Platforma hybrydowej kompozycji, gruntowania i wykonania usług sieciowych w środowisku SOA

L. BELAVA
belava123@gmail.com

Katedra Informatyki, Wydział IEiT, AGH w Krakowie

W artykule omówiono, czym jest hybrydowe tworzenie i gruntowanie planów kompozycji usług, cele ich powstania i ewentualnego zastosowania. Przedstawiono architekturę platformy programowej, omówiono jej poszczególne części. Przedstawiono zaimplementowane w ramach badań oprogramowanie oraz uzyskane wyniki.

Słowa kluczowe: SOA, Web Services, kompozycja usług, gruntowanie usług.

1. Wprowadzenie

Współczesny rozwój metod przetwarzania informacji coraz bardziej zmierza w kierunku systemów sieciowych. Dzięki temu paradygmat SOA (ang. *Service Oriented Architecture*) oraz jego szczególna realizacja w postaci serwisów webowych (ang. *Web Services*) naturalnie pasują do wytwarzania systemów informatycznych będących nie tylko rozproszonymi, ale także dynamicznymi z punktu widzenia swojej struktury wewnętrznej. Ważnymi cechami takich systemów informatycznych są: kompozycja usług podstawowych w ramach tworzenia serwisów złożonych, gruntowanie (ang. *grounding*) oraz wykonanie (ang. *execution*) planów kompozycji. Z kolei kompozycja hybrydowa usług sieciowych jest podejściem pozwalającym łączyć ze sobą różne metody tworzenia kompozycji. Dzięki zastosowaniu tego podejścia proces tworzenia kompozycji usług staje się bardziej elastyczny. W ramach tworzenia jednego planu kompozycji pojawia się możliwość użycia różnych metod dla różnych jego części. W analogiczny sposób wygląda koncepcja hybrydowego gruntowania planów kompozycji, która pozwala na stosowanie różnych metod gruntowania w ramach jednego planu. Niniejszy artykuł omawia te najważniejsze cechy oraz przedstawia realizującą ją platformę programową.

1.1. Kompozycja i gruntowanie serwisów webowych w ramach paradygmatu SOA

SOA – paradygmat architektoniczny opisujący cechy i właściwości systemów informatycznych, budowanych dzięki łączeniu ze sobą usług. W ramach tego paradygmatu usługi są relatywnie niezależnymi od siebie bytami (niekoniecznie sieciowymi) oferującymi różne funkcjonalności. Takie podejście przynosi następujące korzyści: łatwe dodawanie kolejnych usług do istniejącej infrastruktury programowej, łatwa integracja starego oprogramowania z nowymi procesami biznesowymi, możliwość bezpiecznej wymiany i ulepszenia poszczególnych składowych systemu informatycznego, gdyż są słabo powiązane ze sobą [1].

Elementem kluczowym paradygmatu SOA jest serwis (usługa). Model referencyjny SOA określa usługę jako byt informatyczny powołany dla organizacji dostępu do jednej lub więcej możliwości (ang. *capabilities*) [2]. Jednocześnie dostęp ten musi być zorganizowany nie w sposób dowolny, lecz za pomocą ustalonego interfejsu. Trzeba wspomnieć, iż paradygmat SOA jest koncepcją bardzo ogólną i ściśle nie definiuje sposobu własnej realizacji. Obecnie najbardziej popularną realizacją SOA stały się zatem wspomniane wcześniej serwisy webowe. Definicja usługi sieciowej przyjęta przez grupę roboczą W3C podaje, że jest to komponent programowy niezależny od platformy i implementacji, dostarczający określonej funkcjonalności oraz osiągalny za pomocą sieci komputerowej poprzez standardowe protokoły komunikacyjne [3].

Łączenie serwisów ze sobą w ramach jednego ciągu przetwarzania danych nazywamy kompozycją. Tworzenie takich kompozycji pozwala w miarę prosty sposób budować bardzo złożone usługi, korzystając z funkcjonalności poszczególnych usług składowych.

Obecnie zostało opracowanych wiele różnych metod tworzenia kompozycji usług sieciowych. Metody ręczne oraz półautomatyczne wymagają bezpośredniego uczestnictwa użytkownika i są dosyć popularne w świecie nauki. Przykładami takich prac są między innymi [4, 5]. Różnego rodzaju łączenie metod w łańcuch (ang. *chaining*) w przód (ang. *forward chaining*) lub w tył (ang. *backward chaining*) zostało opisane w artykułach [6, 7] itd. Podejścia używające HTN, czyli hierarchicznych sieci zadań (ang. *Hierarchical Task Networks*), są zaproponowane w takich pracach, jak [8, 9]. Opisy ontologiczne usług sieciowych pozwalają na użycie silników wnioskujących dla tworzenia kompozycji [10]. Sieci Petriego zostały użyte do modelowania i komponowania usług w pracy [11].

Metody tworzenia kompozycji usług nie zawsze zakładają fakt istnienia pewnych instancji usług gdziekolwiek w sieci. W takim przypadku możemy mówić o tym, iż plan kompozycji jest abstrakcyjny i wymaga gruntowania (ang. *grounding*) przed wykonaniem. Gruntowanie jest procesem, dzięki któremu plan abstrakcyjny zostaje uzupełniony o informacje niezbędne dla znalezienia w sieci odpowiednich instancji usług. Jako przykłady różnych podejść do tematu gruntowania warto wspomnieć: stosowanie brokerów [12], dopasowywanie (ang. *matching*) [13], metody heurystyczne [14], metody agentowe [15], metody ontologiczne [16, 17].

1.2. Sformułowanie problemu

Jak pokazano w rozdziale 1.1, istnieje wiele zróżnicowanych metod tworzenia planów kompozycji usług oraz ich gruntowania. Jednak nie zawsze jedno szczególne podejście jest najlepszym ze wszystkich punktów widzenia. Wyobraźmy sobie sytuację, gdy użytkownik systemu informatycznego opartego na SOA z góry zna pewne części przyszłego planu (jeszcze przed rozpoczęciem procesu tworzenia kompozycji usług). Chciałby tylko dobudować pozostałe elementy w sposób automatyczny. W odpowiedzi na ten problem powstała hybrydowa kompozycja usług [18]. Pozwala ona w ramach budowy jednego planu łączyć ze sobą różne metody tworzenia kompozycji.

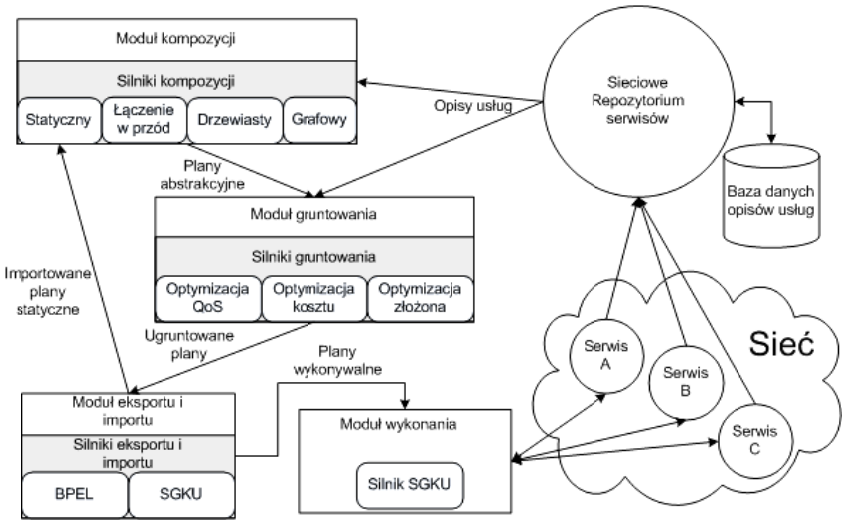
Podobny problem dotyczy również wyboru odpowiedniej metody gruntującej abstrakcyjne plany kompozycji, gdyż mogą one zasadniczo różnić się nie tylko sposobem działania, ale również celami optymalizacji (QoS, koszt itd.). Problem ten stara się rozwiązać koncepcja hybrydowego gruntowania abstrakcyjnych planów kompozycji usług. Opisuje ona pewien mechanizm, dzięki któremu można łączyć ze sobą różne metody gruntowania w ramach procesu gruntowania pojedynczego planu.

Z kolei w literaturze naukowej przedstawiono wiele platform kompozycji usług. Najważniejsze z nich to: SWORD [19], METEOR-S [20], MAESTRO [21], SPICE [22]. Niestety, żadne z tych podejść nie stara się rozwiązać problemu możliwości dogodnego wyboru metod tworzenia oraz gruntowania planów kompozycji usług sieciowych. SWORD używa logiki pierwszego rzędu, METEOR-S stosuje silnik CSP (ang. *Constraint Satisfaction Problem*) dla tworzenia kompozycji, MAESTRO opiera się na metodzie grafowej z łączeniem wstecz, SPICE również stosuje łączenie wstecz z rozgałęzieniami w celach optymalizacyjnych.

Jak widać, opracowane w literaturze naukowej podejścia do metod oraz platform kompozycji i gruntowania serwisów webowych są bardzo zróżnicowane. Żadne z nich nie jest idealne pod każdym względem jednocześnie (na przykład języki tworzonych planów kompozycji lub też cele optymalizacji gruntowania planów abstrakcyjnych). W odpowiedzi na te problemy powstała platforma hybrydowej kompozycji, gruntowania i wykonania usług sieciowych. Użyte w niej koncepcje hybrydowej kompozycji oraz hybrydowego gruntowania planów kompozycji usług sieciowych pozwalają zaadresować te problemy.

2. Przedstawiana platforma w ramach paradygmatu SOA

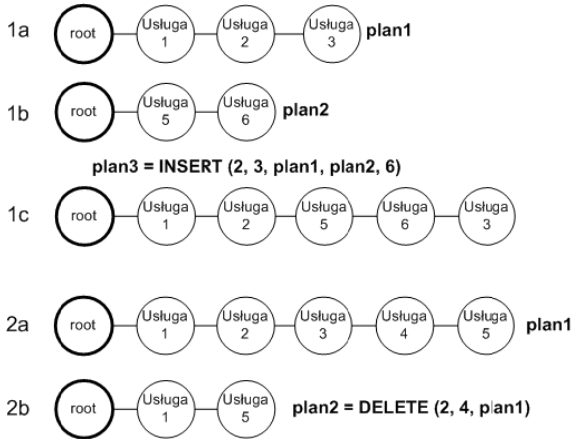
Architektura platformy hybrydowej kompozycji, gruntowania oraz wykonania usług sieciowych w ramach paradygmatu SOA składa się z pięciu kluczowych modułów współpracujących ze sobą oraz elementów zewnętrznych względem systemu – usług sieciowych. Na rysunku 1 przedstawiono koncepcję platformy oraz zaznaczono kierunki przepływu najważniejszych danych między jej częściami składowymi.



Rys. 1. Koncepcja platformy hybrydowej kompozycji, gruntowania oraz wykonania usług sieciowych

Moduł kompozycji jest elementem platformy odpowiadającym za hybrydowe komponowanie usług sieciowych. Współpracuje z repozytorium serwisów, od którego otrzymuje opisy usług. Współpracuje również z modułem eksportu i importu, dzięki czemu silnik statyczny otrzymuje plany kompozycji zapisane w różnych językach modelowania serwisów. Utworzone kompozycje planów są przekazywane dalej modułowi gruntowania. Część składową modułu stanowią różne silniki tworzenia planów kompozycji usług.

1. Silnik kompozycji statycznej pozwala składać kompozycje z gotowych planów, które mogą być wczytane dzięki modułowi eksportu i importu lub skomponowane za pomocą innych silników. Operowanie na planach polega na operacjach wycinania (DELETE) i wstawiania (INSERT) całych planów lub ich części. Na rysunku 2 przedstawiono dwa przykłady schematów wywołania tego silnika.



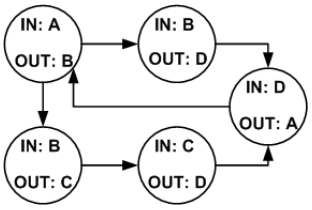
Rys. 2. Przykłady wywołania silnika kompozycji statycznej

Operacja wstawiania jednego planu do środka drugiego (INSERT) jest pokazana za pomocą planów 1a, 1b oraz 1c reprezentujących dwa plany wejściowe (plan1 i plan2) oraz plan wynikowy – plan3. Plan3 powstaje dzięki wywołaniu INSERT (2, 3, plan1, plan2, 6), którego parametry oznaczają wstawianie plan2 od elementu początkowego po usługę o numerze 6 do planu plan1 między usługami o numerach 2 i 3. Operacja wycinania części planu (DELETE) jest pokazana za pomocą planów 2a oraz 2b reprezentujących plan wejściowy plan1 oraz plan wyjściowy plan2. Plan2 powstaje dzięki wywołaniu DELETE (2, 4, plan2), którego parametry oznaczają wycięcie z planu plan1 ciągu wywołań usług poczynając od usługi o numerze 2 po usługę o numerze 4 włącznie.

2. Silnik łączenia w przód realizuje algorytm tworzenia planów kompozycji za pomocą metody łączenia usług w przód. Uproszczony schemat działania takich metod polega na sukcesywnym dołączaniu na koniec istniejącego planu kolejnych elementów typu wejście, które pasują do typu wyjścia bieżącej usługi będącej ostatnią na ścieżce wywołań. Celem jest stworzenie takiego planu, którego końcowy element będzie miał pożądany typ wyjścia.
3. Silnik drzewiasty realizuje drzewiasty algorytm tworzenia planów kompozycji usług. Uproszczony schemat działania takich metod polega na tworzeniu drzew rozwiązań na zasadach podobnych do metod łączenia w przód lub w tył. Drzewa te później są przeszukiwane, dzięki czemu

również powstaje możliwość optymalizacji pożądanych parametrów końcowego planu.

4. Silnik grafowy tworzy kompozycje usług w oparciu o algorytmy poszukiwania ścieżek w grafie. Uproszczony schemat działania takich metod polega na stworzeniu grafu zależności dostępnych usług oraz jego analizie takimi narzędziami, jak algorytm Dijkstry, algorytm mrówkowy, algorytm przeszukiwania wszerz itd. Na rysunku 3 został przedstawiony przykładowy schemat skierowanego grafu zależności usług sieciowych.



Rys. 3. Przykładowy schemat grafu zależności usług sieciowych

Moduł gruntowania – pozwala gruntować abstrakcyjne plany kompozycji usług sieciowych dostarczanych modułem kompozycji. Współpracuje również z repozytorium serwisów, od którego dowiaduje się o dostępnych usługach istniejących w sieci oraz ich parametrach, takich jak koszt, QoS, typy danych wejściowych i wejściowych. Część składową modułu stanowią różne silniki gruntowania:

1. Silnik optymalizacji QoS – zadaniem tego silnika jest optymalizacja planów lub ich części pod względem zadanych parametrów QoS. Przykładowo możemy powiedzieć, że wartość QoS powinna znajdować się w pewnym przedziale. Wtedy algorytm wśród równoważnych usług pod względem wejść lub wyjść będzie szukał takich, które spełniają podane kryterium.
2. Silnik optymalizacji kosztów – podobny do silnika QoS, lecz za zadanie ma optymalizować koszt planu kompozycji usług lub jego części.
3. Silnik optymalizacji złożonej – pozwala na zdefiniowanie hierarchii preferencji gruntowania, która umożliwia stosowanie dodatkowych optymalizacji w przypadkach, kiedy silnik wyższego poziomu znajdzie kilka usług równoważnych. Jako przykład możemy wyobrazić sobie taką sytuację, w której najważniejszym parametrem podczas gruntowania jest koszt usługi. Jeśli jednak zajdzie sytuacja niejednoznaczna, to chcielibyśmy wybrać taką usługę wśród alternatywnych, która zapewni najlepszą jakość.

Moduł eksportu i importu – pozwala importować oraz eksportować plany kompozycji usług (zarówno abstrakcyjne, jak i nadające się do wykonania). Zrealizowane w ramach modułu silniki to odpowiednio:

1. Silnik eksportu i importu BPEL – pozwala importować oraz eksportować plany kompozycji usług zdefiniowane w języku BPEL [23, 24]. Obecnie nie realizuje pełnej funkcjonalności języka BPEL. Jednakże takie zasadnicze elementy modelu, jak instrukcje warunkowe, pętle oraz równoległe wykonanie kilku ścieżek przetwarzania zostały uwzględnione.
2. Silnik eksportu i importu SGKU – pozwala importować oraz eksportować plany kompozycji opisane jako Skierowane Grafy Kompozycji Usług [24]. SGKU jest modelem reprezentacji kompozycji usług sieciowych opartym na grafach skierowanych.

Moduł wykonania – wykonuje ugruntowane plany kompozycji usług. Obecnie moduł zawiera tylko silnik wykonania SGKU. Koncepcja jednak zakłada możliwość dołączenia innych silników. Dlatego w ramach modułu eksportu należy zrealizować odpowiedni silnik eksportu ugruntowanych planów, a w ramach modułu wykonania zrealizować silnik je wykonujący.

Repozytorium serwisów – serwis webowy, którego zadaniem jest zbieranie, przechowywanie i dostarczanie informacji o usługach znajdujących się w sieci. Współpracuje z modułami kompozycji oraz gruntowania. Przechowuje informacje o usługach w bazie danych opisów usług.

3. Przykład zastosowania w praktyce

Jako przykład zastosowania platformy w praktyce możemy wyobrazić sobie system elektronicznego wyszukiwania oraz zamawiania towarów na kredyt. Użytkownik posługując się systemem informatycznym, po podaniu swoich danych osobowych oraz innych informacji o charakterze finansowym, mógłby wprowadzić pewne kryteria wyszukania towaru i zamówić go na kredyt – o ile zostanie zaliczony do osób wiarygodnych do przyznania kredytu.

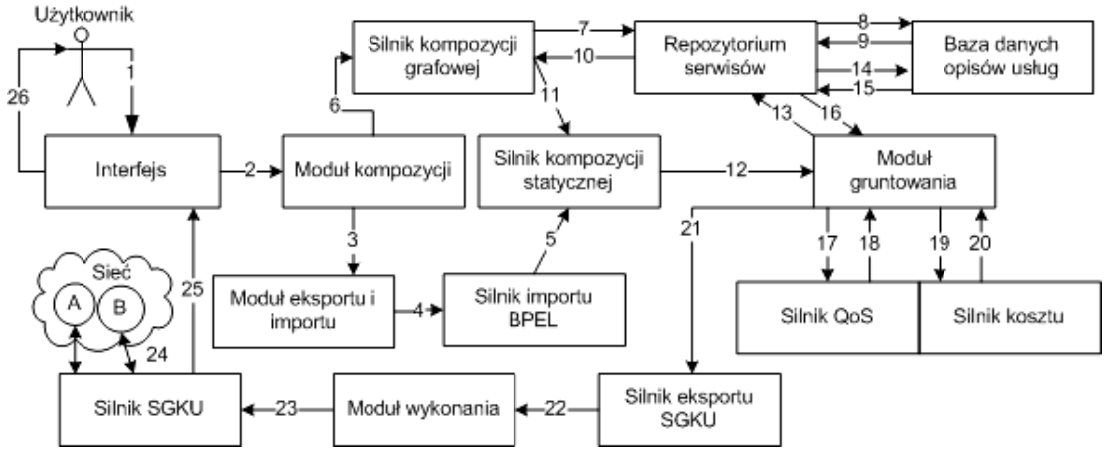
W przypadku systemów takiego typu należy powiedzieć, iż pewne części docelowego planu kompozycji są określone z góry poprzez wymagania prawne lub branżowe standardy. Dane osobowe, dane finansowe, dane związane

z procesowaniem kart kredytowych – wszystkie podlegają różnorodnym regulacjom i muszą być traktowane szczególnie ostrożnie. Z tego powodu hybrydowa metoda tworzenia planów kompozycji usług jest jak najbardziej przydatna, ponieważ pozwala między innymi łączyć gotowe części planów kompozycji z wynikami pracy metod automatycznych lub półautomatycznych.

3.1. Przykładowy przebieg pracy systemu informatycznego od kompozycji do wykonania

Na rysunku 4 został przedstawiony przebieg tworzenia, gruntowania oraz wykonania kompozycji w systemie informatycznym realizującym koncepcję hybrydowej kompozycji i gruntowania usług.

1. Użytkownik za pomocą interfejsu podaje dane osobowe, informacje finansowe oraz określa, jakiego typu produkt i w jakiej ilości chce otrzymać.
2. Dane te są przekazywane modułowi kompozycji.
3. Moduł kompozycji przekazuje modułowi eksportu i importu zlecenie na zaimportowanie (z góry ustalonej standardem) części planu odpowiadającej za pracę z danymi osobowymi i informacjami finansowymi.
4. Moduł eksportu i importu inicjalizuje silnik importu BPEL.
5. Silnik importu BPEL przekazuje zaimportowaną część planu silnikowi kompozycji statycznej, aby była ona później złączona z częścią planu wygenerowaną automatycznie.
6. Moduł kompozycji inicjalizuje silnik kompozycji grafowej i przekazuje mu parametry kompozycji.
7. Silnik kompozycji grafowej odpytuje repozytorium serwisów o listę typów usług dostępnych w sieci.
8. Repozytorium serwisów wysyła do bazy danych zapytanie o dostępne typy (nie instancje) usług.
9. Baza danych przygotowuje wynik zapytania i wysyła do repozytorium.
10. Repozytorium zwraca listę dostępnych typów usług silnikowi kompozycji grafowej.
11. Silnik kompozycji grafowej tworzy abstrakcyjny plan kompozycji i przekazuje go silnikowi kompozycji statycznej.
12. Silnik kompozycji statycznej scala plan zaimportowany silnikiem importu BPEL z planem wygenerowanym silnikiem kompozycji grafowej i przekazuje go modułowi gruntowania.
13. Moduł gruntowania odpytuje repozytorium serwisów o listę usług rzeczywistych, których typy wejść oraz wyjść pasują do typów wejść i wyjść usług użytych w abstrakcyjnym planie kompozycji.
14. Repozytorium serwisów tworzy i wysyła zapytanie do bazy danych.
15. Baza danych przygotowuje odpowiedź i zwraca ją do repozytorium.
16. Repozytorium przekazuje modułowi gruntowania listę instancji usług istniejących w sieci.
17. Moduł gruntowania przekazuje odpowiednią część planu do silnika gruntowania QoS.
18. Silnik QoS gruntuje część planu i zwraca modułowi gruntowania.
19. Moduł gruntowania przekazuje pozostałą część planu silnikowi kosztu.
20. Silnik kosztu gruntuje swoją część planu i zwraca modułowi gruntowania.
21. Ugruntowany plan kompozycji zostaje przekazany do eksportu.
22. Wyeksportowany plan jest przekazywany modułowi wykonania.
23. Moduł wykonania inicjalizuje silnik wykonania planów kompozycji opisanych za pomocą SGKU.
24. Silnik SGKU wykonuje plan kompozycji usług.
25. Silnik SGKU przekazuje interfejsowi użytkownika otrzymane wyniki pracy.
26. Interfejs użytkownika przedstawia w sposób tekstowo-graficzny wyniki pracy systemu.



Rys. 4. Diagram ilustrujący proces kompozycji, gruntowania i wykonania planu kompozycji

3.2. Badanie systemu

Na rysunku 5 została przedstawiona wizualizacja abstrakcyjnego planu kompozycji usług opisanego w języku BPEL.

Rys. 5. Abstrakcyjny plan kompozycji w języku BPEL zawierający ustalony ciąg wywołań usług

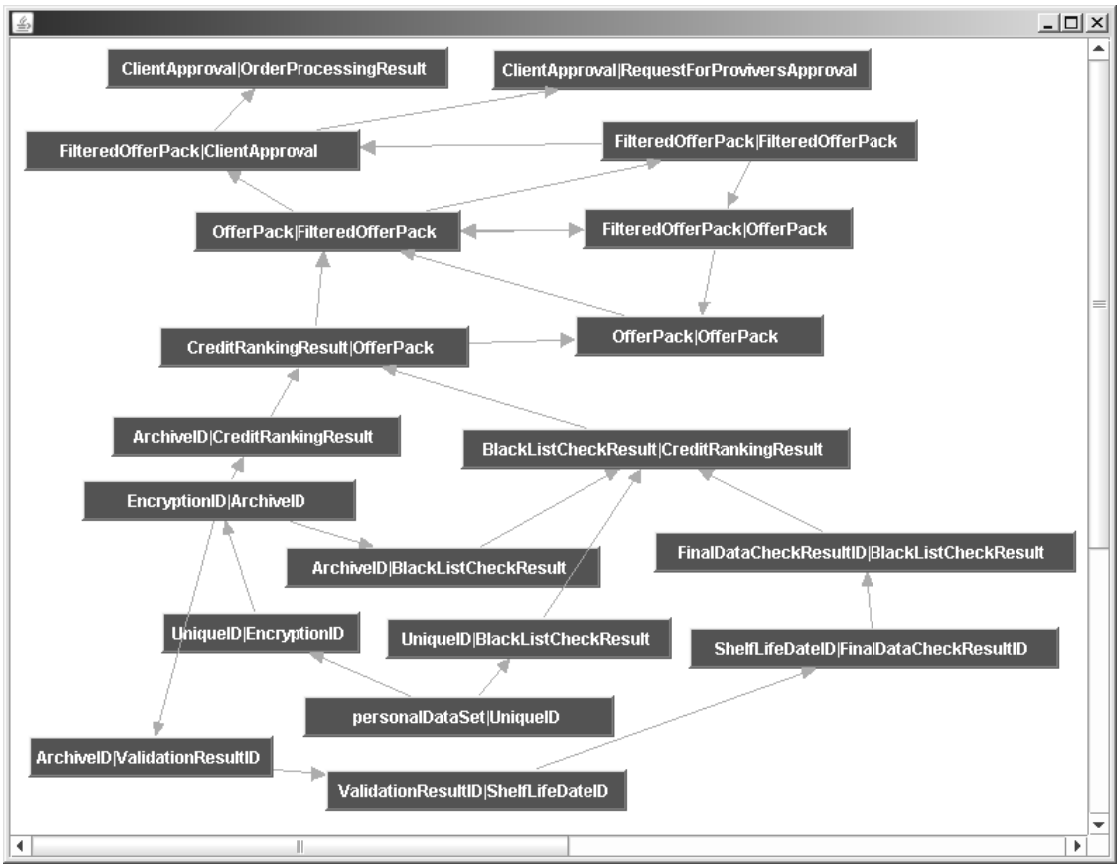
Proces ten składa się z ustalonego ciągu działań z danymi osobowymi oraz finansowymi. Wywołania te reprezentują następujące kroki:

- ”GetUniqueID” – nadaje otrzymanej paczce danych unikatowy identyfikator
- ”Encrypt” – szyfruje dane
- ”Archive” – archiwizuje zaszyfrowaną paczkę danych
- ”DataValidation” – waliduje dane, ”AssignShelfLiveDate” – przypisuje danym datę ważności
- ”FinalDataCheck” – końcowe sprawdzenie danych oraz działań przeprowadzonych w poprzednich krokach.

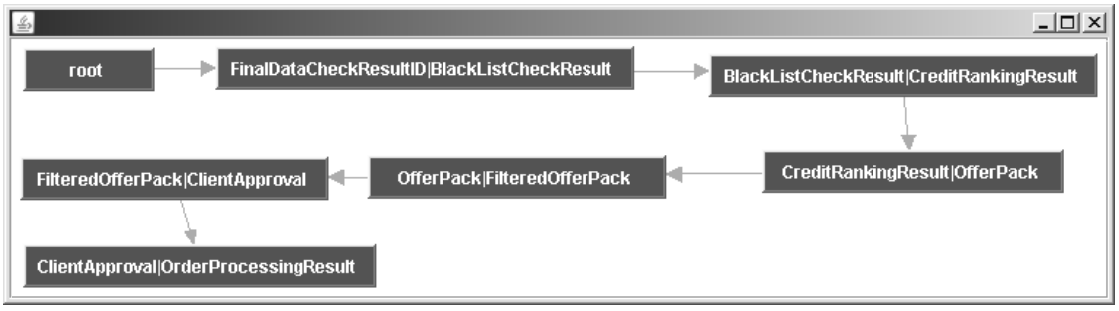
Automatyczne tworzenie drugiej części planu kompozycji, mającej na celu wyszukiwanie i zamówienie towarów na kredyt dla użytkownika, było wykonane za pomocą silnika grafowego. Na rysunku 6 przedstawiona jest wizualizacja pełnego grafu zależności wszystkich rodzajów usług zarejestrowanych w repozytorium. Poszczególne identyfikatory rodzajów usług abstrakcyjnych są nadawane automatycznie przez moduł repozytorium. Każdy z tych identyfikatorów składa się z nazwy typu wejścia usługi, znaku „|” oraz nazwy typu wyjścia usługi.

Na rysunku 7 została przedstawiona wizualizacja abstrakcyjnego planu kompozycji utworzonego w wyniku pracy silnika grafowego. Jako wymagany typ wejściowy do pożądanego planu, silnikowi został podany „FinalDataCheckResultID”, który jest typem wyjściowym końcowej usługi „FinalDataCheck” z ustalonego ciągu wywołań usług. Jako wymagany typ końcowy planu został podany typ „OrderProcessingResult”, czyli typ wyniku pracy usługi rzeczywiście zamawiającej towar. Poszczególne kroki tego planu to:

- ”FinalDataCheckResultID|BlackListCheck-Result” – sprawdzanie danych klienta systemu na czarnych listach finansowych
- ”BlackListCheckResult|CreditRanking-Result” – proces scoringu kredytowego, czyli obliczenie wiarygodności klienta na podstawie jego danych
- ”CreditRankingResult|OfferPack” – automatyczny dobór zbioru ofert dla klienta
- ”OfferPack|FilteredOfferPack” – automatyczne filtrowanie ofert do kilku najbardziej pasujących klientowi
- ”FilteredOfferPack|ClientApproval” – końcowy wybór oferty klienta i jej potwierdzenie do zamówienia
- ”ClientApproval|OrderProcessingResult” – rzeczywiste zamówienie klienta potwierdzonej oferty na kredyt.



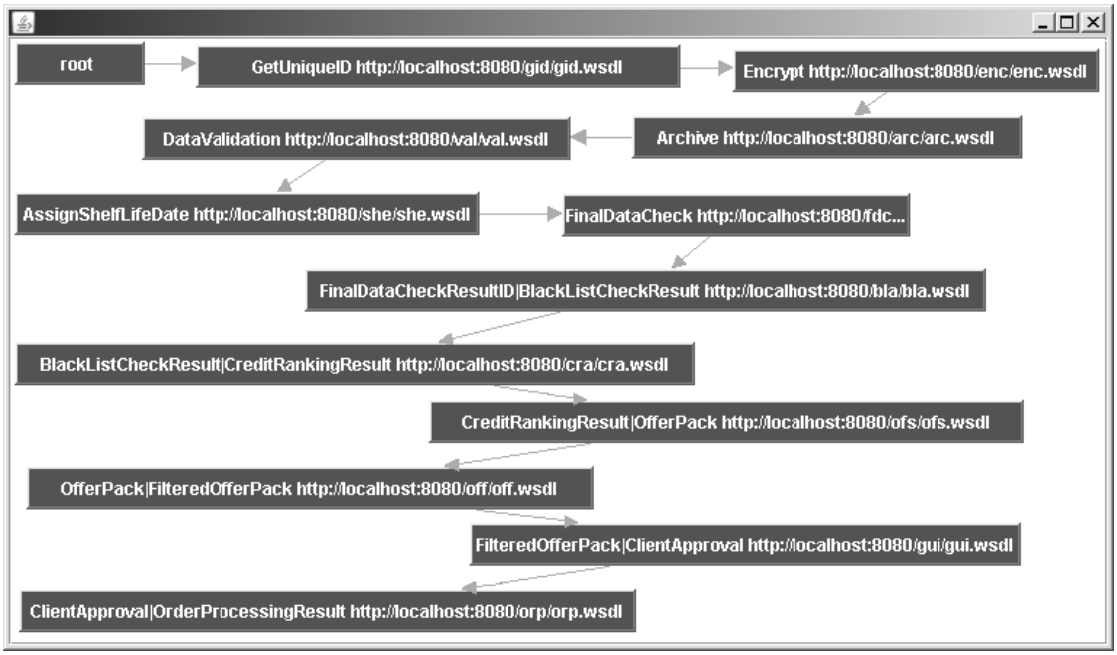
Rys. 6. Wizualizacja pełnego grafu zależności usług w repozytorium



Rys. 7. Wizualizacja automatycznie skomponowanego planu

Po otrzymaniu wyniku pracy silnika kompozycji grafowej został on złączony z zaimportowanym planem kompozycji zawierającym ustalony ciąg działań z danymi o charakterze osobowo-finansowym. Powstały w wyniku tej operacji kompletny plan abstrakcyjny został podzielony na dwie strefy gruntowania. Wszystkie operacje (oprócz wyszukiwania klienta na czarnej liście oraz nadania mu scoringu kredytowego) były gruntowane z optymalizacją kosztu usług. Z kolei te dwie usługi były gruntowane z optymalizacją QoS, ponieważ chcemy jak najlepiej zbadać naszych klientów, zanim przystąpimy do wyszukiwania i zamawiania. Wizualizacja końcowego ugruntowanego planu kompozycji usług jest pokazana na rysunku 8.

Różnica wizualizacji między planem ugruntowanym a abstrakcyjnym polega na tym, że każdej usłudze został przypisany adres sieciowy pliku WSDL (ang. *Web Service Description Language*) zawierający opis rzeczywistej instancji usługi istniejącej w sieci. Plik ten jest jednoznacznie skojarzony z odpowiednią usługą w nim opisaną i reprezentuje jej adres URL.



Rys. 8. Wizualizacja końcowego ugruntowanego, wykonywalnego planu kompozycji usług

Wykonanie planu końcowego odbywało się za pośrednictwem modułu wykonania, który wywoływał odpowiednie serwisy umieszczone w sieci, dostępne dzięki komunikacji używającej technologii SOAP, korzystając z protokołu HTTP.

4. Podsumowanie

Dziedzina automatyzacji tworzenia planów kompozycji usług staje się coraz ważniejsza w czasach popularyzacji rozwiązań chmurowych opierających się na architekturze SOA. Mimo dostępności coraz większej ilości usług, dotychczas proponowane podejścia często nie mogą w pełni zadowolić użytkowników. Oferują one zbyt mały poziom kontroli nad procesami komponowania i gruntowania planów kompozycji usług.

Proponowana platforma umożliwia komponowanie, gruntowanie i wykonywanie złożonych planów kompozycji za pomocą łączenia ze sobą różnych metod tworzenia i gruntowania planów. Dzięki temu użytkownicy systemu mogą osiągnąć znacznie większą kontrolę nad tymi procesami.

Można stwierdzić, iż proponowane rozwiązanie pozwala znacznie poszerzyć użyteczność metod tworzenia i gruntowania kompozycji usług.

5. Bibliografia

[1] K. Cetnarowicz, L. Belava, T. Dyduch, J. Koźlak, G. Wąchocki, M. Żabińska, *Przegląd metod komponowania usług webowych ze szczególnym uwzględnieniem podejścia agendowego*, Raport dla Instytutu Podstaw Informatyki PAN, 2009.

[2] C.M. MacKenzie, K. Laskey, F. McCabe, P. Brown, R. Metz, Reference Model for Service Oriented Architecture 1.0. http://www.oasisopen.org/committees/download.php/19679/soa-rm-cs.pdf 2006.

[3] L. Belava, ”Concept of Hybrid Service Composition in SOA Environment with Agent Enrichment”, in: *Proceedings of the 2010 Second Forum of Young Researcher*, 2010.

[4] E. Sirin, J. Hendler, B. Parsia, ”Semiautomatic Composition of Web Services using Semantic Descriptions”, in: *Proceedings of a Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS 2003*, 2003.

[5] E. Sirin, J. Hendler, B. Parsia, ”Filtering and Selecting Semantic Web Services with Interactive Composition Techniques”, *IEEE Intelligent Systems*, 19, 42 (2004).

[6] S. Thakkar, C. Knoblock, J. Ambite, C. Shahabi, ”Dynamically Composing Web Services from On-line Sources”, in: *Proceedings of the AAAI-2002 Workshop on Intelligent Service Integration*, 2002.

[7] M. Sheshagiri, M. DesJardins, T. Finin, ”A Planner for Composing Services

Described in DAML-S", in: *Proceedings of the AAMAS Workshop on Web Services and Agent-based Engineering*, 2003.
[8] E. Sirin, B. Parsia, D. Wu, J. Hendler, D. Nau, "HTN planning for Web Service composition using SHOP2", *Web Semantics: Science, Services and Agents Journal*, 4, 377 (2004).
[9] S. Sohrabi, J. Baier, S. McIlraith, "HTN Planning with Preferences", *Web Semantics: Science, Services and Agents Journal*, 4, 377 (2004).
[10] A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, "DAML-S:Web Service Description for the Semantic Web", in: *Proceedings of the International Semantic Web Conference* (*ISWC*) 2002.
[11] R. Hamadi, B. Benatallah, "Petri Net-based model for Web Service Composition", in: *Proceedings of the 14th Australasian Database Conference on Database Technologies*, 2003.
[12] D. Chakraborty, Y. Yesha, A. Joshi, "A Distributed Service Composition Protocol for Pervasive Environments", in: *Proceedings of the 2004 IEEE Wireless Communications and Networking Conference*, 2004.
[13] S. Sun, X. Tang, X. Yan, D. Chen, "A Symmetric Matchmaking Engine for Web Service Composition", in: *Proceedings of the 15th International Conference on Parallel and Distributed Systems*, 2009.
[14] D. Liu, Y. Shao, C. Yu, D. Chen, G. Fan, "A Heuristic QoS-Aware Service Selection Approach to Web Service Composition", in: *Proceedings of 8th IEEE/ACIS International Conference on Computer and Information Science*, 2009.
[15] J. Tang, X. Xu, "An Adaptive Model of Service Composition Based on Policy Driven and Multi-Agent Negotiation", in: *Proceedings of the 5th International Conference on Machine Learning and Cybernetics*, 2006.
[16] H. Yan, W. Zhijian, L. Guiming, "A Novel Semantic Web Service Composition Algorithm Based on QoS Ontology", in: *Proceedings of the 2010 International Conference on Computer and Communication Technologies in Agriculture Engineering*.
[17] S. Bleul, T. Weise, "An Ontology for Quality-Aware Service Discovery", in: *Proceedings of the First International Workshop on Engineering Service Compositions*, 2005.
[18] L. Belava, „Koncepcja hybrydowej kompozycji usług w środowisku SOA", *Automatyka*, 13/2, 189–197 (2009).
[19] S. Ponnekanti, A. Fox, "SWORD: A Developer Toolkit for Web Service Composition", in: *Proceedings of the 11th International WWW Conference*, 2002.
[20] R. Aggarwal, "Constraint driven Web service composition in METEOR-S", in: *Proceedings of IEEE International Conference on Services Computing*, 2004.
[21] V. Chifu, I. Salomie, A. Riger, V. Radoi, "A Graph Based Backward Chaining Method for Web Service Composition", in: *Proceedings of IEEE 5th International Conference on Intelligent Computer Communication and Processing*, 2009.
[22] E. Silva, L.F. Pires, M. Sinderen, "An Algorithm for Automatic Service Composition", in: *Proceedings of 1st International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing*, 2007.
[23] L. Belava, „Algorytm konwersji skierowanego grafu kompozycji serwisów do planów kompozycji serwisów webowych w języku BPEL", *Automatyka*, 15/2, 71–80 (2011).
[24] L. Belava, "Transforming BPEL service composition into a service composition directed graph for better composition plan management", in: *Proceedings of 25th European Conference on Modelling and Simulation*, 2011.

Web services hybrid composition, grouping and execution platform

L. Belava

The paper describes a software platform that implements the concept of hybrid composition, grounding and execution. The described platform allows to use different methods to build, ground and execute service composition plans.

Keywords: SOA, Web Services, service composition, web services grounding.