach (EAI). Wiodące technologie dla realizacji SOA wykorzystują standardy WS-*, takie jak SOAP, WSDL i BPEL. Magistrala usług (ESB) pojawiła się w celu umożliwienia integracji i współdziałania różnych aplikacji, usług lub innych zasobów w organizacjach [1]. Magistrala usług dostarcza warstwę mediacyjną umożliwiającą komunikację między współpracującymi aplikacjami. Tak duże jest znaczenie tej technologii, że zostały opracowane komercyjne magistrale usług, takie jak IBM Integration Bus [2], TIBCO Business Works [3] i Oracle Service Bus [4]. Ponadto, powstały implementacje magistrali usług typu Open Source, takie jak JBoss Fuse [5], Mule ESB [6], Petals ESB [7], WSO2 Enterprise Service Bus [8], OpenESB [9] i Talend ESB [10]. Dostawcy magistral usług typu Open Source poczynili znaczne postępy w stosunku do komercyjnych dostawców ESB i teraz stanowią realną opcję dla coraz większej liczby przedsiębiorstw [11]. Dodatkowo, zastosowanie magistrali typu Open Source w rozwiązaniach integracyjnych może prowadzić do oszczędności w postaci redukcji kosztów zakupu platformy wykonawczej ESB oraz kosztów utrzymania [12]. Uwzględniając powyższe, w artykule zdecydowano się poddać badaniu magistrale usług typu Open Source. Przedmiotem artykułu jest przedstawienie i analiza wyników badań wydajności wybranych magistral usług. Interesujące było uzyskanie odpowiedzi na pytanie: jak wybrane magistrale usług radzą sobie pod względem wydajności z transmisją zróżnicowanych komunikatów przy różnej liczbie użytkowników przesyłających te komunikaty? W badaniach interesujące było dla nas uzyskanie szerokiego spektrum miar wydajności: czasu transmisji komunikatu, całościowego czasu przetwarzania komunikatu, przepływności, wykorzystania mocy procesora. Ciekawe było także zmierzenie zakresu zmienności czasu transmisji komunikatu. Do badań wydajności zostały wybrane magistrale usług: Mule ESB [6], WSO2 Enterprise Service Bus [8] oraz Talend ESB [10]. Przy doborze magistral kierowaliśmy się ich zakresem stosowania w rozwiązaniach komercyjnych. Drugim argumentem była obecność magistral w testach wydajności przeprowadzanych przez firmy komercyjne, np.: WSO2 [13, 14], AdroitLogic [15]. Magistrala Mule ESB jest szeroko stosowana w rozwiązaniach integracyjnych już przynajmniej od 6 lat [16], a magistrala WSO2 Enterprise Service Bus jest coraz szerzej stosowana [17]. Dla wszystkich wybranych magistral możliwe jest graficzne projektowanie przepływów integracyjnych bez konieczności pisania bezpośrednio w kodzie XML. Magistrala Talend wybrana została także dlatego, ponieważ jest produktem firmy specjalizującej się w integracji danych. Autorzy artykułu spodziewali się innego sposobu podejścia do projektowania

tego produktu, więc też konsekwencji dla wydajności tego rozwiązania. Pozostałe magistrale Open Source pominięto ze względu na nakład prac niezbędny do implementacji rozwiązania integracyjnego na każdej z platform.

Artykuł został ułożony w przedstawiony dalej sposób. Sekcja 2 to przegląd prac powiązanych tematycznie z wydajnością magistral usług. W sekcji 3 przedstawiono przypadek biznesowy, który został zaimplementowany na każdej z wybranych magistral usług. Sekcja 4 opisuje konfigurację rozwiązania integracyjnego. W sekcji 5 przedstawiono wykorzystane w badaniu miary wydajności magistral usług. Sekcja 6 stanowi opis przypadków testowych, specyfikacji komunikatów testowych oraz środowiska testowego. Sekcja 7 zawiera wyniki badań wydajnościowych wraz z ich analizą. Podsumowanie i kierunki dalszych prac przedstawione zostały w sekcji 8.

2. Prace powiązane tematycznie

W celu wyszukania artykułów zajmujących się podobną tematyką dokonano trzech zapytań do bazy ScienceDirect. W każdym z trzech zapytań interesowały nas artykuły po roku 2005 w obszarze „Computer Science". W pierwszym zapytaniu użyto słowa kluczowego „Enterprise Service Bus" i uzyskano 21 artykułów. W drugim zapytaniu użyto słów kluczowych: „Performance" i „Service Oriented Architecture" i uzyskano 104 artykuły. W trzecim zapytaniu użyto słów kluczowych: „Performance" i „Evaluation". W tym zapytaniu listę artykułów ograniczono do tematyki: „model, service, QoS, message, performance, component". Ponadto, zdecydowano się także ograniczyć listę artykułów do czasopism z obszaru inżynierii systemów informatycznych: „Applied Soft Computing, Future Generation Computer Systems, Journal of Systems and Software, Simulation Modelling Practice and Theory, Performance Evaluation". Uzyskano listę 79 artykułów, a więc łącznie 204 artykuły. Następnie dokonano przeglądu tytułów i abstraktów, kwalifikując artykuły dotyczące architektury usługowej i jednocześnie wydajności. Na tej podstawie wyodrębniono 25 artykułów, których tematyka koncentrowała się wokół następujących obszarów: „peformance prediction", „Quality of Service", „performance evaluation/assessment" oraz „performance optimization/improvement".

Zagadnienie przewidywania wydajności obejmuje zastosowanie różnego rodzaju modeli do przewidywania wydajności rozwiązania integracyjnego na etapie jego projektowania. Becker proponuje stosowanie modelu Palladio Component Model (PCM) do specyfikacji architektury rozwiązania w sposób parametryczny [18]. Costa [19] używa metody Architecture Tradeoff Analysis Method (ATAM) do oceny rozwiązań w architekturze usługowej budowanej zgodnie ze stylem Representational State Transfer (REST). Do przewidywania czasów odpowiedzi usługi Närman [20] stosuje architekturę korporacyjną z wykorzystaniem modelu Hybrid Probabilistic Relational Model (HPRM). Na poziomie wykonawczym Brosig [21] proponuje zastosowanie

meta-modelu będącego podzbiorem Descartes Meta-Model (DMM) do predykcji wydajności on-line. Casale [22] proponuje algorytm Method of Moments (MoM) dający dokładne rozwiązania dla zamkniętych modeli kolejkowych z setkami lub tysiącami użytkowników korzystających z usług w architekturze usługowej. Sharma [23] proponuje model hierarchiczny dla predykcji m.in. wydajności oprogramowania, gdzie stosuje dyskretny model Markowa. To podejście wspomaga także identyfikację wąskich gardeł. Natomiast Teixeira [24] modeluje system w architekturze usługowej za pomocą Petri Nets i na bazie modelu estymuje poziomy Quality-of-Service (QoS). Wykorzystuje to na etapie określania Service Level Agreement (SLA). W obszarze QoS istotnymi zagadnieniami są minimalizacja kosztu wyboru usługi (Fudzee [25]) oraz kosztu kompozycji usług (Ko [26]). Ponadto, Yang [27] opisuje estymację QoS dla usług złożonych (ang. *composite service*). Zagadnienie negocjacji QoS między dostawcą usługi a jej klientem opisuje Menascé [28]. Natomiast Chang [29] porusza coraz istotniejszy temat sieci Mobile Social Network in Proximity (MSNP) i proponuje adaptacyjny osadzony na urządzeniu mobilnym framework wspierający proaktywne odkrywanie usług (ang. *service discovery*). Rozwiązanie to bazuje na magistrali usług.

Kluczowy dla nas był obszar oceny wydajności rozwiązań w architekturze usługowej. W tym obszarze Koziolek [30] przedstawia przegląd ponad dwudziestu metod oceny wydajności rozwiązań wykorzystujących komponenty. Hachicha [31] proponuje podejście do analizy i oceny współpracujących procesów biznesowych w architekturze usługowej w celu utrzymania ich wydajności. Mi [32] pokazuje wpływ wybuchowości (ang. *burstiness*) na wydajność systemów wielowarstwowych. Pokazuje on, że występowanie autokorelacji propaguje się między warstwami systemu i ma ona wpływ na wydłużanie się czasu odpowiedzi. Martinez [17] dokonuje porównania wydajności magistral usług: Fuse, Mule, Petals, WSO2 oraz JBoss, w kontekście budowy zintegrowanych środowisk biznesowych (ang. *Integrated Business Environments*). Zastosowanie symulacji do oceny wydajności rozwiązań w architekturze usługowej poruszają Mentis [33] oraz Górski [34]. Negash [35] proponuje rozproszoną magistralę usług (ang. *lightweight service bus* — LISA) dla Internet of Things (IoT), który charakteryzuje się heterogenicznością w kontekście protokołów sieciowych i platform. Sachs [36] proponuje metodologię oceny wydajności Message-oriented middleware (MOM), ze szczególnym uwzględnieniem komunikacji Java Messaging Service (JMS).

W obszarze poprawy wydajności Bezemer [37] przedstawia podejście wspomagające identyfikację potencjalnych ulepszeń wydajności (ang. *performance improvement opportunities* (PIOs) w identyfikacji wąskich gardeł (ang. *bottleneck detection*). Propozycję zastosowania architektury usługowej w aplikacjach naukowych do dynamicznego grupowania usług dla optymalizacji czasu wykonania aplikacji przedstawia Glatard [38]. Zagadnienie budowy środowiska współpracy dla niezależnych aplikacji z zastosowaniem magistrali usług i jej wydajność oraz

łatwość realizacji zadań poruszają Iqbal [39] oraz Górski [40]. Li [41] proponuje efektywny model transmisji dla współpracy heterogenicznych danych przestrzennych z zastosowaniem metod kodowania danych Geography Markup Language (GML) i GeoJSON oraz technik kompresji/dekompresji LZMA i DEFLATE. Potena [42] przedstawia model optymalizacji w celu minimalizacji kosztu adaptacji zmiany w architekturze usługowej przy uwzględnieniu wymagań pozafunkcjonalnych, w szczególności wydajności. Natomiast Wu [43] pokazuje dynamiczny schemat alokacji zasobów do pojedynczych usług podnoszący wydajność usług złożonych (ang. *composite services*). Dokonaliśmy także zapytania tylko ze słowem kluczowym „Performance" i zwróciło ono aż 54 817 artykułów. Wynika z tego, że wydajność jest cały czas jedną z kluczowych cech jakościowych rozwiązań w informatyce.

3. Opis przypadku biznesowego

Do badania wydajności magistral usług wybrano proces realizacji zamówienia złożonego w sklepie internetowym ze sprzętem elektronicznym. Założono, że firma prowadząca ten sklep internetowy ma sieć hurtowni elektronicznych posiadających wspólne systemy do zarządzania finansami, magazynami i dystrybucji asortymentu.

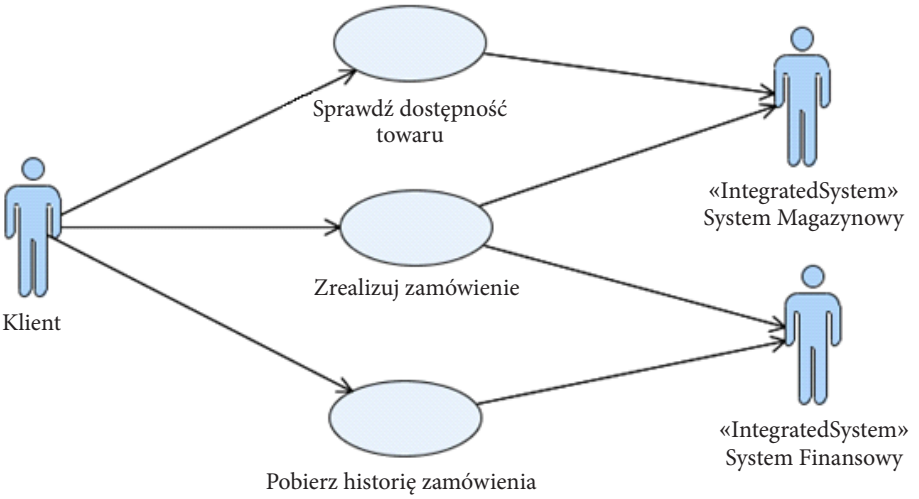
W celu realizacji zamówienia muszą współpracować następujące systemy informatyczne:

- eHurtownia — umożliwia złożenie zamówienia, sprawdzenie dostępności produktów oraz stanu realizacji zamówienia,
- System Magazynowy — umożliwia zarządzanie stanami magazynowymi oraz dystrybucją produktów,
- System Finansowy — umożliwia zarządzanie finansami firmy oraz udostępnia historię zamówień klientów.

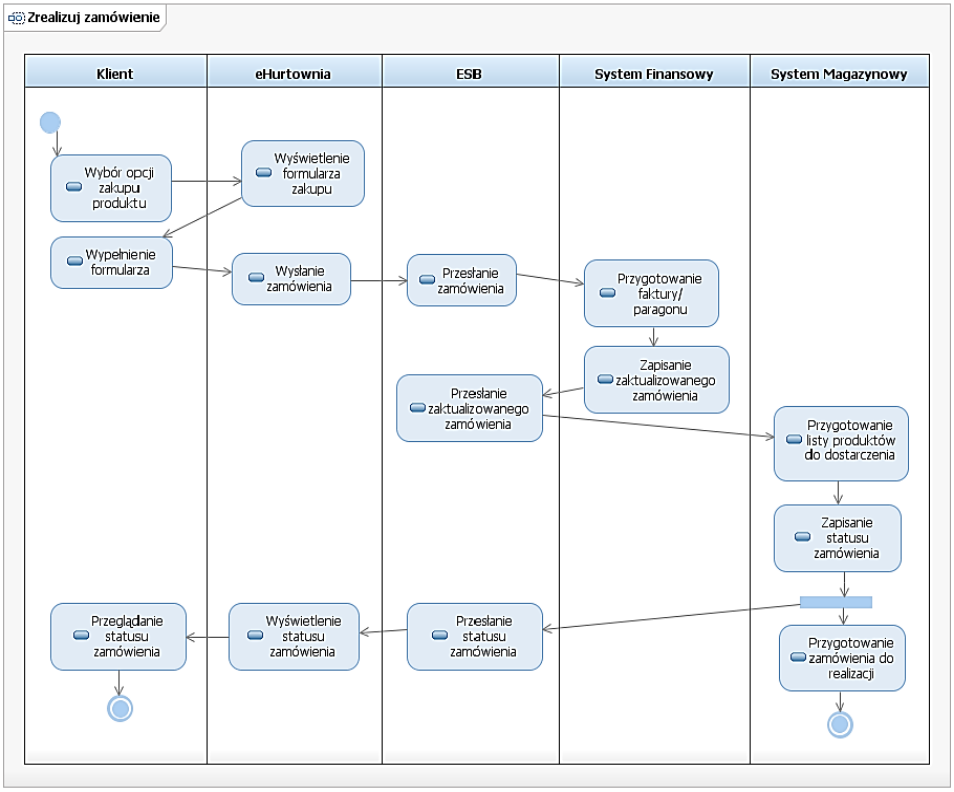
Rysunek 1 przedstawia diagram przypadków użycia systemu eHurtownia z wyodrębnionymi integrowanymi systemami informatycznymi. Na diagramie tym wykorzystano stereotyp «IntegratedSystem» z profilu UML Profile for Integration Platform [44].

Rysunek 2 przedstawia diagram aktywności języka UML dla przypadku użycia *Zrealizuj zamówienie* systemu eHurtownia. W opisie przebiegu zdarzeń oprócz klienta i systemu eHurtownia występują także systemy integrowane: Finansowy oraz Magazynowy.

Klient przegląda produkty i umieszcza wybrane w koszyku. W celu dokończenia zamówienia i dokonania płatności klient wybiera opcję „Zrealizuj zamówienie". Klient wypełnia pola formularza swoimi danymi i wybiera płatność przy odbiorze. Po wypełnieniu wszystkich potrzebnych pól formularza zamówienie jest wysyłane do Systemu Finansowego, gdzie przygotowywana jest faktura lub paragon, a następnie zapisywane jest w bazie danych Systemu Finansowego. Po tej operacji zamówienie



Rys. 1. Diagram przypadków użycia systemu eHurtownia



Rys. 2. Diagram aktywności dla przypadku użycia „Zrealizuj zamówienie" systemu eHurtownia

wzbogacone o fakturę/paragon przesyłane jest do Systemu Magazynowego. System Magazynowy na podstawie danych zamówienia przygotowuje listę produktów, w skład której wchodzą numer produktu, jego nazwa oraz umiejscowienie w magazynie. Tak przygotowane zamówienie jest oznaczane jako gotowe do realizacji przez Magazyniera. Po przygotowaniu zamówienia zapisywany jest jego status w bazie danych Systemu Magazynowego. Status zamówienia przesyłany jest następnie do systemu eHurtownia, gdzie jest wyświetlony Klientowi. Podobnie określono przebiegi zdarzeń dla przypadków użycia „Pobierz historię zamówienia" oraz „Sprawdź dostępność towaru" systemu eHurtownia oraz narysowano dla nich diagramy aktywności.

4. Konfiguracja rozwiązania integracyjnego

Wszystkie magistrale zostały pobrane ze stron producentów wraz ze środowiskami wykonawczymi i poddane testom na domyślnych konfiguracjach. Dla magistrali WSO2 Enterprise Service Bus [8] domyślna konfiguracja została poddana modyfikacji ze względu na problemy z ukończeniem przygotowanych przypadków testowych. Bez zmiany domyślnej konfiguracji magistrala WSO2 Enterprise Service Bus nie przeszłaby pomyślnie testów wydajnościowych. Domyślnie konfiguracja protokołu HTTP w magistrali WSO2 wykorzystywała transport o nazwie *PassThrough* [45]. Jednak podczas przeprowadzania testów napotkano na problemy wydajnościowe, które uniemożliwiały dokończenie testów. Aby mieć możliwość wykonania wszystkich przypadków testowych, zdecydowano się na zmianę transportu na HTTP-NIO (ang. *Non-blocking Input/Output*). Z tych samych powodów domyślne ustawienie dwóch wątków zmieniono na 50 wątków. Podobnie, domyślne ustawienia -Xms256m oraz -Xmx1024m okazały się zbyt niskie przy niektórych przypadkach testowych i zostały zwiększone do -Xms3000m oraz -Xmx5600m. Ustawienia wartości domyślnych dla magistral Mule oraz Talend pozostawiono niezmienione. W przeprowadzonych badaniach nie badano wpływu wartości tych parametrów na wydajność magistral. Zapewniono tylko jednakowe sprzętowe środowisko wykonawcze wystarczające do wykonania wszystkich przypadków testowych.

Tabela 1 przedstawia przyjęte do badań wydajnościowych wartości parametrów konfiguracyjnych magistral.

Realizacja każdego z przypadków użycia pokazanych na rysunku 1 wymaga komunikacji między systemami: eHurtownia, System Magazynowy oraz System Finansowy. W komunikacji tej uczestniczy magistrala usług. Systemy informatyczne najczęściej posiadają różne formaty danych. Ponadto przesyłane między systemami informatycznymi dane są najczęściej w formatach XML oraz JSON. Dlatego pojawia się potrzeba konfiguracji przepływów integracyjnych między systemami informatycznymi, które wymieniają między sobą dane. W przepływach integracyjnych zastosowano wzorce mediacyjne.

Tabela 1

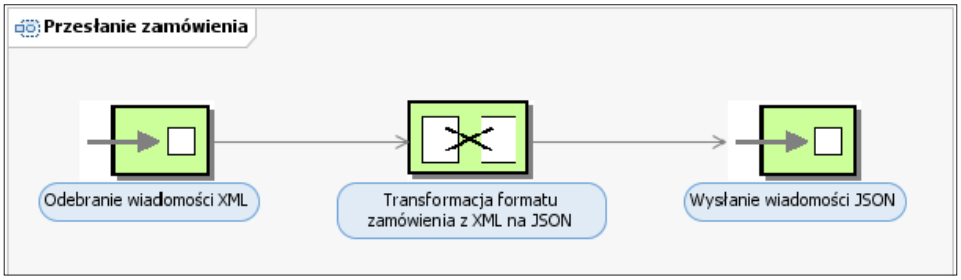
Parametry konfiguracyjne magistral

Magistrala	Połączenie HTTP	Liczba wątków	Parametry JVM (1.7)
WSO2	HTTP-NIO	50	-Xms3000m -Xmx5600m
MULE	HTTP	16	-Xms1024m -Xmx1024m
TALEND	HTTP	20	-Xms128m -Xmx512m

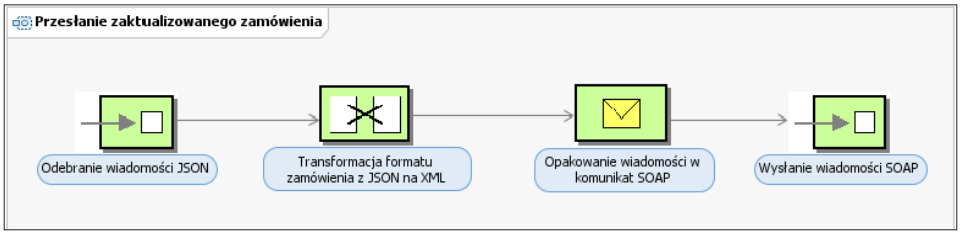
W realizacji przypadku użycia „Zrealizuj zamówienie" wykorzystano wzorce mediacyjne opakowania wiadomości (ang. *Envelope Wrapper*) oraz transformacji (ang. *Translator*). Dodatkowo przetestowano udział trzech punktów końcowych i związaną z tym kolejność przetwarzania komunikatów. Komunikat wejściowy zostaje przekształcony z formatu XML na format JSON i wysłany do Systemu Finansowego. Odpowiedź z tego systemu zostaje przekształcona z formatu JSON na XML, następnie opakowana w kopertę SOAP i wysłana do Systemu Magazynowego. Odpowiedź z tego systemu zostaje przekształcona z formatu SOAP na format XML i zwrócona do systemu eHurtownia.

Na diagramach przepływów integracyjnych (rys. 3-5) zastosowano stereotypy i ikony do nich przypisane określone w profilu UML Profile for Integration Flows [44].

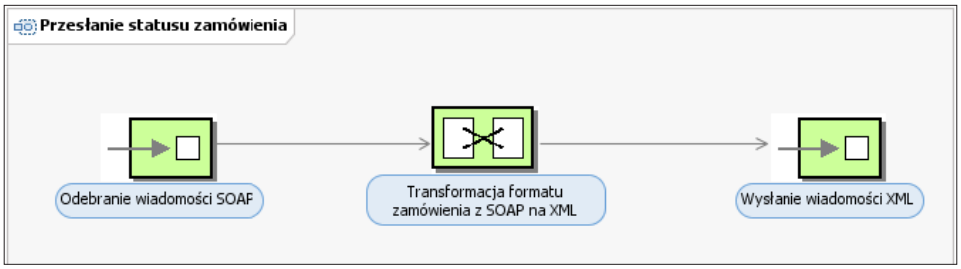
Rysunek 3 przedstawia diagram przepływu integracyjnego dla przypadku użycia „Zrealizuj zamówienie" pomiędzy systemami eHurtownia oraz Finansowym (działanie Przesłanie zamówienia). Rysunek 4 przedstawia diagram przepływu integracyjnego dla przypadku użycia „Zrealizuj zamówienie" pomiędzy Systemami Finansowym oraz Magazynowym (działanie „Przesłanie zaktualizowanego zamówienia"). Rysunek 5 przedstawia diagram przepływu integracyjnego dla przypadku użycia „Zrealizuj zamówienie" pomiędzy Systemami Magazynowym oraz eHurtownia (działanie „Przesłanie statusu zamówienia").



Rys. 3. Diagram przepływu integracyjnego „Przesłanie zamówienia"



Rys. 4. Diagram przepływu integracyjnego „Przesłanie zaktualizowanego zamówienia"

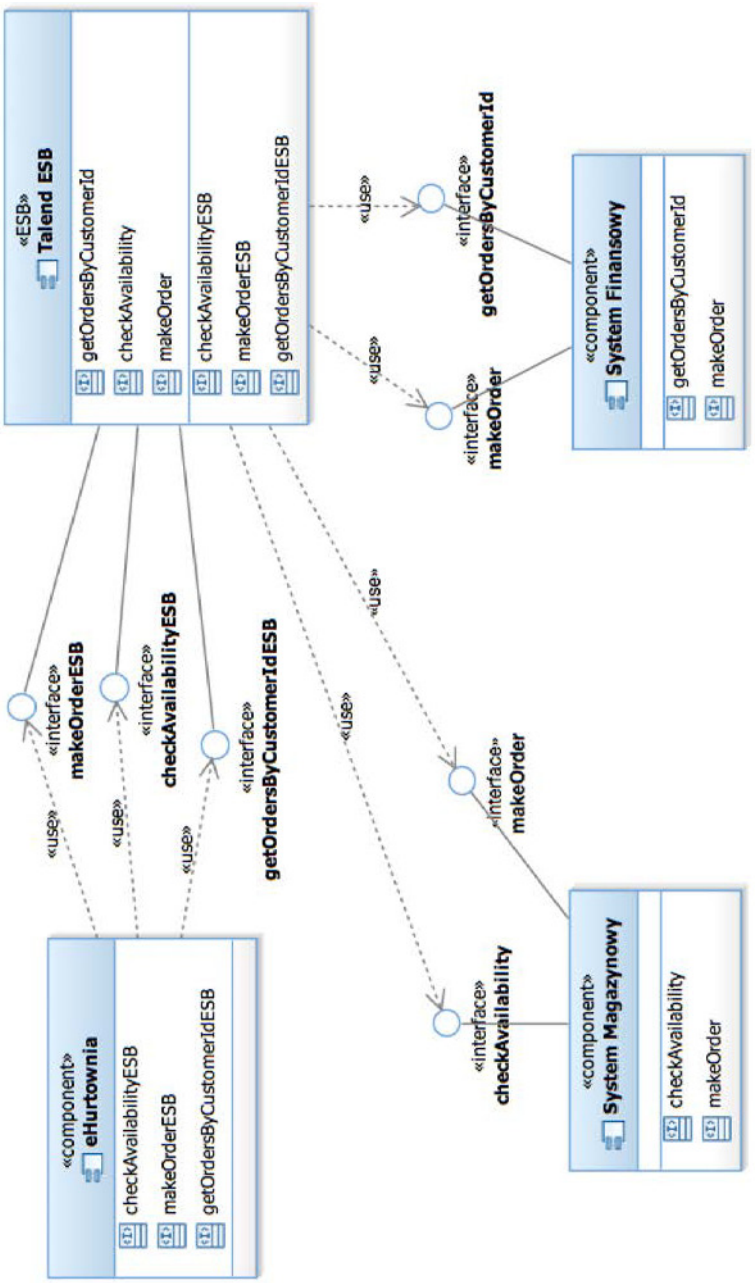


Rys. 5. Diagram przepływu integracyjnego „Przesłanie statusu zamówienia"

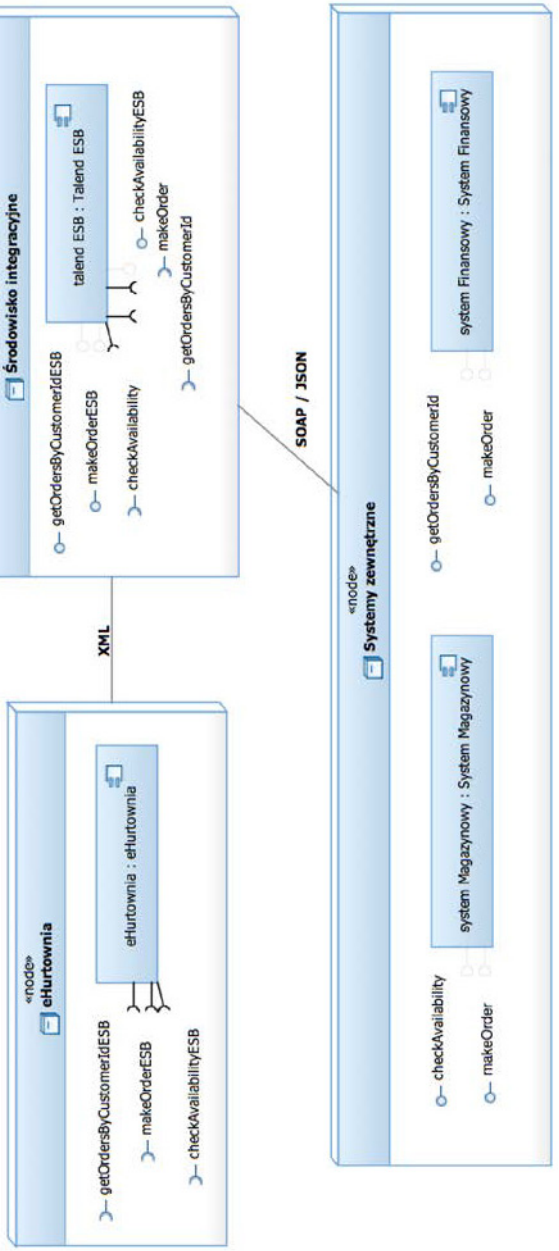
Podobnie zdefiniowano przepływy integracyjne dla przypadków użycia „Pobierz historię zamówienia" oraz „Sprawdź dostępność towaru" systemu eHurtownia.

W realizacji przypadku użycia „Pobierz historię zamówienia" zaimplementowano dwie usługi. Jedna z nich umożliwia pobranie pojedynczego zamówienia na podstawie identyfikatorów zamówienia oraz klienta (getOrderByOrderIdAndCustomerId) oraz wszystkich zamówień na podstawie identyfikatora klienta (getOrdersByCustomerId). W realizacji tego przypadku użycia wykorzystano wzorce mediacyjne trasowania (ang. *ContentBasedRouter*) oraz transformacji (ang. *Translator*). Komunikat wejściowy zostaje poddany mechanizmowi trasowania i w zależności od jego zawartości przesłany dalej. Następnie komunikat ten zostaje poddany transformacji z formatu XML na format JSON i wysłany do Systemu Finansowego. Odpowiedź z Systemu Finansowego zostaje poddana transformacji z formatu JSON na format XML i zwracana jest do systemu eHurtownia.

W realizacji przypadku użycia „Sprawdź dost*ępność* towaru" wykorzystano wzorce mediacyjne ograniczenia zawartości wiadomości (ang. *ContentFilter*), opakowania wiadomości (ang. *EnvelopeWrapper*) oraz transformacji (ang. *Translator*). Komunikat wejściowy zostaje poddany mechanizmowi ograniczenia zawartości wiadomości w celu wysłania do punktu docelowego tylko wymaganych danych. Następnie komunikat zostaje opakowany w kopertę SOAP i wysłany do Systemu Magazynowego. Odpowiedź z Systemu Magazynowego zostaje poddana transformacji z formatu SOAP na format XML i zwrócona do systemu eHurtownia.



Rys. 6. Diagram komponentów rozwiązania integracyjnego z magistralą Talend



Rys. 7. Diagram wdrożeniowy przedstawiający środowisko wykonawcze z magistralą Talend

Rysunek 6 przedstawia diagram komponentów ilustrujący komponenty rozwiązania integracyjnego wraz z usługami realizowanymi przez poszczególne komponenty. Na diagramie tym wykorzystano stereotyp «ESB» z profilu UML Profile for Integration Platform [44].

Na diagramie tym pokazano także użycie usług przez komponenty rozwiązania. Dla przejrzystości rysunku pominięto komponenty magistral usług Mule oraz WSO2. Komponenty dla magistral usług Mule oraz WSO2 realizują i korzystają z takich samych usług jak przedstawiony na rysunku 6 komponent magistrali Talend.

Natomiast rysunek 7 przedstawia diagram wdrożenia ilustrujący sposób rozmieszczenia poszczególnych komponentów środowiska wykonawczego z magistralą Talend. Dla przejrzystości rysunku pominięto komponenty magistral usług Mule oraz WSO2.

5. Miary wydajności magistral usług

W badaniach wydajności rozwiązania integracyjnego zbudowanego przy wykorzystaniu magistrali usług przyjęto następujące miary wydajności:

- T_C — całkowity czas przetwarzania komunikatu — czas od chwili wysłania komunikatu z systemu źródłowego do chwili otrzymania odpowiedzi w systemie źródłowym:
  - T_{CMIN} — minimalny całkowity czas przetwarzania komunikatu,
  - T_{CMAX} — maksymalny całkowity czas przetwarzania komunikatu,
  - T_{CSTD} — odchylenie standardowe całkowitego czasu przetwarzania komunikatu;
- T_{ESB} — czas przetwarzania komunikatu przez magistralę — suma czasów przesyłania komunikatu przez magistralę usług w trakcie przetwarzania komunikatu:
  - T_{ESBMIN} — minimalny czas przetwarzania komunikatu przez magistralę,
  - T_{ESBMAX} — maksymalny czas przetwarzania komunikatu przez magistralę,
  - T_{ESBSTD} — odchylenie standardowe czasu przetwarzania komunikatu przez magistralę;
- T_{SF} — czas przetwarzania komunikatu przez System Finansowy;
- T_{SM} — czas przetwarzania komunikatu przez System Magazynowy;
- T_R — pozostały czas przetwarzania komunikatu;
- P — przepływność — liczba komunikatów, jaką rozwiązanie integracyjne przetwarza w jednostce czasu;
- CPU — zajętość procesora — procentowe wykorzystanie mocy procesora.

6. Testy rozwiązania integracyjnego

W testach wydajnościowych wykorzystano następujące przypadki użycia: „Zrealizuj zamówienie", „Sprawdź dostępność towaru", „Pobierz historię zamówienia" (rys. 1). Dla każdego z przypadków użycia przygotowano przynajmniej jeden przypadek testowy. Łącznie przygotowano siedem przypadków testowych. Tabela 2 przedstawia przypadki testowe wraz z nazwami zaimplementowanych usług oraz rozmiarami komunikatów wejściowych i wyjściowych.

Tabela 2

Przypadki testowe z określonymi rozmiarami komunikatów wejściowych i wyjściowych

Nr przypadku testowego	Nazwa serwisu	Rozmiar komunikatu wejściowego [B]	Rozmiar komunikatu wyjściowego [B]	Komunikat zwrotny zawiera zdjęcia
1	getOrderByOrderIdAndCustomerId	156	448	Nie
2	getOrderByOrderIdAndCustomerId	156	16 900	Tak
3	getOrdersByCustomerId	156	4 140	Nie
4	getOrdersByCustomerId	156	221 000	Tak
5	checkAvailability	440	400	Nie
6	makeOrder	27 000	29	Tak
7	makeOrder	99 000	29	Tak

W celu uzyskania wartości miar wydajności rozwiązania integracyjnego przeprowadzono testy wydajnościowe. W ramach uruchomienia pojedynczego przypadku testowego przesłano każdorazowo 20 000 komunikatów i wyznaczono wartości średnie dla rozpatrywanych miar wydajności. Każdy z przypadków testowych z tabeli 1 powtórzono dziesięć razy. Wartość miary wydajności stanowi średnia z dziesięciu powtórzeń przypadku testowego. Ponadto, procedura testowania dla każdego przypadku testowego powtórzona została dla 5, 20 oraz 50 użytkowników. Należy zaznaczyć, że liczba komunikatów nie zwiększała się. Natomiast rozdzielano je na wiele źródeł generujących komunikaty do przesłania przez magistralę usług. Obciążenie procesora mierzono na komputerze „*Środowisko integracyjne*" (rys. 7), wykorzystując do tego celu monitor wydajności wchodzący w skład systemu Windows. Wartości estymatorów miar wydajności zbierano przy wykorzystaniu programu Apache JMeter, który został skonfigurowany do wykonania przedstawionych w tabeli (tab. 1) przypadków testowych.

7. Analiza wyników badań wydajności

Wyniki uzyskane z przeprowadzonych testów wydajnościowych przedstawiono w tabelach 3-6. Po analizie wyników testów wydajnościowych sformułowano dalej przedstawione wnioski. Tabela 3 zawiera zestawienie uzyskanych czasów przetwarzania komunikatu przez magistralę dla wszystkich przypadków testowych.

Tabela 3

Zestawienie czasów przetwarzania komunikatu przez magistralę T_{ESB}

Czas przetwarzania komunikatu przez magistralę T_{ESB} [ms]									
Nr przypadku testowego	WSO2			MULE			TALEND		
	5	20	50	5	20	50	5	20	50
1	9,41	23,39	32,31	16	67	168	14,22	30,88	78,67
2	9,96	26,04	38,43	15	68	173	17,48	48,02	65,9
3	9,5	25,69	35,02	15	62	167	17,1	29,24	32,08
4	73,36	291,46	719,22	11	13	17	28,93	32,99	32,3
5	10,3	25,78	33,26	14,71	52	150	17,44	55,76	146,41
6	14,29	53,11	108,38	23	97	246	22,28	24,5	24,64
7	10,82	32,51	52,31	20	62	177	31,14	33,89	34,1

Dla wszystkich magistral i dla wszystkich przypadków testowych wraz ze wzrostem liczby użytkowników czas przetwarzania komunikatu przez magistralę wzrasta.

Natomiast warto zauważyć, że magistrala Talend uzyskuje najbardziej zbliżone do siebie czasy przetwarzania komunikatu T_{ESB} niezależnie od przypadku testowego i liczby użytkowników. Ponadto, tylko dla tej magistrali w 18 na 21 eksperymentów (86%) czasy przetwarzania komunikatu mieszczą się w granicach 60 ms. W pozostałych 2 eksperymentach na 3 średnie czasy przetwarzania mieszczą się w granicach 80 ms. Z dalszej analizy wynika, że tylko w 1 na 21 eksperymentów (5%) uzyskany czas znacząco różni się od pozostałych (przypadek testowy nr 5 dla 50 użytkowników). Dla porównania dla magistrali Mule tylko w 10 eksperymentach na 21 (48%) czasy przetwarzania komunikatu mieszczą się w granicach 60 ms. Natomiast w aż 6 eksperymentach na 21 (29%) czas T_{ESB} znacząco przekracza 100 ms (przypadki testowe nr 1, 2, 3, 5, 6, 7 dla 50 użytkowników). Dla magistrali WSO2 w 17 eksperymentach na 21 (81%) czasy przetwarzania komunikatu mieszczą się w granicach 60 ms. W 3 eksperymentach na 21 (14%) czasy przetwarzania komunikatu znacząco przekraczają 100 ms (przypadki testowe: nr 6 dla 50 użytkowników, nr 3 dla 20 użytkowników oraz nr 4 dla 50 użytkowników), przy czym dla przypadku nr 4 dla 50 użytkowników WSO2 uzyskała najgorszy wynik ze wszystkich magistral wynoszący 719,22 ms. Z jednej strony WSO2, w większości przypadków testowych, uzyskuje

najkrótsze wartości T_{ESB} dla liczby 5 użytkowników, lecz potrafi uzyskać najdłuższą wartość T_{ESB} (przypadek testowy nr 4 dla 50 użytkowników).

Warto także uwypuklić wyniki uzyskane przez magistrale dla liczby 50 użytkowników. Magistrala Talend w żadnym z przypadków testowych nie uzyskała najgorszego wyniku w sensie najdłuższego czasu T_{ESB}, w żadnym nie przekroczyła 150 ms. Ponadto, dla trzech przypadków testowych (numer 3, 6 oraz 7) uzyskała najkrótszą wartość tego czasu. Magistrala Mule aż dla 6 przypadków testowych uzyskała najdłuższy czas T_{ESB} (przypadki testowe o numerach: 1, 2, 3, 5, 6 oraz 7) wynoszący 150 ms albo więcej. Tylko raz magistrala ta uzyskała najkrótszy czas T_{ESB}. Magistrala WSO2 dla 3 przypadków testowych uzyskała najkrótszy czas T_{ESB} (przypadki testowe o numerach: 1, 2 oraz 5), ale też raz uzyskała najdłuższy czas T_{ESB} i to w dodatku wynoszący 719,22 ms. Można zauważyć, że najkrótsze czasy przetwarzania komunikatu uzyskała magistrala WSO2 w 14 eksperymentach, magistrala Talend w 4, a magistrala Mule w 3. Natomiast najdłuższe czasy przetwarzania uzyskała magistrala Mule w 13 eksperymentach, magistrala Talend w 5, a magistrala WSO2 w 3 eksperymentach.

Tabela 4

Zestawienie odchylenia standardowego T_{ESBSTD} testowanych magistral

Odchylenie standardowe T_{ESBSTD} [ms]									
Nr przypadku testowego	WSO2			MULE			TALEND		
	5	20	50	5	20	50	5	20	50
1	19,18	35,19	41,93	17	38	78	12,8	31,49	60,81
2	19,81	36,71	45,57	18	37	77	15,69	36,75	61,39
3	18,12	36,25	42,67	16	37	76	15,27	29,01	35
4	14,47	34,22	92,38	8	15	26	12,33	22,18	15,84
5	20,15	37,77	45,57	18,99	42	87	14,67	36,21	82,36
6	40	16,82	37,92	23	52	107	16,99	24,3	21,41
7	28,75	90,65	166,79	12	43	93	11,96	13,5	13,79

Ponadto w tabeli 4 przedstawiono uzyskane wartości odchylenia standardowego T_{ESBSTD} dla porównywanych magistral. Najmniejsze wartości odchylenia standardowego T_{ESBSTD} uzyskała magistrala Talend w aż 14 eksperymentach na 21 (67%). Z drugiej strony, największe wartości odchylenia standardowego T_{ESBSTD} uzyskała magistrala WSO2 w aż 11 eksperymentach na 21 (52%). Magistrala Mule uzyskała wyniki pośrednie. Widać z tego, że najbardziej zbliżone do siebie wyniki poszczególnych testów uzyskała magistrala Talend. Dodatkowo warto zwrócić uwagę, że wartość odchylenia standardowego T_{ESBSTD} dla magistrali Talend nigdy nie osiągnęła najgorszego wyniku.

Tabela 5 zawiera zestawienie procentowego udziału czasu przetwarzania komunikatu przez magistralę w całkowitym czasie przetwarzania komunikatu dla wszystkich przypadków testowych.

Tabela 5

Zestawienie wartości stosunku T_{ESB}/T_C

T_{ESB}/T_C [%]									
Nr przypadku testowego	WSO2			MULE			TALEND		
	5	20	50	5	20	50	5	20	50
1	30	36	26	32	44	47	29	21	25
2	27	34	26	28	42	45	39	35	20
3	25	32	22	28	40	45	29	15	7
4	47	70	74	6	3	2	15	4	2
5	34	40	29	32	42	43	37	42	43
6	20	29	24	27	40	42	15	4	2
7	7	11	7	12	22	27	7	2	1

Dla magistrali WSO2 średnia wartość T_{ESB}/T_C ze wszystkich eksperymentów wynosi 30,95%. Zaobserwowano, że wartość T_{ESB}/T_C wzrasta wraz ze wzrostem liczby użytkowników pomiędzy liczbą 5 a 20 użytkowników (dla wszystkich przypadków testowych). Natomiast wartość T_{ESB}/T_C maleje wraz ze wzrostem liczby użytkowników pomiędzy liczbą 20 a 50 użytkowników (dla 6 przypadków testowych). Dla 5 przypadków testowych wartość T_{ESB}/T_C dla 50 użytkowników była niższa niż dla 5 użytkowników.

Dla magistrali Talend średnia wartość T_{ESB}/T_C ze wszystkich eksperymentów wynosi 18,76%. Wartość T_{ESB}/T_C dla Talend nie przekroczyła 43%, więc jest zdecydowanie niższa niż dla WSO2. Ponadto średnia wartość T_{ESB}/T_C dla Talend stanowi tylko 60,6% tej wartości dla WSO2. W odróżnieniu od WSO2 wartość T_{ESB}/T_C maleje wraz ze wzrostem liczby użytkowników (dla 5 przypadków testowych). Należy podkreślić, że dla przypadków testowych 6 i 7 dotyczących przypadku użycia „Zrealizuj zamówienie" magistrala Talend uzyskuje najlepsze wyniki zarówno w czasie bezwzględnym T_{ESB}, jak i wartości T_{ESB}/T_C. Jest to o tyle istotne, że w tych przypadkach testowych przesyłane są duże komunikaty ze zdjęciami oraz wymagane są aż trzy przesłania komunikatu na magistrali w celu wykonania przypadku testowego.

Dla magistrali Mule średnia wartość T_{ESB}/T_C ze wszystkich eksperymentów wynosi 30,91%. Wartość T_{ESB}/T_C dla Mule nie przekroczyła 47%, więc podobnie jak dla Talend. Jednak średnia wartość T_{ESB}/T_C dla Mule jest praktycznie identyczna z tą wartością dla WSO2. W odróżnieniu od Talend, wartość T_{ESB}/T_C dla Mule wzrasta wraz ze wzrostem liczby użytkowników (dla 6 przypadków testowych).

Tabela 6 zawiera zestawienie wartości przepływności magistral dla wszystkich przypadków testowych.

Tabela 6

Zestawienie wartości przepływności magistral

Przepływność *P* [liczba komunikatów/s]									
Nr przypadku testowego	WSO2			MULE			TALEND		
	5	20	50	5	20	50	5	20	50
1	106,27	42,94	31,31	64,26	14,88	5,93	69,56	32,49	12,83
2	100,95	38,35	26,02	66,33	14,58	5,78	57,15	20,79	15,15
3	104,53	39,17	28,37	66,36	16,97	6	58,94	34,41	30,94
4	14,59	4,42	1,39	93,55	74,34	57,41	34,56	30,27	30,95
5	97,7	39,1	30,39	68,97	19,18	6,82	57,57	18,04	6,76
6	68,82	19,04	9,09	43,83	10,27	4,05	44,29	40,97	40,63
7	95,44	31,15	18,90	50,32	16,02	5,59	33,01	29,42	29,3

W przypadku przepływności dla wszystkich badanych magistral zaobserwowano tę samą własność. Przepływność magistrali maleje wraz ze wzrostem liczby użytkowników (tab. 6).

Porównując przepływność magistral, najwyższe wartości *P* dla liczby 5 użytkowników uzyskała magistrala WSO2 (dla 6 na 7 przypadków testowych). Jednak wraz ze wzrostem liczby użytkowników jej przewaga maleje. Natomiast dla 50 użytkowników wartość *P* jest najwyższa dla magistrali Talend (4 na 7 przypadków testowych). Najsłabiej przy analizie *P* wypada magistrala Mule, która ma porównywalne wartości do magistrali Talend dla liczby 5 użytkowników. Jednak wraz ze wzrostem liczby użytkowników wartości przepływności maleją i są niższe od wartości przepływności dla magistrali Talend (dla 5 przypadków testowych). Najbardziej zbliżone do siebie wartości przepływności, biorąc pod uwagę liczbę użytkowników, uzyskała magistrala Talend.

Przeanalizowano także uzyskane wartości czasu T_R. Wysoka wartość tego czasu oznacza, że magistrala nie była w stanie na bieżąco obsłużyć pewnej liczby komunikatów i przetrzymywała je aż do momentu, w którym dalsze przetwarzanie było możliwe. Przez to całościowy czas przetwarzania komunikatu rósł mimo niezwiększania czasu przetwarzania przez samą magistralę. Znalazło to swoje odzwierciedlenie w wartościach T_{ESB}/T_C. Wartość T_R jest także skorelowana z przepustowością magistrali i im bardziej czas T_R wzrasta, tym bardziej przepustowość spada.

W wyniku przeprowadzonych testów uzyskano także wartości procentowego wykorzystania mocy procesora CPU. Tabela 7 zawiera zestawienie wartości procentowego wykorzystania mocy procesora dla wszystkich przypadków testowych.

Tabela 7

Zestawienie wartości procentowego wykorzystania mocy procesora

Zajętość procesora CPU [%]									
Nr przypadku testowego	WSO2			MULE			TALEND		
	5	20	50	5	20	50	5	20	50
1	21,7	38,4	50,4	46	50	55	38,8	62	84,9
2	23,4	41,4	54,8	44	50	58	46,8	77,3	81,1
3	25,3	36,6	48,6	42	54	61	44	61,8	73,4
4	28,2	43,1	47,2	25	45	52	32,5	41,6	38,5
5	18,8	35,7	48,2	33,7	44	51	47	80,1	85,7
6	28,5	60,8	73,2	42	50	71	27,4	27,9	36,1
7	24	40	50,5	25	48	64	20,5	23,2	24,4

Dla magistrali WSO2 średnia wartość CPU ze wszystkich eksperymentów wynosi 39,94%. Dla magistrali WSO2 wartość CPU rosła wraz ze wzrostem liczby użytkowników, jednak dla większości eksperymentów — 20/21 (95%) — nie przekroczyła 61%. Dla magistrali Mule średnia wartość CPU ze wszystkich eksperymentów wynosi 48,13%. Dla magistrali Mule wartość CPU rosła wraz ze wzrostem liczby użytkowników, jednak dla większości eksperymentów — 19/21 (90,5%) — nie przekroczyła 61%. Dla magistrali Talend średnia wartość CPU ze wszystkich eksperymentów wynosi 50,24%. Dla magistrali Talend wartość CPU rosła wraz ze wzrostem liczby użytkowników. Tym razem tylko w 13 eksperymentach na 21 (61,91%) wartość CPU nie przekroczyła 61%. Ponadto, tylko dla Talend wartość CPU przekroczyła 80% i to w aż 4 eksperymentach na 21 (19,05%). Warto tutaj nadmienić, że dla przypadku testowego nr 4 dla 50 użytkowników wartość CPU jest niższa niż dla 20 użytkowników.

Można zatem wyciągnąć wniosek, że dla wszystkich magistral i dla wszystkich przypadków testowych wraz ze wzrostem liczby użytkowników procentowe wykorzystanie mocy procesora CPU wzrasta. Można też zauważyć, że magistrala Mule uzyskuje najbardziej zbliżone do siebie wartości CPU niezależnie od przypadku testowego i liczby użytkowników. Ponadto dla tej magistrali w 19 eksperymentach na 21 (90,5%) wartość CPU nie przekracza 61%. Dla magistrali WSO2 w aż 20 eksperymentach na 21 (95%) wartość CPU nie przekracza 61%. Dzięki temu średnia wartość CPU dla magistrali WSO2 jest najniższa z porównywanych magistral. Dla porównania dla magistrali Talend tylko w 13 eksperymentach na 21 (61,91%) wartość CPU nie przekracza 61% i dlatego uzyskała najwyższą średnią wartość CPU.

Warto także uwypuklić wyniki uzyskane przez magistrale dla liczby 50 użytkowników. Magistrala Talend aż dla 4 przypadków testowych uzyskała najwyższą wartość CPU (przypadki testowe o numerach: 1, 2, 3 oraz 5), dodatkowo wynoszącą ponad 70%. Z drugiej strony, dla trzech przypadków testowych (numer: 4, 6 oraz 7)

uzyskała ona najniższą wartość CPU. Magistrala Mule dla 2 przypadków testowych uzyskała najwyższą wartość CPU (przypadki testowe o numerach: 4 oraz 7) wynoszącą ponad 50%. Ani razu magistrala Mule nie uzyskała najniższej wartości CPU. Magistrala WSO2 aż dla 4 przypadków testowych uzyskała najniższą wartość CPU (przypadki testowe o numerach: 1, 2, 3 oraz 5), ale też raz uzyskała najwyższą wartość CPU (przypadek testowy nr 6). Najbardziej efektywnie korzysta z mocy procesora magistrala WSO2. Wyniki eksperymentów pokazały, że ma ona jeszcze duży zapas mocy procesora w przypadku zwiększania obciążenia magistrali komunikatami. Z drugiej strony, należy podkreślić, że magistrala Talend bardzo umiejętnie korzysta z mocy procesora. Zwiększa ona wykorzystanie procesora ze względu na potrzeby transmisji komunikatów. Mocniej absorbuje procesor, zapewniając przewidywalne czasy transmisji komunikatów.

Analiza wyników testów pokazuje, że magistrala WSO2 uzyskuje dużo lepsze rezultaty dla plików o małych rozmiarach. Transmisja dużych plików może prowadzić do gwałtownego wydłużania czasów transmisji — siedmiokrotnie dłuższy czas transmisji niż u pozostałych dwóch magistral. Najbardziej wydajna w transmisji dużych plików (przypadki testowe nr: 2, 3, 6 oraz 7) okazała się platforma Talend, osiągając krótki czas przetwarzania komunikatu, małe odchylenie standardowe oraz wysoką przepływność, jednak przy wykorzystaniu procesora w granicach 70-85%. Warto także jeszcze raz podkreślić wyniki uzyskane przez magistralę Talend dla liczby 50 użytkowników. Magistrala ta w żadnym z przypadków testowych nie uzyskała najdłuższego czasu T_{ESB}. Ponadto, dla trzech przypadków testowych (numer: 3, 6 oraz 7) uzyskała najkrótszą wartość tego czasu.

8. Podsumowanie i kierunki dalszych prac

W artykule przedstawiono wyniki badań wydajności rozwiązania integracyjnego zbudowanego każdorazowo przy wykorzystaniu jednej z trzech badanych magistral usług firm: WSO2, Mule oraz Talend. Rozwiązanie integracyjne zbudowano do realizacji jednego przypadku biznesowego dotyczącego zakupów w sklepie internetowym sieci sprzedającej sprzęt elektroniczny. Warto zauważyć, że magistrala Talend uzyskuje najbardziej zbliżone do siebie czasy przetwarzania komunikatu T_{ESB} niezależnie od rodzaju testu i liczby użytkowników. Warto też przypomnieć, że magistrala Talend (dla 50 użytkowników) w 3 przypadkach testowych na 7 uzyskała najkrótszą wartość czasu T_{ESB}, a w żadnym z przypadków testowych nie uzyskała najgorszego wyniku. Najmniejsze wartości odchylenia standardowego czasu T_{ESB} uzyskała magistrala Talend, bo aż w 14 eksperymentach, oraz w żadnym eksperymencie nie uzyskała najgorszej wartości tego parametru, a największe wartości uzyskała magistrala WSO2. Widać z tego, że najbardziej zbliżone do siebie wyniki poszczególnych testów uzyskała magistrala Talend.

Porównując przepływność magistral, najwyższe wartości *P* dla liczby 5 użytkowników uzyskała magistrala WSO2 (dla 6 na 7 przypadków testowych). Jednak wraz ze wzrostem liczby użytkowników jej przewaga maleje. Natomiast dla 50 użytkowników wartość *P* jest najwyższa dla magistrali Talend (4 na 7 przypadków testowych). Najsłabiej przy analizie *P* wypada magistrala Mule. Najbardziej zbliżone do siebie wartości przepływności, biorąc pod uwagę liczbę użytkowników, uzyskała magistrala Talend. Ponadto, dla wszystkich magistral wraz ze wzrostem liczby użytkowników procentowe wykorzystanie mocy procesora CPU wzrasta.

Z analizy wyników czasu można wyciągnąć wniosek, że magistrala Talend jest najlepszym wyborem przy budowie rozwiązania integracyjnego w środowisku biznesowym ze zróżnicowanym charakterem napływających komunikatów wymagających transmisji między systemami informatycznymi. Można także wyciągnąć wniosek, że ma największy potencjał w obszarze zwiększania obciążenia: liczby przesyłanych komunikatów oraz liczby użytkowników. Ponadto, należy podkreślić, że przewidywalność czasów transmisji komunikatu magistrali Talend jest ogromną zaletą w zastosowaniach biznesowych, gdzie często określane są wymagania dotyczące wydajności stanowiące podstawowe parametry dla Service Level Agreement.

Z punktu widzenia przeprowadzenia testów wydajności trzeba nadmienić, że w przypadku magistral Mule oraz Talend należy skonfigurować środowisko wykonawcze, do którego wgrywane będą zaimplementowane usługi. Natomiast magistrala WSO2 po pobraniu ze strony producenta działa na zasadzie „out of the box" i po rozpakowaniu pobranego pliku można od razu przystąpić do implementowania przepływów integracyjnych. Jednak uzyskanie wyników dla magistrali WSO2 możliwe było tylko dzięki modyfikacji domyślnej konfiguracji (tab. 1). Przy ustawieniach standardowych nie można było ukończyć wszystkich przypadków testowych ze względu na problemy wydajnościowe magistrali. Wprowadzona modyfikacja umożliwiła przeprowadzenie wszystkich testów, jednak mogła przyczynić się do uzyskania lepszych wyników niż pozostałe magistrale. Ponadto, dla wszystkich magistral istnieje duży zbiór wiedzy wraz z przykładami na stronach producentów oraz aktywne fora, na których można uzyskać odpowiedź w razie napotkanych problemów.

Podsumowując, przeprowadzone testy magistral usług typu Open Source pokazały, że uzyskują one krótkie czasy transmisji komunikatów przy efektywnym wykorzystaniu zasobów sprzętowych. Magistrale te mogą być brane pod uwagę jako element rozwiązań integracyjnych.

W toku dalszych prac planowane jest badanie wpływu różnych wartości parametru „liczba wątków" na wydajność magistral usług. W dalszych badaniach mierzona będzie także zajętość pamięci operacyjnej. Planowane są również badania wydajności magistral z zastosowaniem wzorców architektury usługowej. Celem tych badań będzie weryfikacja wpływu zastosowania wzorców na utrzymanie określonego poziomu wydajności rozwiązania integracyjnego w warunkach zwiększającego się obciążenia.

Źródło finansowania: środki własne autorów.

Artykuł wpłynął do redakcji 13.02.2017 r. Zweryfikowaną wersję po recenzjach otrzymano 15.03.2017 r.

LITERATURA

[1] He W., Xu L.D., *Integration of Distributed Enterprise Applications: A Survey*, IEEE Transactions on Industrial Informatics, 2012.

[2] IBM Integration Bus, http://www.ibm.com/middleware/integration/en-us/enterprise-service-bus-esb.html (dostęp: 18 grudnia 2016).

[3] TIBCO Business Works, http://www.tibco.com/products/automation/application-integration/activematrix-businessworks/enterprise-service-bus (dostęp: 18 grudnia 2016).

[4] Oracle Service Bus, http://www.oracle.com/us/products/middleware/soa/service-bus/overview/index.html (dostęp: 18 grudnia 2016).

[5] JBoss Fuse, http://www.jboss.org/products/fuse/overview/ (dostęp: 18 grudnia 2016).

[6] Mule ESB, https://www.mulesoft.com/platform/soa/mule-esb-open-source-esb (dostęp: 18 grudnia 2016).

[7] Petals ESB, http://petals.ow2.org (dostęp: 18 grudnia 2016).

[8] WSO2 Enterprise Service Bus, http://wso2.com/products/enterprise-service-bus/ (dostęp: 18 grudnia 2016).

[9] OpenESB, http://www.open-esb.net (dostęp: 18 grudnia 2016).

[10] Talend Application Integration, https://www.talend.com/products/application-integration (dostęp: 18 grudnia 2016).

[11] Vollmer K., Gilpin M., Sander R., *The Forrester WaveTM: Enterprise Service Bus*, Q2. Forrester Research Report, 2011.

[12] Umar A., Zordan A., *Reengineering for Service Oriented Architectures: A Strategic Decision Model for Integration Versus Migration*, Journal of Systems and Software, 82(3), 2008, pp. 448-462.

[13] WSO2 ESB Performance Round 6.5, http://wso2.com/library/articles/2013/01/esb-performance-65/

[14] WSO2 ESB Performance Round 7.5, http://wso2.com/library/articles/2014/02/esb-performance-round-7.5/

[15] AdroitLogic, http://esbperformance.org/#ESBPerformanceTesting-Round7-Overview.

[16] Górski T., *Performance analysis of selected frameworks for an integration platform development*, Bulletin of Institute of Computer and Information Systems, 7, 2011, pp. 9-17.

[17] Martínez-Carreras M.A., García Jimenez F.J. & Gómez Skarmeta A.F., *Building integrated business environments: analysing open-source ESB*, Enterprise Information Systems, 7575 (May), 2013, pp. 1-35.

[18] Becker S., Koziolek H. & Reussner R., *The Palladio component model for model-driven performance prediction*, Journal of Systems and Software, 82(1), 2009, pp. 3-22.

[19] Costa B. et al., *Evaluating REST architectures-Approach, tooling and guidelines*, Journal of Systems and Software, 112, 2014, pp. 156-180.

[20] Närman P. et al., *Using enterprise architecture analysis and interview data to estimate service response time*, Journal of Strategic Information Systems, 22(1), 2013, pp. 70-85.

[21] Brosig F., Huber N. & Kounev S., *Architecture-level software performance abstractions for online performance prediction*, Science of Computer Programming, 90(PART B), 2014, pp. 71-92.

[22] Casale G., *Exact analysis of performance models by the Method of Moments*, Performance Evaluation, 68(6), 2011, pp. 487-506.

[23] Sharma V.S. & Trivedi K.S., *Quantifying software performance, reliability and security: An architecture-based approach*, Journal of Systems and Software, 80(4), 2007, pp. 493-509.

[24] Teixeira M. et al., *A quality-driven approach for resources planning in Service-Oriented Architectures*, Expert Systems with Applications, 42(12), 2015, pp. 5366-5379.

[25] Md Fudzee M.F. & Abawajy J.H., *QoS-based adaptation service selection broker*, Future Generation Computer Systems, 27(3), 2011, pp. 256-264.

[26] Ko J.M., Kim C.O. & Kwon I.H., *Quality-of-service oriented web service composition algorithm and planning architecture*, Journal of Systems and Software, 81(11), 2008, pp. 2079-2090.

[27] Yang Y. et al., *Generalized aggregate Quality of Service computation for composite services*, Journal of Systems and Software, 85(8), 2012, pp. 1818-1830.

[28] Menascé D.A., Ruan H. & Gomaa H., *QoS management in service-oriented architectures*, Performance Evaluation, 64(7-8), 2007, pp. 646-663.

[29] Chang C., Srirama S.N. & Ling S., *Towards an adaptive mediation framework for Mobile Social Network in Proximity*, Pervasive and Mobile Computing, 12, 2014, pp. 179-196.

[30] Koziolek H., *Performance evaluation of component-based software systems: A survey*, Performance Evaluation, 67(8), 2010, pp. 634-658.

[31] Hachicha M. et al., *Performance assessment architecture for collaborative business processes in BPM-SOA-based environment*, Data and Knowledge Engineering, 105, 2016, pp. 73-89.

[32] Mi N. et al., *Performance impacts of autocorrelated flows in multi-tiered systems*, Performance Evaluation, 64(9-12), 2007, pp. 1082-1101.

[33] Mentis A., Katsaros P. & Angelis L., *A simulation process for asynchronous event processing systems: Evaluating performance and availability in transaction models*, Simulation Modelling Practice and Theory, 29, 2012, pp. 66-77.

[34] Górski T., *Symulacyjne środowisko badania wydajności platformy integracyjnej rejestrów medycznych*, Roczniki Kolegium Analiz Ekonomicznych, 29, 2013, pp. 595-610.

[35] Negash B. et al., *LISA: Lightweight internet of things service bus architecture*, Procedia Computer Science, 52(1), 2015, pp. 436-443.

[36] Sachs K. et al., *Performance evaluation of message-oriented middleware using the SPECjms2007 benchmark*, Performance Evaluation, 66(8), 2009, pp. 410-434.

[37] Bezemer C.P. & Zaidman A., *Performance optimization of deployed software-as-a-service applications*, Journal of Systems and Software, 87(1), 2014, pp. 87-103.

[38] Glatard T. et al., *A Service-Oriented Architecture enabling dynamic service grouping for optimizing distributed workflow execution*, Future Generation Computer Systems, 24(7), 2008, pp. 720-730.

[39] Iqbal R. et al., *Integration, optimization and usability of enterprise applications*, Journal of Network and Computer Applications, 36(6), 2013, pp. 1480-1488.

[40] Górski T., *The use of Enterprise Service Bus to transfer large volumes of data*, Journal of Theoretical and Applied Computer Science, 8(4), 2014, pp. 72-81.

[41] Li W. et al., *Performance improvement techniques for geospatial web services in* a *cyberinfrastructure environment — A case study with* a *disaster management portal*, Computers, Environment and Urban Systems, 54, 2015, pp. 314-325.

[42] Potena P., *Optimization of adaptation plans for* a *service-oriented architecture with cost, reliability, availability and performance tradeoff*, Journal of Systems and Software, 86(3), 2013, pp. 624-648.

[43] Wu B. et al., *Workflow-based resource allocation to optimize overall performance of composite services*, Future Generation Computer Systems, 25(3), 2009, pp. 199-212.

[44] Górski T., *UML profiles for architecture description of an integration platform*, Biuletyn Wojskowej Akademii Technicznej/Bulletin of the Military University of Technology, 62, 2, 2013, pp. 43-56.

[45] WSO2 ESB PassThrough Transport, http://wso2.com/library/articles/2013/12/demystifying-wso2-esb-pass-through-transport-part-i/.

T. GÓRSKI, K. PIETRASIK

Performance analysis of Enterprise Service Buses of Open Source type

Abstract. The growing interest in business integration and interoperability of IT systems led to an increase in the importance of Service-Oriented Architecture (SOA), which provides tools for Enterprise Application Integration (EAI). In this sense, Enterprise Service Bus (ESB) provides technical capabilities for communication between IT systems.

The aim of this article is to present the results of performance tests of selected ESBs of Open Source type. The basis for our performance analysis was a business case of the order realization in the Internet shop of electronics retailer. The order realization requires a collaboration of three IT systems. The business case has been implemented with the use of each selected ESBs: WSO2, Mule, and Talend. Test scenarios were defined and performance tests were conducted for each of the three selected ESBs. On the basis of the collected results it can be concluded that each service bus has its own strengths and weaknesses. Focusing on the merits it should be noted that WSO2 copes very well with files of small size, for small number of users, and also does not burden the CPU significantly. Results for Talend ESB, show that it does well with both kind of files, small and large sizes, for large number of users, but harder utilizes CPU. Research shows that it is worth to analyze the parameters of transmitted messages while selecting ESB for building integration solution. The results show that the Talend ESB may be a good choice for construction of integration solution in a business environment with a large number of users and diversified communications.

Keywords: Enterprise Service Bus, performance, integration, Service Oriented Architecture
DOI: 10.5604/01.3001.0009.9487