Roczniki Kolegium Analiz Ekonomicznych
nr 29/2013

Tomasz Górski

Wydział Cybernetyki
Wojskowa Akademia Techniczna w Warszawie

Symulacyjne środowisko badania wydajności platformy integracyjnej rejestrów medycznych

1. Wprowadzenie

Budowa rozwiązań integracyjnych wielu systemów informatycznych jest przedsięwzięciem złożonym. Jedną z kluczowych kwestii jest określenie komunikacji między systemami informatycznymi. Sposoby wymiany danych i udostępniania funkcji systemów informatycznych powinny zostać precyzyjnie zaprojektowane. Musi być również zapewnione odpowiednie środowisko komunikacyjne dla integrowanych systemów informatycznych. Architektura usługowa[1] (ang. *Service-Oriented Architecture* – SOA) jest koncepcją tworzenia systemów informatycznych, bazującą na definiowaniu usług, jakie te systemy powinny oferować. Podstawowym elementem w architekturze usługowej jest szyna usług[2] (ang. *Enterprise Service Bus* – ESB). Jest to oprogramowanie umożliwiające sprawną i zestandaryzowaną komunikację między współpracującymi systemami informatycznymi. Usługi współdziałają między sobą przy wykorzystaniu języka XML[3]. Przez platformę integracyjną rozumiemy szynę usług i podłączone do niej systemy informatyczne.

Jednym z kluczowych zagadnień jest zapewnienie odpowiedniego poziomu wydajności platformy integracyjnej. Zagadnienie szacowania i badania wydajności

[1] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall, Crawfordsville (Indiana, USA) 2005.

[2] M. Keen, A. Achraya, *Implementing an SOA Using an Enterprise Service Bus*, IBM Redbook, 2010.

[3] eXtensible Markup Language, http://www.w3.org/XML/ [dostęp 05.12.2012].

złożonych systemów jest obecne w literaturze[4]. Przy tak złożonych systemach trudno jest analitycznie wyliczyć wymagania wydajnościowe projektowanego rozwiązania integracyjnego. Natomiast budowa prototypu w celu weryfikacji wymaganej wydajności wiąże się z czasem oraz kosztami. Prototyp taki także nie da dokładnego oszacowania wymagań wydajnościowych dla kompletnej platformy integracyjnej. W takiej sytuacji, aby oszacować wymaganą wydajność platformy integracyjnej, można zastosować symulację komputerową. W tym celu należy opisać widoki architektoniczne platformy integracyjnej przedstawiające interakcje między systemami informatycznymi. Następnie należy zbudować model symulacyjny platformy integracyjnej. Kolejnym krokiem jest utworzenie środowiska symulacyjnego umożliwiającego dostosowanie modelu platformy do parametrów rozpatrywanego rozwiązania integracyjnego. Po przeprowadzeniu eksperymentów symulacyjnych i dokonaniu estymacji miar wydajności można oszacować parametry platformy integracyjnej wymagane do osiągnięcia określonego poziomu wydajności rozwiązania integracyjnego. W artykule przedmiotem rozważań jest integracja rejestrów medycznych w Polsce. Tematyka medyczna, a w szczególności integracja rejestrów medycznych oraz modelowanie i symulacja procesów medycznych, jest przedmiotem aktualnie realizowanych projektów informatycznych oraz badań naukowych[5].

[4] T. Górski, *Badanie wydajności wybranych środowisk budowy platform integracyjnych*, „Biuletyn" Wojskowej Akademii Technicznej, t. 61, nr 1, Warszawa 2012, s. 353–372; T. Nowicki, *Efficiency estimation of organization described by workflow model*, w: *Contemporary corporate management*, Publishing House of Poznań University of Technology, Poznań 2009; S. Becker, H. Koziolek, R. Reussner, *The Palladio component model for model-driven performance prediction*, „The Journal of Systems and Software" 2009, no. 82, s. 3–22; J. Happe, S. Becker, C. Rathfelder, *Parametric performance completions for model-driven performance prediction*, „Performance Evaluation" 2010, no. 67, s. 694–716; E. Burger, R. Reussner, *Performance certification of Software Components*, „Electronic Notes in Theoretical Computer Science" 2011, no. 279, s. 33–41.

[5] T. Górski, *Architektura platformy integracyjnej dla elektronicznego obiegu recept*, „Roczniki" Kolegium Analiz Ekonomicznych SGH, z. 25, Oficyna Wydawnicza SGH, Warszawa 2012, s. 67–83; T. Nowicki, G. Bliźniuk, M. Lignowska, *Badanie efektywności procedur medycznych zapisanych w postaci ścieżek klinicznych*, „Roczniki" Kolegium Analiz Ekonomicznych SGH, z. 25, Oficyna Wydawnicza SGH, Warszawa 2012, s. 37–53; R. Waszkowski, A. Chodowska, *Zasady wykonywania zadań automatycznych z przekazywaniem sterowania do podsystemów wspomagania decyzji działających na bazie modeli dynamicznych oraz symulacji komputerowej*, w: *Modelowanie i symulacja procesów oraz określenie komputerowo wspomaganych procedur w zakresie zarządzania ryzykiem bezpieczeństwa żywności i żywienia*, red. J. Bertrandt, K. Lasocki, BELStudio, Warszawa 2012, s. 1317–1320; G. Bliźniuk, *Koncepcja implementacji warunków interoperacyjności systemu ścieżek klinicznych i elektronicznego rekordu pacjenta*, „Biuletyn" Instytutu Systemów Informatycznych, nr 6, Warszawa 2010, s. 1–10.

2. Rejestry medyczne

Funkcjonowanie służby zdrowia w Polsce jest regulowane wieloma aktami prawnymi. Wśród nich można wymienić m.in. następujące:

- ustawa z dnia 6 września 2001 r. Prawo farmaceutyczne (tj. Dz.U.04.53.533 z późn. zm.),
- ustawa z dnia 27 sierpnia 2004 r. o świadczeniach opieki zdrowotnej finansowanych ze środków publicznych (Dz.U.04.210.2135 z późn. zm.),
- ustawa z dnia 5 grudnia 1996 r. o zawodach lekarza i lekarza dentysty (tj. Dz.U.05.226.1943 z późn. zm.),
- ustawa z dnia 5 lipca 1996 r. o zawodach pielęgniarki i położnej (tj. Dz.U.01.57.602),
- ustawa z dnia 30 sierpnia 1991 r. o zakładach opieki zdrowotnej (tj. Dz.U.07.14.89 z późn. zm.),
- ustawa z dnia 8 września 2006 r. o Państwowym Ratownictwie Medycznym (Dz.U.06.191.1410 z późn. zm.).

Z analizy powyższych dokumentów wynika, że w służbie zdrowia są eksploatowane m.in. następujące typy rejestrów medycznych:

- podmiotowe – gromadzi się w nich i udostępnia dane o podmiotach funkcjonujących w ochronie zdrowia,
- przedmiotowe – gromadzi się w nich elementarne dane statystyczne dotyczące różnych grup chorób.

W rejestrach podmiotowych są dane pochodzące bezpośrednio od rejestrowanych obiektów i stanowią one źródło danych o tych obiektach. Wśród istniejących rejestrów podmiotowych można wyróżnić m.in.:

- Centralny Rejestr Lekarzy,
- Centralny Rejestr Pielęgniarek i Położnych,
- Rejestr Diagnostów Laboratoryjnych,
- Rejestr Zakładów Opieki Zdrowotnej,
- Rejestr Jednostek Ratownictwa Medycznego,
- Rejestr Aptek Ogólnodostępnych, Aptek Szpitalnych i Punktów Aptecznych,
- Rejestr Produktów Leczniczych Dopuszczonych do Obrotu na Terytorium Rzeczypospolitej Polskiej,
- Wykaz Refundowanych Produktów Leczniczych i Wyrobów Medycznych,
- Centralny Wykaz Ubezpieczonych.

W rejestrach przedmiotowych gromadzi się dane statystyczne dotyczące różnych grup chorób. Wśród rejestrów przedmiotowych wyróżnić można m.in.:

- Krajowy Rejestr Nowotworów,
- Narodowy Rejestr Raka Płuca,
- Centralny Rejestr Gruźlicy,
- Krajowy Rejestr Operacji Kardiochirurgicznych (KROK),
- Rejestr Zachorowań na Choroby Zakaźne i Dodatnich Wyników Badań Laboratoryjnych,
- Centralny Rejestr Chorób Zawodowych.

Pomiędzy wyróżnionymi rejestrami a rejestrami referencyjnymi takimi, jak PESEL, TERYT czy KEP musi zachodzić komunikacja.

Przy tak dużej liczbie rejestrów i wymaganej liczbie połączeń między nimi określenie wymagań wydajnościowych staje się istotnym problemem. Pierwszym krokiem jest dokonanie opisu architektonicznego komunikacji między rejestrami mającej się odbywać dzięki zbudowanej platformie integracyjnej. Do tego celu niezbędny jest dostosowany do potrzeb rozwiązań integracyjnych opis architektoniczny.

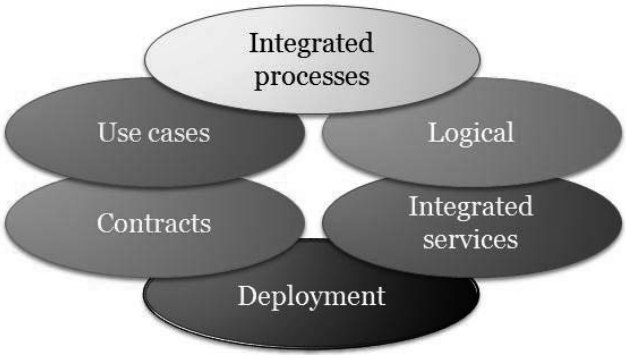
3. Opis architektoniczny

Istnieje wiele modeli z różnymi zestawami widoków architektonicznych, np.: 4+1, RM-ODP, model Siemens, widoki SEI[6], nie zapewniają one jednak możliwości kompletnego opisu architektury platform integracyjnych. Dlatego zaproponowano model widoków architektonicznych 1+5[7], dostosowany do potrzeb modelowania platform integracyjnych. W modelu tym wyodrębniono następujące widoki architektoniczne:

- integrowanych procesów (ang. *integrated processes*),
- przypadków użycia (ang. *use cases*),
- logiki (ang. *logical*),
- integrowanych usług (ang. *integrated services*),
- kontraktów (ang. *contracts*),
- rozlokowania (ang. *deployment*).

[6] N. Rozanski, E. Woods, *Software Systems Architecture. Working with stakeholders using Viewpoints and Perspectives*, Addison Wesley, Crawfordsville (Indiana, USA) 2005.

[7] T. Górski, *Architectural view model for an integration platform*, „Journal of Theoretical and Applied Computer Science" 2012, vol. 6, no. 1, s. 25–34.



Rysunek 1. Model widoków architektonicznych 1+5

Źródło: opracowanie własne.

Podstawowym widokiem architektonicznym jest widok integrowanych procesów. Następne cztery widoki służą do przedstawienia projektu platformy integracyjnej. Natomiast wszystkie produkty powstające w ramach architektury platformy integracyjnej umożliwiają zaprojektowanie systemu funkcjonującego w określonym środowisku uruchomieniowym. W widoku integrowanych procesów są modelowane procesy biznesowe, które mają być zautomatyzowane na platformie integracyjnej. Widok przypadków użycia zawiera wymagania funkcjonalne na system informatyczny integrowany w ramach platformy integracyjnej. W widoku integrowanych usług są prezentowane usługi wystawiane z systemów informatycznych oraz sposób ich włączenia na korporacyjną szynę usług. Widok kontraktów przedstawia komponenty reprezentujące systemy informatyczne i kontrakty określone między tymi systemami. W widoku tym są przedstawiane także przepływy mediacyjne dla każdego kontraktu. W rozpatrywanym podejściu zostały zaproponowane modele oraz diagramy języków BPMN[8] oraz UML[9] z rozszerzeniem o konstrukcje języka SoaML[10] do modelowania architektury platformy integracyjnej. Szczegółowy opis przedstawianego modelu widoków architektonicznych znajduje się w literaturze[11].

[8] Business Process Model and Notation (BPMN) 2.0, OMG 2011, http://www.omg.org/spec/BPMN/2.0/.

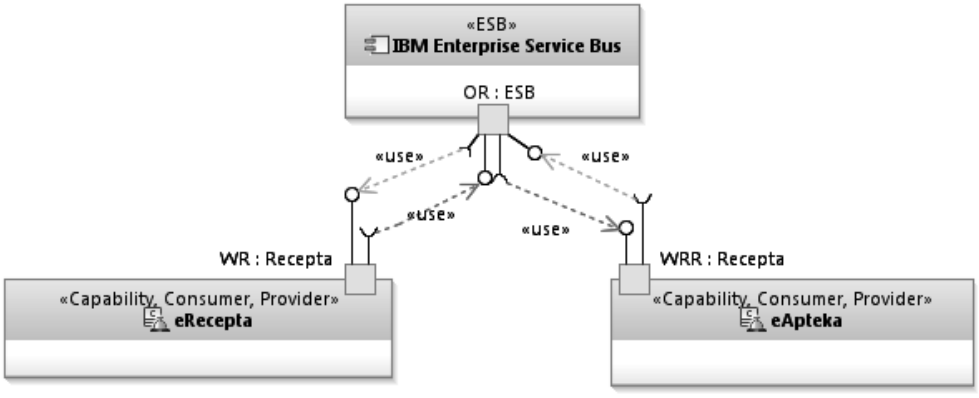
[9] OMG Unified Modeling Language Specification Version 2.3, May 2010, http://www.omg.org/spec/UML/2.3/.

[10] Service oriented architecture Modeling Language (SoaML) Specification for the UML Profile and Metamodel for Services (UPMS) v. 1.0, 2008, http://www.omgwiki.org/SoaML/doku.php?id=specification.

[11] T. Górski, *Platformy integracyjne. Zagadnienia wybrane*, Wydawnictwo Naukowe PWN, Warszawa 2012, s. 109–135; T. Górski, *Architectural view model for an integration platform*…, op.cit.

4. Opis usług i przepływów mediacyjnych na platformie integracyjnej

Wykorzystując widoki architektoniczne modelu 1+5, należy przedstawić rejestry oraz usługi włączane na platformę integracyjną. Usługi wystawiane są na platformę integracyjną za pomocą języka WSDL[12]. Usługi wystawiane z poszczególnych rejestrów przedstawiane są na diagramie komponentów języka UML. Rejestr wystawiający usługę oznaczany jest stereotypem <<provider>>, natomiast rejestr korzystający z usługi poprzez platformę integracyjną oznaczany jest stereotypem <<consumer>>. Obydwa te stereotypy są elementami języka SoaML. Diagram ten stanowi reprezentację widoku architektonicznego integrowanych usług z modelu 1+5. Rysunek 2 przedstawia przykład diagramu komponentów w widoku integrowanych usług dla wywołań usług między systemami e-Recepta oraz e-Apteka[13]. Na tym diagramie zastosowano stereotyp <<ESB>> z profilu UML „UML Profile for Integration Platform"[14].



Rysunek 2. Diagram komponentów widoku integrowanych usług

Źródło: opracowanie własne.

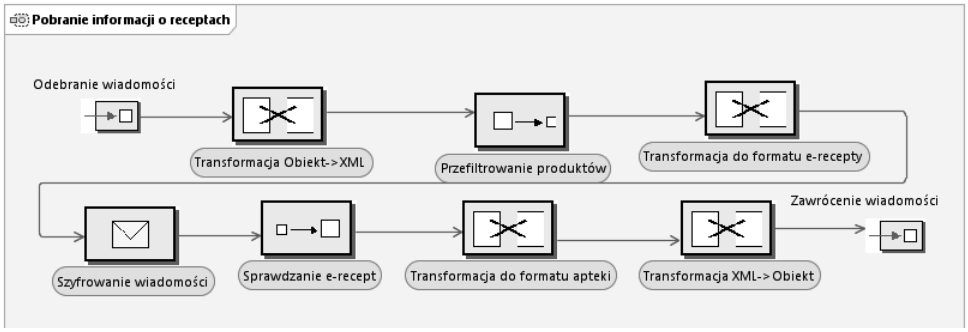
[12] Web Services Description Language (WSDL) Version 2.0, W3C 2007 http://www.w3.org/TR/wsdl20/.

[13] T. Górski, *Architektura platformy integracyjnej dla elektronicznego obiegu recept…*, op.cit.

[14] T. Górski, *Profil „UML Profile for Integration Platform" do modelowania architektury platformy integracyjnej*, w: *Integracja systemów informatycznych – nowe wyzwania*, red. J. Górski, C. Orłowski, PWNT, Gdańsk 2011.

Wymiana danych wiąże się z wyborem formatu danych przesyłanych na platformie integracyjnej. Najczęściej stosowanym formatem jest język XML[15]. Zastosowanie tego języka niesie jednak ze sobą narzuty w rozmiarze przesyłanych plików między systemami. W realizacji platformy integracyjnej należałoby rozważyć zastosowanie formatu danych, który nie dokłada tak znacznych narzutów w rozmiarze plików, np. VTD–XML[16] czy Efficient XML Interchange (EXI)[17].

Ponadto wymiana danych między rejestrami wiąże się często z potrzebą wykonania operacji mediacyjnych[18] przez platformę integracyjną. W widoku integrowanych usług stosuje się diagram przepływów mediacyjnych. Rysunek 3 przedstawia przykład przepływu mediacyjnego przy wywołaniu usługi „Pobranie informacji o receptach". Zastosowano w tym przepływie stereotypy z profilu UML „UML Profile for Integration Flows".



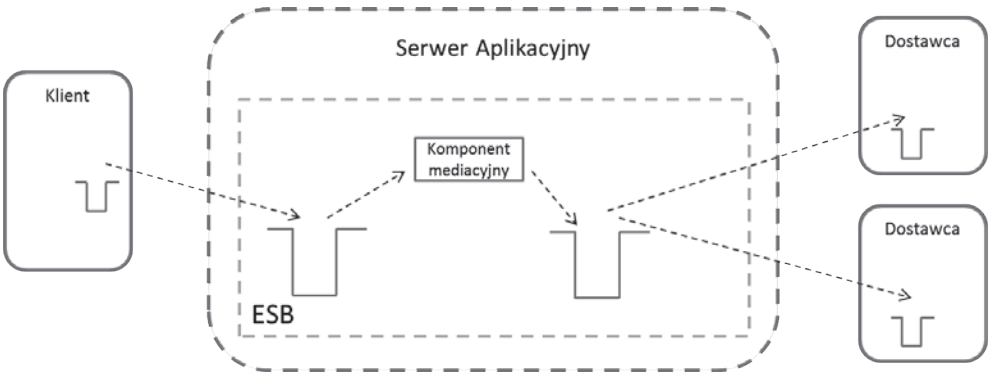
Rysunek 3. Przepływ mediacyjny pobrania informacji o receptach pacjenta
Źródło: opracowanie własne.

5. Model platformy integracyjnej

Komunikacja między rejestrami odbywa się przez szynę usług. Szyna usług jest osadzona na serwerze aplikacyjnym i wykorzystuje mechanizm kolejek do przesyłania komunikatów między rejestrami. Pomiędzy kolejką wejściową a kolejką wyjściową na szynie usług znajduje się komponent mediacyjny, w którym

[15] eXtensible Markup Language, http://www.w3.org/XML/.
[16] VTD–XML, http://vtd-xml.sourceforge.net/.
[17] Efficient XML Interchange (EXI) Format 1.0, http://www.w3.org/TR/exi/.
[18] G. Hohpe, B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging*, Addison-Wesley, Westford (Massachusetts, USA) 2003.

definiowane są przepływy mediacyjne. Rysunek 4 przedstawia poglądowy schemat komunikacji między konsumentem a dostawcą usługi przez szynę usług.



Rysunek 4. Schemat komunikacji przez szynę usług

Źródło: opracowanie własne.

W modelu platformy integracyjnej można wyróżnić następujące elementy:

- aplikację początkową,
- usługę dostawcy,
- szynę usług.

Aplikacja początkowa A_p składa się z generatora, który wysyła żądanie wywołania usługi do szyny usług. Aplikacja początkowa może zawierać także adapter.

$$A_p = <g, k, a>, \tag{1}$$

gdzie:

g – rozkład prawdopodobieństwa czasu generowania komunikatu,

k – rozkład prawdopodobieństwa czasu odpytywania kolejki odpowiedzi (przy wywołaniu asynchronicznym),

a – rozkład prawdopodobieństwa czasu przetwarzania przez adapter.

Usługa dostawcy U jest to element docelowy przepływu, do którego jest wysyłane żądanie z szyny usług. Jej elementy to generator czasu obsługi przetwarzania żądania. W zależności od podłączenia do szyny usług może ona zawierać jeszcze adapter.

$$U = <h, b>, \tag{2}$$

gdzie:

h – rozkład prawdopodobieństwa czasu obsługi żądania,

b – rozkład prawdopodobieństwa czasu przetwarzania przez adapter.

Szyna usług *S* składa się z kolejki wejściowej wywołań usług, przepływu mediacyjnego żądania, przepływu mediacyjnego odpowiedzi (opcjonalnie) oraz kolejki docelowej usług.

$$S = <q_w, P_ż, P_o, q_d>, \tag{3}$$

gdzie:
q_w – kolejka wejściowa usług,
$P_ż$ – przepływ mediacyjny żądania,
P_o – przepływ mediacyjny odpowiedzi,
q_d – kolejka docelowa usług.

Przepływy mediacyjne żądania i odpowiedzi są typu *P*. Przepływ mediacyjny *P* definiuje sekwencję operacji mediacyjnych. Komponent mediacyjny może mieć zdefiniowany tylko przepływ mediacyjny żądania (wywołanie asynchroniczne) albo zarówno przepływ mediacyjny żądania, jak i przepływ mediacyjny odpowiedzi.

$$P = <p_1, \ldots, p_i, \ldots, p_n>, \tag{4}$$

gdzie:
p_i – rozkład prawdopodobieństwa czasu przetwarzania przez *i*-tą operację mediacyjną,
n – liczba elementów w przepływie.

6. Miary wydajności platformy integracyjnej

Jednym z podstawowych wymagań wydajnościowych jest czas uzyskania wyników realizacji usługi. Można przyjąć następujące miary wydajności platformy integracyjnej związane z wywołaniami usług:

- opóźnienie w jedną stronę (ang. *One-Way Delay* – OWD) – jest to czas, jaki zajmuje wiadomości osiągnięcie docelowej lokalizacji, czyli jest liczony od wysłania z punktu źródłowego do dotarcia do celu; na OWD składają się dwa elementy, które związane są z przesyłaniem danych:
  - opóźnienie propagacji – jest to czas pokonania drogi komunikatu od jednego odcinka do drugiego przy uwzględnieniu jedynie szybkości fizycznej łącza, jakim jest przesyłany,
  - opóźnienie przetwarzania (ang. *Serialization Delay*) – jest to czas potrzebny do konwersji czy przetwarzania komunikatu przez całą drogę, którą przebywa,

- opóźnienie w obie strony (ang. *Round Trip Time* – RTT) – jest to czas, w jakim komunikat pokona drogę do punktu docelowego i z powrotem; mogłoby się wydawać, że jest to po prostu dwukrotność czasu opóźnienia w jedną stronę, lecz w rzeczywistości mogą to być całkiem różne czasy,
- maksymalne opóźnienie (ang. *Maximum Delay*) – największa wartość czasu opóźnienia w jedną stronę; jest ono ważne dla aplikacji, które mają założone maksymalne odchylenie czasu opóźnienia,
- zmiana opóźnienia (ang. *Delay Variation*) – jest to zmiana opóźnienia w jedną lub w dwie strony w czasie; może to zależeć od intensywności wysyłanych w jednej chwili komunikatów.

Dla jednostronnych wywołań mierzone jest opóźnienie w jedną stronę OWD. Dla tego typu wywołań typową miarą jest wartość liczby komunikatów obsługiwanych w zadanym okresie. Przy wywołaniach, w których oczekiwana jest odpowiedź, czas jest mierzony od wysłania żądania do momentu odbioru odpowiedzi przez żądającego (RTT).

7. Projekt środowiska symulacyjnego

Przy projektowaniu środowiska symulacyjnego założono, że możliwe są następujące typy interakcji:

- jednostronny,
- synchroniczny: żądanie – odpowiedź,
- asynchroniczny: żądanie – odpowiedź.

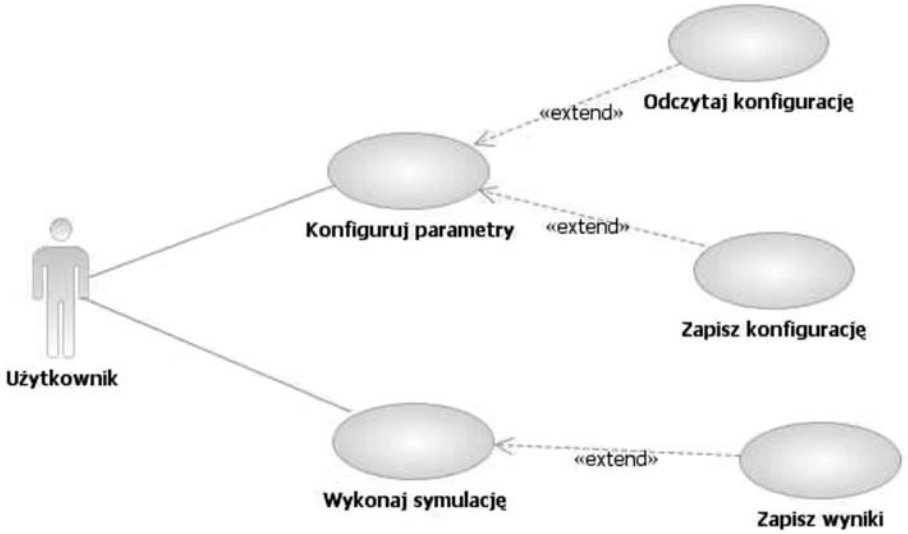
Na rysunku 5 zaproponowano podstawowe przypadki użycia projektowanego środowiska symulacyjnego. Zaprojektowane środowisko pozwala skonfigurować parametry oraz uruchamiać symulację. Jedną z podstawowych funkcji jest możliwość zapisywania oraz odczytywania konfiguracji parametrów z pliku. Użytkownik może wykonać symulację, po której zostaną mu zwrócone wartości wcześniej określonych miar wydajności. Dodatkową funkcjonalnością jest zapis wyników do pliku w celu ewentualnego późniejszego ich wykorzystania.

Aplikacja została podzielona na dwa moduły:

- symulacyjny,
- interfejsu użytkownika.

Szyna usług zawiera dwie kolejki: kolejkę wejściową usług oraz kolejkę docelową usług. Ponadto na szynie definiowane są przepływy mediacyjne. Usługa zawiera generator czasu adaptera oraz generator czasu obsługi. Element aplika-

cja początkowa również zawiera generatory, dodatkowym z nich jest generator czasu odpytywania kolejki odpowiedzi. Elementem centralnym skupiającym drogę przejścia komunikatu na szynie jest przepływ mediacyjny. Zawiera on sekwencję operacji mediacyjnych. Aplikacja symulacyjna została zaprojektowana i następnie zaimplementowana w języku Java. Rysunek 6 przedstawia okno główne zaprojektowanej aplikacji symulacyjnej.

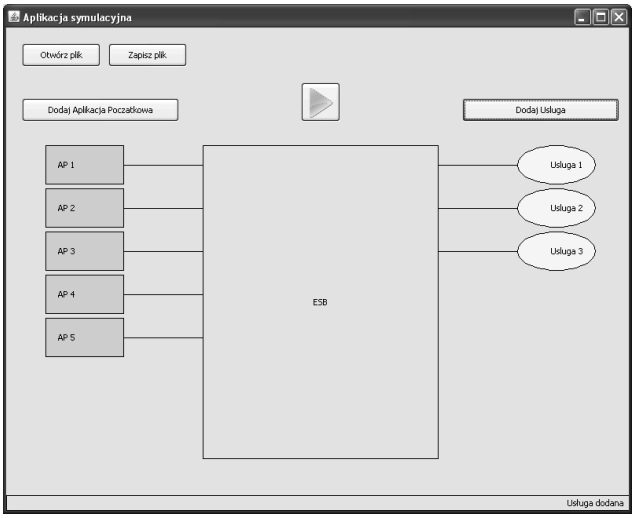


Rysunek 5. Diagram przypadków użycia dla środowiska symulacyjnego platformy integracyjnej

Źródło: opracowanie własne.

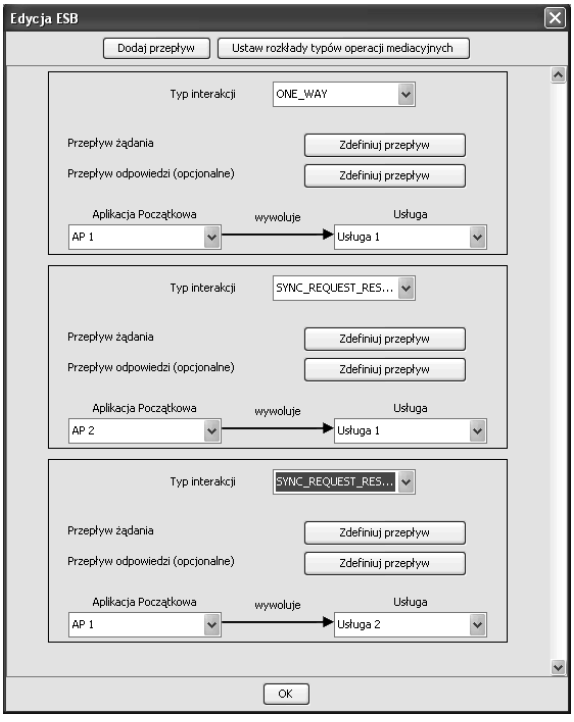
Najbardziej rozbudowanym spośród edytorów w całym środowisku symulacyjnym jest edytor szyny usług (rysunek 7). Edytor ten umożliwia zdefiniowanie przepływów występujących pomiędzy aplikacją a wywoływaną usługą.

Po skonfigurowaniu parametrów symulacyjnych można uruchomić symulację. Wyniki symulacji można zapisać do pliku XML, który jest generowany technologią XMLBeans przy pomocy odpowiedniego schematu zawartego w pliku XSD. Rysunek 8 przedstawia postać takiego pliku XSD.



Rysunek 6. Okno główne aplikacji symulacyjnej

Źródło: opracowanie własne.



Rysunek 7. Edytor szyny usług

Źródło: opracowanie własne.

```
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:po="http://openuri.org/easypo"
    targetNamespace="http://openuri.org/easypo"
    elementFormDefault="unqualified">
    <xs:element name="Results">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="przeplyw" type="po:przeplyw"/>
                <xs:element name="czasSymulacji" type="xs:string" minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="przeplyw">
        <xs:sequence>
            <xs:element name="przeplywNo" type="xs:int"/>
            <xs:element name="avgTimeOWD" type="xs:double"/>
            <xs:element name="avgTimeRTT" type="xs:double"/>
            <xs:element name="maxDelay" type="xs:double"/>
            <xs:element name="avgTimeAskResponseQueue" type="xs:double"/>
            <xs:element name="interactionTypeName" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Rysunek 8. Struktura pliku XSD do generacji wyników symulacji

Źródło: opracowanie własne.

8. Podsumowanie i kierunki dalszych badań

Najważniejszą korzyścią, którą niesie ze sobą stosowanie opracowanego środowiska symulacyjnego, jest możliwość badania wydajności platformy integracyjnej na etapie jej specyfikacji i projektowania. Dzięki zastosowaniu opracowanego środowiska można przeprowadzić symulację różnych przepływów na platformie integracyjnej odpowiadających interakcjom wymaganym między rejestrami medycznymi. Dostępne na rynku środowiska budowy platform integracyjnych są konfigurowalne. Przez umiejętną konfigurację możemy podnieść wydajność projektowanej platformy integracyjnej. Opracowane środowisko symulacyjne umożliwia zbadanie wartości parametrów platformy, przy których będzie działać najbardziej wydajnie. Jednym z elementów, które można sprawdzić, jest dobór wzorców interakcji.

Dzięki zastosowaniu tego typu środowiska symulacyjnego można uniknąć kosztowych błędów projektowych na wczesnych etapach projektowania rozwiązania integracyjnego. Ponadto można oszacować poziom potrzebnych zasobów sprzętowych i określić docelowy ruch komunikacyjny na platformie, a przez

to jej wymaganą przepustowość. Dzięki temu lepiej można zwymiarować sprzęt potrzebny do zakupu w ramach budowy złożonych rozwiązań integracyjnych.

Dalsze prace są ukierunkowane na opracowanie konfiguracji modelu dla wybranych rejestrów medycznych i przeprowadzenie wydajnościowych badań symulacyjnych. Planowane są badania wybranych rejestrów włączanych do aktualnie projektowanej Elektronicznej Platformy Gromadzenia, Analizy i Udostępniania Zasobów Cyfrowych o Zdarzeniach Medycznych. Kolejnym istotnym kierunkiem badań jest automatyzacja projektowania platformy integracyjnej przez zastosowanie podejścia *Model-Driven Architecture*[19]. Trwają także prace nad dopracowaniem konfiguracji autorskiego procesu projektowania platform integracyjnych – *Integration Platform Development Process*.

Bibliografia

1. Becker S., Koziolek H., Reussner R., *The Palladio component model for model-driven performance prediction*, „The Journal of Systems and Software" 2009, no. 82, s. 3–22.
2. Bliźniuk G., *Koncepcja implementacji warunków interoperacyjności systemu ścieżek klinicznych i elektronicznego rekordu pacjenta*, „Biuletyn" Instytutu Systemów Informatycznych, nr 6, Warszawa 2010, s. 1–10.
3. Burger E., Reussner R., *Performance certification of Software Components*, „Electronic Notes in Theoretical Computer Science" 2011, no. 279, s. 33–41.
4. Erl T., *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall, Crawfordsville (Indiana, USA) 2005.
5. Górski T., *Architectural view model for an integration platform*, „Journal of Theoretical and Applied Computer Science" 2012, vol. 6, no. 1, s. 25–34.
6. Górski T., *Architektura platformy integracyjnej dla elektronicznego obiegu recept*, „Roczniki" Kolegium Analiz Ekonomicznych SGH, z. 25, Oficyna Wydawnicza SGH, Warszawa 2012, s. 67–83.
7. Górski T., *Badanie wydajności wybranych środowisk budowy platform integracyjnych*, „Biuletyn" Wojskowej Akademii Technicznej, t. 61, nr 1, Warszawa 2012, s. 353–372.
8. Górski T., *Platformy integracyjne. Zagadnienia wybrane*, Wydawnictwo Naukowe PWN, Warszawa 2012.

[19] H.S. Lahman, *Model-Based Development: Applications*, Pearson Education Inc., 2011.

9. Górski T., *Profil „UML Profile for Integration Platform" do modelowania architektury platformy integracyjnej*, w: *Integracja systemów informatycznych – nowe wyzwania*, red. J. Górski, C. Orłowski, PWNT, Gdańsk 2011, s. 109–118.
10. Happe J., Becker S., Rathfelder C., *Parametric performance completions for model-driven performance prediction*, „Performance Evaluation" 2010, no. 67, s. 694–716.
11. Hohpe G., Woolf B., *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging*, Addison-Wesley, Westford (Massachusetts, USA) 2003.
12. Keen M., Achraya A., *Implementing an SOA Using an Enterprise Service Bus*, IBM Redbook, 2010.
13. Lahman H.S., *Model-Based Development: Applications*, Pearson Education Inc., Westford (Massachusetts, USA) 2011.
14. Nowicki T., *Efficiency estimation of organization described by workflow model*, w: *Contemporary corporate management*, red. K. Grzybowska, A. Stachowiak, Publishing House of Poznań University of Technology, Poznań 2009.
15. Nowicki T., Bliźniuk G., Lignowska M., *Badanie efektywności procedur medycznych zapisanych w postaci ścieżek klinicznych*, „Roczniki" Kolegium Analiz Ekonomicznych SGH, z. 25, Oficyna Wydawnicza SGH, Warszawa 2012, s. 37–53.
16. Rozanski N., Woods E., *Software Systems Architecture. Working with stakeholders using Viewpoints and Perspectives*, Addison Wesley, Crawfordsville (Indiana, USA) 2005.
17. Waszkowski R., Chodowska A., *Zasady wykonywania zadań automatycznych z przekazywaniem sterowania do podsystemów wspomagania decyzji działających na bazie modeli dynamicznych oraz symulacji komputerowej*, w: *Modelowanie i symulacja procesów oraz określenie komputerowo wspomaganych procedur w zakresie zarządzania ryzykiem bezpieczeństwa żywności i żywienia*, red. J. Bertrandt, K. Lasocki, BELStudio, Warszawa 2012, s. 1317–1320.

Źródła sieciowe

1. Business Process Model and Notation (BPMN) 2.0, OMG 2011, http://www.omg.org/ spec/BPMN/2.0/.
2. Efficient XML Interchange (EXI) Format 1.0, http://www.w³.org/TR/exi/.
3. eXtensible Markup Language, http://www.w³.org/XML/.
4. OMG Unified Modeling Language Specification Version 2.3, May 2010, http://www.omg.org/spec/UML/2.3/.
5. *Service oriented architecture Modeling Language (SoaML) Specification for the UML Profile and Metamodel for Services (UPMS) v. 1.0*, 2008, http://www.omgwiki.org/SoaML/doku.php?id=specification.
6. VTD–XML, http://vtd-xml.sourceforge.net/.
7. Web Services Description Language (WSDL) Version 2.0, W3C 2007, http://www.w3.org/TR/wsdl20/.

* * *

Simulation environment for integration platform's performance analysis of medical registries

Summary

The paper presents the problem of medical registries integration using an integration platform in service-oriented architecture. The article focuses on an architectural description of services and interactions between information systems. The main aspect considered in the paper is the performance of the integration platform which connects considerable amount of medical registries. The paper provides an integration platform model. In addition, the performance measures of this class of information systems were presented. The paper also describes a design of simulation environment. The summary contains the benefits of the simulation environment for early estimation and verification of performance requirements of integration platform.

Keywords: integration platforms, architecture of information system, data registry