AUTOMATYKA • 2009 • Tom 13 • Zeszyt 2

Krzysztof Cetnarowicz*, Tadeusz Dyduch*, Jarosław Koźlak*, Małgorzata Żabińska*

Koncepcja zastosowania podejścia agentowego w systemach SOA

1. Wprowadzenie

W ostatnich latach dużą popularność zyskała technologia SOA (*Service Oriented Architecture*), stało się to dzięki jej cechom, zwłaszcza w zakresie elastycznej kompozycji usług webowych i ich wykonania, co umożliwia wykorzystywanie do tworzenia różnych zastosowań. Jednakże brakuje dojrzałych rozwiązań z zakresu automatycznego zestawiania i wywoływania złożonych usług, pomimo intensywnego rozwoju i znacznego zaawansowania narzędzi Semantic Web.

Nasz zespół badawczy podjął prace nad budową systemu opartego na zestawie odpowiednich, współpracujących usług, wybierając jako dziedzinę zastosowania zintegrowaną akcję ratowniczą – automatyzację i optymalizację stanowiska kierowania akcjami ratunkowymi telefonu 112, w zakresie koordynacji działań transportu i usług medycznych, straży pożarnej i policji. Przyjęty przez nas sposób działania można określić jako podejście agentowe do semantycznie opisanych usług sieciowych – *Agent Approach to Semantic Web Services* (AA2SWS). W proponowanej koncepcji systemu, środowisko agentowe realizuje następujące zasadnicze funkcje: dopasowanie i wybór usług, nadzór procesu ich wywoływania oraz modyfikację planów działań w razie takiej potrzeby, integrację oferty odrębnych usługodawców oraz zapewnienie uczenia i doskonalenia się systemu. Używana podczas reprezentacji usług dziedzina pojęć jest opisana językiem ontologicznym, co umożliwia prowadzenie wnioskowań i diagnostyki, zaś oferowane serwisy posiadają opisy OWL-S [6] i WSDL [10] oraz są rejestrowane na serwerze UDDI.

Idea realizacyjna systemu jest następująca. Stanowisko zarządzania odbiera zgłoszenie zdarzenia. Operator, prowadzony przez system ekspertowy, ustala lub wybiera spośród gotowych wzorów proces obsługi zdarzenia, opisany grafem sieci usług. System agentowy dobiera do tego procesu konkretnych usługodawców i ich usługi, kontaktując się z serwerem UDDI i serwisami webowymi usługodawców. Powstały w ten sposób szczegółowy plan realizacji procesu zapisywany jest w języku WS-BPEL (*Web Service-Business*

* Katedra Informatyki, Akademia Górniczo-Hutnicza w Krakowie

Process Execution Language) [4] i realizowany przez serwer aplikacyjny pod nadzorem agenta. Procesy obsługi mają możliwości wywoływania kolejnych procesów, co nadaje im niezbędną elastyczność.

2. Przegląd dziedziny, stosowane metody i narzędzia

Tworząc system tego typu, sięgnąć należy do różnych dziedzin: języków ontologicznej reprezentacji wiedzy do opisu usług, narzędzi do automatycznego dopasowywania usług do zadanych wymagań (*matchmaking*), opisu procesów biznesowych wywołujących różne inne usługi przy użyciu WS-BPEL oraz planerów umożliwiających dynamiczną budowę łańcuchów usług składających się na usługi złożone.

2.1. Opis semantyki usług

Podstawowymi narzędziami do pracy z ontologiami są edytory wspomagające edycję plików ontologicznych, w szczególności OWL [1]. Jednym z najpopularniejszych edytorów jest Protege OWL [9], którego funkcjonalność może być zwiększana przez liczne rozszerzenia pozwalające na zapis wiedzy w różnych formatach, wnioskowanie w oparciu o motory bazujące na logikach opisowych lub systemach regułowych, a także integrację ontologii i ich wizualizację.

Do semantycznego opisu usług webowych może być używana ontologia OWL-S [6] zapisana w języku OWL. Składa się ona z następujących części: profil, model procesu, uziemienie i usługa. Profil (*Profile)* opisuje zdolności i cechy różnicujące *Web Services* stosowane do celów ogłaszania w sieci oraz dopasowywania. Model procesu (*Process model*) dostarcza opisu struktury aktywności dostarczanej przez usługę, z tych informacji użytkownicy usługi mogą uzyskać informacje o sposobie jej uruchomienia oraz wzorcach interakcyjnych. Uziemienie (*Grounding*) zawiera opis, jak abstrakcyjna wymiana informacji opisana w modelu procesu jest mapowana na konkretne komunikaty przepływające między dostarczycielem usługi oraz odbiorcą usługi, natomiast usługa (*Service*) dostarcza środki do powiązania ze sobą informacji z profilu, modelu procesu i uziemienia.

2.2. Dopasowanie usług opisanych semantycznie dla potrzeb systemu SOA

Istotnym aspektem tworzenia systemów w technologii SOA jest dopasowywanie usług, odzwierciedlające charakter potrzeb związanych z dziedziną systemu. Powstały liczne narzędzia do dopasowywania usług, najbardziej rokujące, naszym zdaniem, są opisane poniżej.

OWL-S UDDI Matchmaker [7] łączy możliwości rejestru UDDI oraz semantycznego opisu usług webowych przy użyciu OWL-S. Jest on rozszerzeniem jUDDI do obsługi rejestru UDDI.

OWL-SM [2] stosuje rankingowe dopasowywanie dla usług opisanych w OWL-S, a konkretnie w odpowiednich instancjach klasy *Profile* tej ontologii. Zasadniczymi właści-

wościami *Profile* z punktu widzenia procesu integracji są: *Input*, *Output*, *Precondition* i *Effect*, z których każda jest podklasą klasy *Parameter*. Zakłada się, że odpowiednie reprezentacje tych parametrów muszą być wspierane przez motor wnioskowania. Twórcy przyjmują zatem, że są one dokonane w OWL, a algorytm może operować w oparciu o relację subsumpcji oraz równoważność pojęć/klas. Dopasowujący algorytm składa się z czterech etapów: dopasowanie wejścia, dopasowanie wyjścia, dopasowanie kategorii usługi, rezultaty tych trzech kroków są agregowane w kroku czwartym – gdzie zdefiniowane przez użytkowanika ograniczenia lub funkcjonalności mogą uzupełnić rezultat dopasowania. Podczas trzech pierwszych etapów rezultatem może być: równoważność, subsumcja, niepowodzenie, odpowiedź nieznana.

OWL-MX [3,5] jest hybrydowym matchmakerem usług opisanych w OWL-S oraz wykorzystującym ontologie zapisane w OWL. OWL-MX wykorzystuje motor wnioskowania Pellet oraz metryki podobieństw (oparte na miarach Jacquarda oraz Jensena-Shannona).

3. Koncepcja systemu do komponowania usług

Kompozycja usług stanowi zasadniczą część systemu i ten element determinuje użyteczność rozwiązania. Zastosowanie agentów umożliwia uzyskanie pożądanych cech systemu związanych z jego elastycznością. W ramach prowadzonych prac zaproponowano połączenie koncepcji użycia agentów z technologiami usług webowych. Proponowany system obejmuje zatem dwie warstwy: usługi webowe oraz agentów. Warstwa usług webowych zawiera:

- usługi, dostępne pod zadanymi adresami URI w przestrzeni Internetu, usługi te mają dobrze opisane swoje wejścia oraz wyjścia wyrażone w OWL-S;
- usługi złożone, które mogą stanowić kompozycję innych usług opisaną w języku BPEL;
- repozytoria – zawierają informację na temat dostępnych usług, zwracają informację na temat usług danego typu, zazwyczaj zgodnych ze wzorcem wyrażonym w języku zapytania;
- usługi – brokery – zawierają dobrze sklasyfikowaną i przeanalizowaną informację na temat usług danych typów.

W skład warstwy agentowej wchodzą agenci różnych typów, ich podstawowe zadania są następujące: stworzenie workflow usług, przeprowadzenie wykonania workflow usług oraz kontrola tego wykonania, generacja stworzonych workflow w postaci BPEL oraz ich udostępnianie jako usług złożonych, a także zarządzanie brokerami i uzupełnianie ich wiedzy.

Klient może wchodzić w interakcje z systemem na dwa sposoby: poprzez specyfikację odpowiedniego URI usługi oraz dostarczając jej informacje/zasoby wejściowe oraz pożądane informacje/zasoby wyjściowe lub też specyfikując odpowiednie URI, posiadane informacje/zasoby wejściowe oraz pożądane informacje/zasoby wyjściowe i łącząc się z węzłem agentowym (agentem, który odpowiada na nim za komunikację z otoczeniem), agenci

wchodzący w skład węzła agentowego budują workflow umożliwiający wykonanie zlecenia klienta,

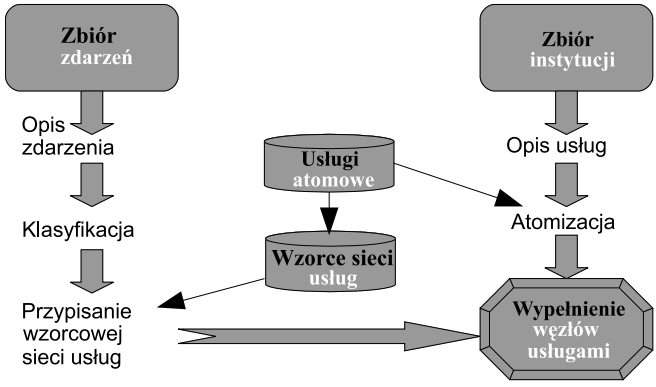
3.1. Zasady opisu usług, ich oferowania i doboru

Kluczowym dla architektury i efektywności całego systemu, opisu dziedziny, kompozycji usług i ich wykonania jest adekwatny opis dostępnych w dziedzinie serwisów webowych (usług).

Zasadnicze znaczenie ma tu **granulacja**: istniejące usługi musimy zdekomponować na taką głębokość, aby móc wyróżnić wszystkie będące przedmiotem zainteresowania i decyzji wyboru, a także, aby istniejące i oferowane usługi opisać jako kompozycje usług atomowych (patrz rys. 1).

Rejestr UDDI usług atomowych, oprócz odesłań do ich opisów w języku WSDL, musi zawierać relacje do innych usług atomowych: bezpośredniego następowania oraz równoczesnego występowania w sensie ich niezbędności. Relacje te opisane są za pomocą omawianego wcześniej OWL-S.

Rysunek 1 ilustruje zasadę działania systemu w oparciu o atomizację oferowanych i dostępnych usług. Zarejestrowane w systemie instytucje oferują określone usługi. Usługi te zarejestrowane są w systemie jako kolekcje usług atomowych (atomizacja). System posiada także zbiór zdarzeń, dla których opracowane są procesy obsługi (wzorce postępowania w postaci grafów sieci usług). Zgłoszenie wystąpienia zdarzenia, wymagającego interwencji, uruchamia proces klasyfikacji, w efekcie którego zdarzeniu przypisany zostaje właściwy proces obsługi. Aby proces ten mógł zostać zrealizowany, należy każdej usłudze, zapisanej w węźle sieci usług, przyporządkować atomową usługę konkretnej instytucji. Tak powstały zestaw zleceń dla konkretnych instytucji, dopasowany do zgłoszonego zdarzenia, może zostać skierowany do realizacji.



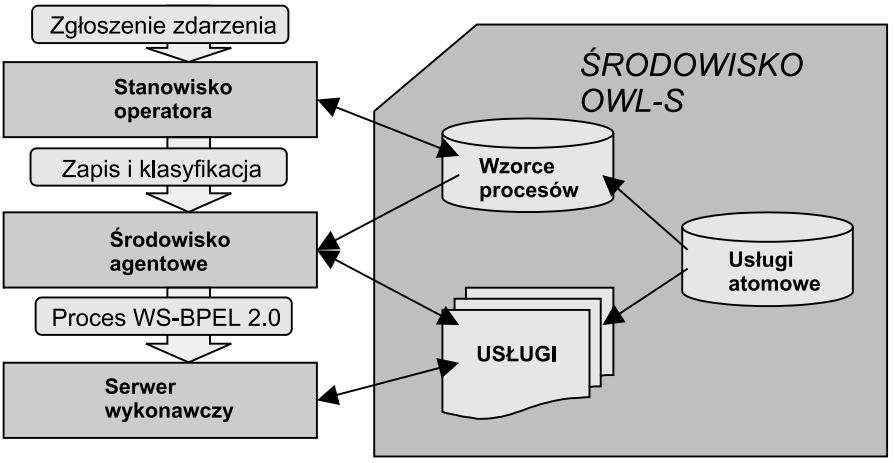
Rys. 1. Ilustracja syntezy złożonych serwisów w oparciu o atomizację usług

Istniejące usługi są opisane jako kolekcje usług atomowych, z dodatkowymi ograniczeniami i określoną pojemnością, dostępnością i stanem.

3.2. Przyjmowanie zgłoszeń

Równie ważne jak efektywne operowanie zbiorem oferowanych usług jest precyzyjne opisanie potrzeb, tj. procesów zbudowanych z powiązanych i/lub równolegle występujących usług atomowych, z określeniem także potrzeb ilościowych zgłoszonych zdarzeń.

Opis jest budowany na podstawie bazy wzorców sieci usług – wzorców procesów obsługi. Baza ta zawiera wszystkie ważne i przeanalizowane przypadki procesów generycznych. Konkretna potrzeba, zgłoszona na stanowisku przyjmowania zgłoszeń, jest charakteryzowana przez dopasowanie do niej procesu generycznego, a także specyficzne cechy ilościowe i lokalizację. W razie braku wzorca, musi on zostać stworzony przez operatora centrum operacyjnego, przyjmującego zgłoszenie (rys. 2). Przewidujemy użycie w tym celu narzędzia do planowania działań.



Rys. 2. Architektura systemu

3.3. Środowisko agentowe

W naszej koncepcji jednym z głównych elementów systemu jest środowisko agentowe (rys. 2). Funkcje agentów obejmują:

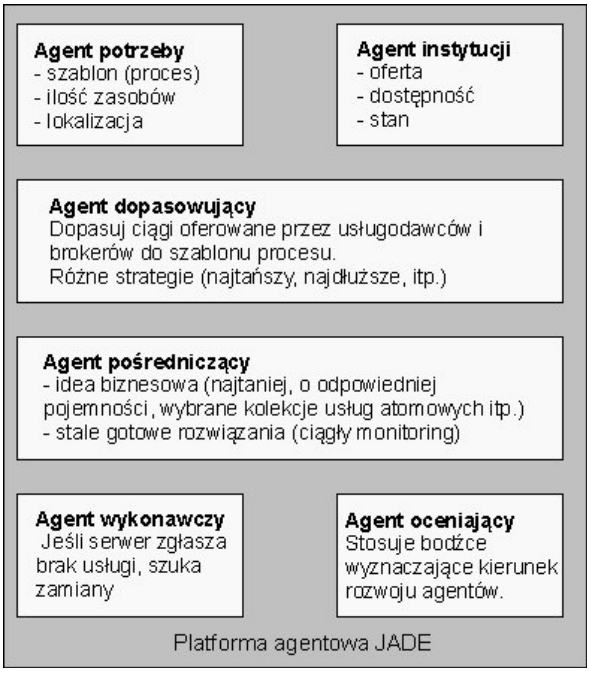
- Konstrukcję worflow usług – w tym celu mogą być wykorzystane różnego rodzaju algorytmy inteligentnego planowania.
- Wykonanie workflow usług, zarządzanie przebiegiem ich wykonania oraz zbieranie informacji na temat przebiegu wykonania.
- Generację stworzonych workflow w postaci BPEL oraz ich udostępnianie jako usług złożonych. W tym celu konieczne jest zapisanie złożonej usługi w postaci BPEL. Taka usługa powinna zostać opisana przy pomocy języka OWL-S. Po stworzeniu i udostępnieniu takiej usługi powinna ona zostać zapisana w repozytorium.

– Zarządzanie brokerami, uzupełnianie ich wiedzy. Agenci sterują zapytaniami do repozytoriów oraz zbierają informację odpytując usługi i gromadząc informację o ich wykonaniu.
– Monitorowanie przebiegu rezerwacji zasobów, doboru usług oraz wykonania skomponowanego łańcucha usług. W przypadku problemów, agenci powinni dokonać przebudowy łańcucha.

Podczas projektowania agentów należy wyspecyfikować:

– Podstawowe stosowane modele i architektury. Podstawowy sposób działania agentów powinien być oparty na schemacie agenta reaktywnego wykorzystującego protokół *Contract Net* [8], a architektura reaktywnego agenta, np. może być wersją warstwowej architektury podporządkowania (*subsumption architecture*). Podejście to oparte jest na wykonywaniu akcji związanych ze spełnionymi warunkami, czyli reguł mających składnię *JEŚLI warunek TO akcja.* Wykonywane są reguły, których warunki zajścia są spełnione w kolejności według przydzielonych priorytetów (zaklasyfikowania do warstw). Działając według protokołu *Contract Net* [8] albo też *Extended Contract Net* z podwójnym zatwierdzaniem, agent rozsyła ofertę do innych agentów, a następnie przyjmuje od nich ich propozycje jej realizacji, wybierając ofertę.
– Elementy modelu i algorytmy niezależne od stosowanej dziedziny obejmują: szablon działania architektury podporządkowania oraz obsługi reguł, podstawową komunikację między agentami (język FIPA ACL – *Agent Communication Language*), obsługę protokołu *Contract Net* (w wersji z pojedynczym i podwójnym potwierdzeniem), podstawową architekturę agenta – brokera oraz algorytmy jego działania.
– Elementy modelu i algorytmy zależne od stosowanej dziedziny (np. ratownictwo, łańcuchy dostaw) zawierają indywidualne reguły działania agenta, potem uzupełnione o dodatkowe elementy wynikające ze stosowania bardziej rozbudowanych architektur.

Działając według powyższych założeń, określiliśmy następujące elementy środowiska agentowego (rys. 3). *Agent potrzeby* jest powoływany przez nadejście zlecenia do obsługi i konfigurowany przez parametry związane z tym zadaniem. Działanie agenta jest opisywane przez szablon uzupełniony adresami zakontraktowanych serwisów zapisywany w WS-BPEL 2.0 i ładowany do serwera aplikacji. *Agent instytucji* prezentuje ofertę dostępnych usług, podając ich opis i charakterystykę np. dostępność, stan itp. *Agent dopasowujący* (*matchmaker*) ma za zadanie skojarzyć potrzeby wynikające ze zlecenia z ofertami instytucji oferującej usługi. Ponadto występują także *agenci pośredniczący* – brokerzy, którzy specjalizują się w wybranych usługach, oferując najlepsze rozwiązania, stale monitorując lub subskrybując komunikaty o stanach i parametrach konkretnych usługodawców. *Agent wykonawczy* (*invoker*) obserwuje komunikaty serwera, uruchamiając w miarę konieczności opracowanie i podmianę pliku sterującego. Wypełnianie szablonów realizują agenci dopasowujący (matchmakery). Zachowanie agentów dopasowujacych i pośredniczących powinno odzwierciedlać pożądane cechy i cele rozwoju środowiska agentowego. Dlatego utworzony jest *Agent oceniający*, który waliduje i nagradza działania agentów.



Rys. 3. Komponenty architektury środowiska agentowego

4. Realizacja

W celu sprawdzenia realizowalności przedstawionej koncepcji oraz przetestowania użyteczności dostępnych narzędzi opracowane zostały pilotowe realizacje dedykowane badaniu poszczególnych aspektów systemu. Związane są one z przygotowaniem opisu i wstępnych realizacji usług związanych ze zintegrowaną akcją ratownicza, wykonaniem usług złożonych opisanych w BPEL, dopasowywaniem usług, kompozycją usług przy użyciu automatycznych narzędzi planujących oraz monitorowaniem i korygowaniem przebiegu wykonania złożonego ciągu usług.

5. Podsumowanie

W ramach prac przygotowano wstępną koncepcję systemu łączącego podejścia agentowe oraz technologię SOA używającą usług webowych, przeznaczonego do prowadzenia zintegrowanej akcji ratowniczej oraz przygotowano przedstawione cząstkowe projekty prototypowe. Przewiduje się kontynuacje prac w zakresie rozbudowy i doprecyzowania opisu koncepcji systemu oraz stworzenie wstępnego prototyp całości systemu, integrującego wykonane elementy.

Literatura

[1] Dean M., Schreiber G., Bechhofer S., van Harmelen F., Hendler J., Ian Horrocks I., McGuinness D., Patel-Schneider P., Stein L., *OWL Web Ontology Language Reference*. W3C Recommendation, 10 February 2004. Latest version is available at http://www.w3.org/TR/owl-ref/.

[2] Jaeger M., Rojec-Goldmann G., Liebetruth Ch., Mühl G., Geihs K., *Ranked Matching for Service Descriptions Using OWL-S*. KiVS 2005, 91–102.

[3] Klusch M., Fries B., Sycara K., *Automated Semantic Web Service Discovery with OWLS-MX*. Proceedings of 5th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Hakodate, Japan, ACM Press 2006.

[4] OASIS Standard, *Web Services Business Process Execution Language Version 2.0.* http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf 11 April 2007.

[5] The OWLS-MX project web page, http://www.dfki.de/~klusch/owls-mx.

[6] OWL-S: Semantic Markup for Web Services http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/, W3C Member Submission 22 November 2004.

[7] OWL-S UDDI Matchmaker, http://www.semwebcentral.org/projects/owl-s-uddi-mm/.

[8] Smith R., *The contract net protocol: high-level communication and control in a distributed problem solver*. IEEE Transactions on Computer, December 1980.

[9] The Protege Ontology Editor and Knowledge Acquisition System, http://protege.stanford.edu

[10] W3C Recommendation, *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language.* http://www.w3.org/TR/wsdl20/ 26 June 2007.