Lev Belava*

Algorytm konwersji skierowanego grafu kompozycji serwisów do planów kompozycji serwisów webowych w języku BPEL

1. Wprowadzenie

Rozwój SOA (*Service Oriented Architecture*) oraz serwisów webowych (*Web Services*) skutkowały powstaniem różnych notacji i języków formalnych służących potrzebom modelowania oraz egzekucji procesów kompozycji. Jednym z najbardziej popularnych języków w tej dziedzinie jest BPEL (*Business Process Execution Language*). Jego infrastruktura oferuje duży wybór oprogramowania dla edycji, wizualizacji oraz egzekucji planów kompozycji serwisów webowych. Jednak automatyczna edycja planów nie została w pełni opracowana. Z tego powodu na potrzeby realizacji koncepcji Hybrydowej Kompozycji Usług [1] został opracowany Skierowany Graf Kompozycji Serwisów oraz algorytm jego konwersji do planów kompozycji w języku BPEL. Niniejszy artykuł przedstawia taki graf i sposób, w jaki może być on skonwertowany do planu kompozycji serwisów w języku BPEL, a także oprogramowanie implementujące algorytm konwersji oraz uzyskane z jego pomocą przykładowe wyniki.

1.1. Kompozycja serwisów webowych w ramach paradygmatu SOA

SOA jest paradygmatem opisującym cechy i właściwości, jakie powinien mieć system informatyczny, i przedstawia pogląd na tworzenie oprogramowania składającego się z wielu relatywnie niezależnych od siebie komponentów zwanych usługami lub serwisami. Przynosi to następujące korzyści: łatwe dodawanie kolejnych usług do istniejącej infrastruktury, łatwa integracja i wchłonięcie starego oprogramowania w nowe procesy biznesowe oraz duże możliwości bezpiecznej wymiany i ulepszenia poszczególnych słabo związanych między sobą części składowych [2].

* AGH Akademia Górniczo-Hutnicza, Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki, Katedra Informatyki, doktorant

Model referencyjny SOA określa serwis jako byt informatyczny powołany dla organizacji dostępu do jednej lub więcej możliwości (*capabilities*) [3], przy czym dostęp ten musi być zorganizowany nie w sposób dowolny, ale za pomocą ustalonego interfejsu.

Obecnie najbardziej popularną realizacją SOA jest podejście związane z użyciem serwisów webowych. Definicja serwisu przyjęta przez grupę roboczą W3C podaje, że jest to komponent programowy niezależny od platformy i implementacji, dostarczający określonej funkcjonalności i osiągalny za pomocą sieci komputerowej poprzez standardowe protokoły komunikacyjne [4].

Łączenie kilku serwisów ze sobą nazywa się kompozycją. Kompozycja serwisów pozwala w prosty sposób tworzyć i modelować różne problemy, procesy biznesowe lub ich części składowe. Powstające plany kompozycji serwisów oraz serwisy złożone są używane podczas budowy oprogramowania w ramach koncepcji SOA.

Istnieje kilka podejść do problemu kompozycji. Język WS-CDL (*Web Services Choreography Description Language*) jest wspomniany standardem W3C [5] i jest przystosowany do tego, aby łączyć orkiestrowanie w BPEL z choreografią. Podejście semantyczne używa opisy serwisów w językach ontologicznych. Popularnym w tej dziedzinie jest język OWL (*Web Ontology Language*) [6] oraz ontologia serwisów OWL-S (*OWL – Semantic Markup for Web Services*) [7, 8]. Algebraiczna kompozycja porównuje typy wejść serwisów z typami wyjść. Podejście [9] skupia się na modelowaniu i komponowaniu serwisów jako sieci Petriego. Podejście HTN (*hierarchical task networks*) rekursywnie dekomponuje zadania, aż do chwili otrzymania operacji podstawowych, które mogą być wykonane serwisami. W podejściu [10] zaproponowano translację z OWL-S do domeny SHOP2 (*Simple Hierarchical Ordered Planner*), dzięki czemu można traktować problem kompozycji jako problem planowania. Hybrydowa kompozycja usług [1] skupia się na łączeniu powyżej wymienionych typów komponowania serwisów złożonych.

1.2. Prace powiązane i sformułowanie problemu

Zaproponowano kilka metod konwersji różnych struktur danych opisujących kompozycje serwisów do planów w języku BPEL. W [11] pokazano metodę translacji specyficznych modeli kompozycji w UML (*Unified Modeling Language*) do kodu BPEL. [12] omawia podejście semantyczne opierające się na generowaniu kompozycji serwisów BPEL za pomocą silnika ontologicznego pracującego na ontologii OWL. W [13] pokazano podejście translacji do BPEL automatów czasowych, reprezentujących kompozycje serwisów. Autorzy [14] opisali podejście automatycznej generacji kodu kompozycji w BPEL oraz opisów serwisów w WSDL na podstawie modeli specjalistycznych automatów skończonych AFSM (*Annotated Finite-State Machine*). Praca [15] prezentuje algorytm BPELGEN generujący plany w języku BPEL na podstawie kompozycji opisanych za pomocą odpowiednio przygotowanych skierowanych acyklicznych grafów. W [16] autorzy opisali podejście translacji nieco innych kierowanych grafów do kompozycji serwisów w języku BPEL.

Opracowane podejścia do kompozycji serwisów webowych są bardzo zróżnicowane, ale żadne z nich nie jest idealne pod wszystkimi względami, na przykład pod takimi jak język produkowanych planów kompozycji lub problematyka gruntowania planów abstrakcyjnych. Dla tego w systemach SOA powstają problemy ręcznej i automatycznej edycji oraz konwersji planów. Z drugiej strony obecnie język BPEL jest najbardziej popularnym językiem zapisu planów kompozycji serwisów webowych dzięki temu, że korzysta z XML i jego infrastruktura oferuje wiele narzędzi programistycznych. Jednak przeprowadzenie edycji i tworzenia planów kompozycji serwisów w surowym XMLu nie jest podejściem efektywnym i z tego powodu został opracowany algorytm konwersji Skierowanego Grafu Kompozycji Serwisów do planów kompozycji serwisów webowych w języku BPEL, który pozwala tworzyć i edytować plany kompozycji na wysokim poziomie abstrakcji.

2. Skierowany graf kompozycji serwisów

Skierowany Graf Kompozycji Serwisów jest rodzajem acyklicznego grafu skierowanego i może być przedstawiony jako para G = (N, L) oraz ograniczenia jej dotyczące:

1) N jest zbiorem skończonym,
2) $L \subseteq (N \times N)$,
3) $\forall\ x1 = (n1,n2) \wedge x2 = (n1, n2) \Rightarrow x1 = x2$.

N jest zbiorem wierzchołków dwóch podstawowych rodzajów: serwisowych oraz sterujących. Wierzchołki serwisowe reprezentują atomowe wywołania funkcjonalności serwisów webowych. Wierzchołki sterujące natomiast mają za zadanie odzwierciedlać miejsca kompozycji, w których następują procesy decyzyjne lub zmiany w przepływie sterowania. Niektóre typy wierzchołków sterujących mogą zawierać zagnieżdżone Skierowane Grafy Kompozycji Serwisów. L jest zbiorem skierowanych krawędzi i reprezentuje kierunek przepływu sterowania i danych w Skierowanym Grafie Kompozycji Serwisów.

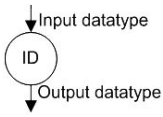
Ograniczenia Skierowanego Grafu Kompozycji Serwisów mówią nam że:

1) zbiór wierzchołków jest skończony, a więc graf jest również skończonym;
2) wierzchołki są połączone między sobą tylko za pomocą krawędzi;
3) tylko jedna krawędź może łączyć dwa wierzchołki, tym samym unika się powielenia połączeń.

Skierowany Graf Kompozycji Serwisów jest grafem acyklicznym.

2.1. Wierzchołki serwisów

Wierzchołek serwisu jest przedstawiony na rysunku 1 i jest głównym blokiem budowlanym każdej kompozycji. W Skierowanym Grafie Kompozycji Serwisów są one atomowe i odzwierciedlają pojedyncze wywołania serwisów webowych.



Rys. 1. Wierzchołek serwisu w Skierowanym Grafie Kompozycji Serwisów

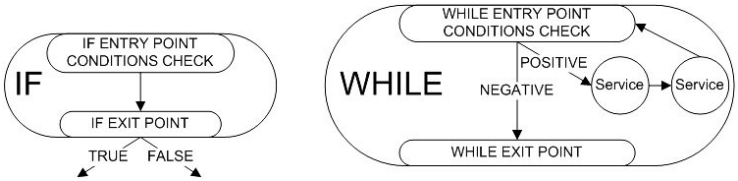
Każdy z nich charakteryzuje się następującymi cechami.

1) Jest pojedynczym wywołaniem jednej jednego serwisu webowego.
2) Posiada informację o swoich typach wejściowych i wyjściowych danych.
3) Posiada unikatowy numer identyfikacyjny ID.

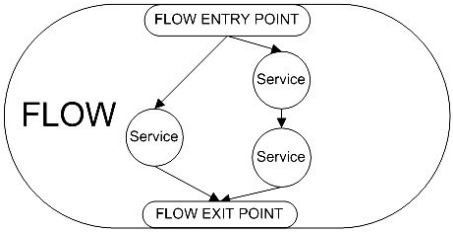
2.2. Wierzchołki sterujące

Wierzchołki sterujące umożliwiają zarządzanie przepływem sterowania oraz danych w kompozycjach serwisów, wprowadzają też możliwości zrównoleglenia pewnych części planów kompozycji. Każdy wierzchołek sterujący charakteryzuje się następującymi cechami.

1) Wierzchołek musi należeć do jednego z trzech typów: *IF, FLOW, WHILE* (patrz rys. 2 oraz 3).
2) Posiada unikatowy numer identyfikacyjny ID.
3) Każdy wierzchołek typu *IF* lub *WHILE* zawiera warunek decyzyjny, dzięki któremu zostaje podejmowana odpowiednia decyzja o dalszym przepływie sterowania i danych.
4) Wierzchołki *FLOW* oraz *WHILE* zawierają zagnieżdżone Skierowane Grafy Kompozycji Serwisów, które opisują kompozycje wykonywane w tych wierzchołkach sterujących.



Rys. 2. Wierzchołki sterujące typów IF oraz WHILE
w Skierowanym Grafie Kompozycji Serwisów



Rys. 3. Wierzchołek sterujący typu FLOW
w Skierowanym Grafie Kompozycji Serwisów

2.3. Krawędzie

Krawędzie Skierowanego Grafu Kompozycji Serwisów łączą między sobą wierzchołki wszystkich typów i charakteryzują się następującymi cechami.

1) Każda krawędź jednoznacznie określa kierunek łączenia dwóch wierzchołków o pewnych unikatowych identyfikatorach ID. Dzięki temu między połączonymi wierzchołkami zawsze można jednoznacznie wskazać kierunek przepływu sterowania i danych.
2) Może zawierać informację uzupełniającą. Przykładem jest informacją dotycząca wyboru ścieżki przepływu sterowania po wierzchołku sterującym typu *IF*.

3. Algorytm

Algorytm konwersji Skierowanych Grafów Kompozycji Serwisów do planów kompozycji serwisów webowych w języku BPEL jest algorytmem rekursywnie obchodzącym graf od wierzchołku korzenia (*root node*) w kierunku wskazywanym przez krawędzie skierowane. Wynikiem pracy jest kod BPEL. Sprawdzenie poprawności podawanego na wejście grafu nie jest przewidywane. Pseudokod algorytmu jest przedstawiony na rysunku 4.

```
1: INIT exporter (add appropriate intro BPEL code)
2: FIND root node in graph (dummy)
3: FIND nodes connected to root and PASS them to 4
4: PARSE nodes
    4_1: IF node TYPE is CONTROL and control TYPE is IF
        4_1_1: ADD BPEL code based on IF node properties
        4_1_2: FIND THEN and ELSE outgoing node arcs
        4_1_3: WITH destination nodes EXECUTE 4
        4_1_4: ADD IF BPEL outro code
    4_2: IF node TYPE is CONTROL and control TYPE is WHILE
        4_2_1: ADD BPEL code based on WHILE node properties
        4_2_2: GET Nested Composition Sequence
        4_2_3: WITH Nested Composition Sequence EXECUTE 2
        4_2_4: ADD WHILE BPEL outro code
        4_2_5: GOTO 5
    4_3: IF node TYPE is CONTROL and control TYPE is FLOW
        4_3_1: ADD BPEL code based on FLOW node properties
        4_3_2: GET Nested Composition Sequence
        4_3_3: WITH Nested Composition Sequence EXECUTE 2
        4_3_4: ADD FLOW BPEL outro code
        4_3_5: GOTO 5
    4_4: IF NODE TYPE is SERVICE
        4_4_1: ADD BPEL code based on SERVICE node properties
        4_4_2: GOTO 5
5: FIND outgoing directed arc
    5_1: IF EXISTS WITH destination node EXECUTE 4
6: FINALIZE exporter (add appropriate outruo BPEL code)
```

Rys. 4. Pseudokod algorytmu konwersji Skierowanego Grafu Kompozycji Serwisów

Kroki 1 oraz 6 inicjują i kończą działanie algorytmu również dodają kawałki kodu BPEL.

W kroku 2 algorytm szuka wierzchołek rozpoczynający graf.

W kroku 3 są wyszukiwane wierzchołki połączone z korzeniem i podawane do kroku 4.

Krok 4 parsuje wszystkie podane wierzchołki i dokonuje wyboru ich typu.

Kroki 4_1, 4_2, 4_3 i 4_4 rozpoczynają translacje wierzchołków *IF, WHILE, FLOW* i serwisu.

W krokach 4_1_1, 4_2_1, 4_3_1 oraz 4_4_1 do wyniku są dodawane części kodu w języku BPEL odpowiednio spreparowane na podstawie parametrów wierzchołku.

W kroku 4_1_2 są wyszukiwane wychodzące krawędzie typów *THEN* oraz *ELSE*.

W 4_1_3 wierzchołki na końcach znalezionych krawędzi są podawane do wywołania z kroku 3.

Kroki 4_2_2 oraz 4_3_2 pobierają zagnieżdżony Skierowany Graf Kompozycji Serwisów.

W krokach 4_2_3 oraz 4_3_3 zagnieżdżony graf zostaje przekazany do wywołania z kroku 2.

W krokach 4_1_4, 4_2_4 oraz 4_3_4 do wyniku jest dodawany odpowiedni kod w języku BPEL.

Kroki 4_2_5, 4_3_5 oraz 4_4_2 odsyłają algorytm do wykonania kroku 5.

W kroku 5 jest przeprowadzane wyszukiwanie krawędzi wychodzącej z bieżącego wierzchołki.

Jeśli taka istnieje, w kroku 5_1 wierzchołek na końcu krawędzi zostaje przekazany do kroku 4.

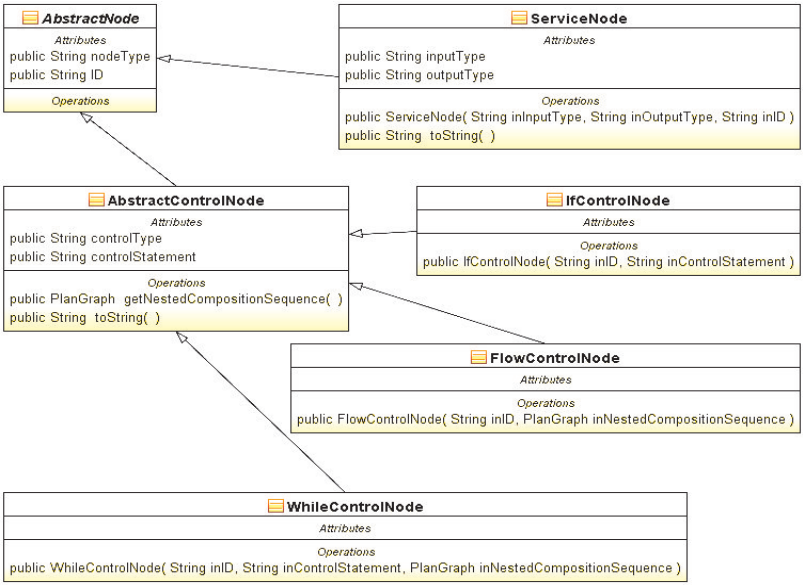
4. Implementacja oraz wyniki

Implementacja algorytmu konwersji Skierowanych Grafów Kompozycji Serwisów do planów kompozycji w języku BPEL była wykonana jako aplikacja napisana w języku programowania Java. Graf podlegający konwersji jest przygotowany za pomocą algorytmu konwersji planów w języku BPEL do postaci grafowej. Wynikiem działania oprogramowania jest kod BPEL.

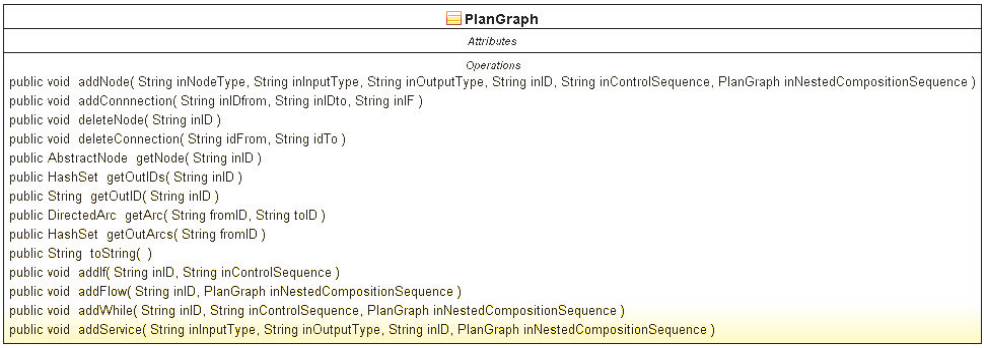
Jądro systemu zostało zbudowano przy wykorzystaniu biblioteki grafowej JgraphT. Biblioteka JGraph była wykorzystana dla wizualizacji. Struktura grafu była zdefiniowana jako graf skierowany jgrapht w którym klasa abstrakcyjna „*AbstractNode*" jest wierzchołkiem a klasa „*DirectedArc*" jest pochodna klasy bibliotecznej „*DefaultEdge*" i służy dla reprezentacji skierowanych krawędzi grafu.

Rysunek 5 przedstawia klasę „*AbstractNode*" oraz hierarchię klas pochodnych.

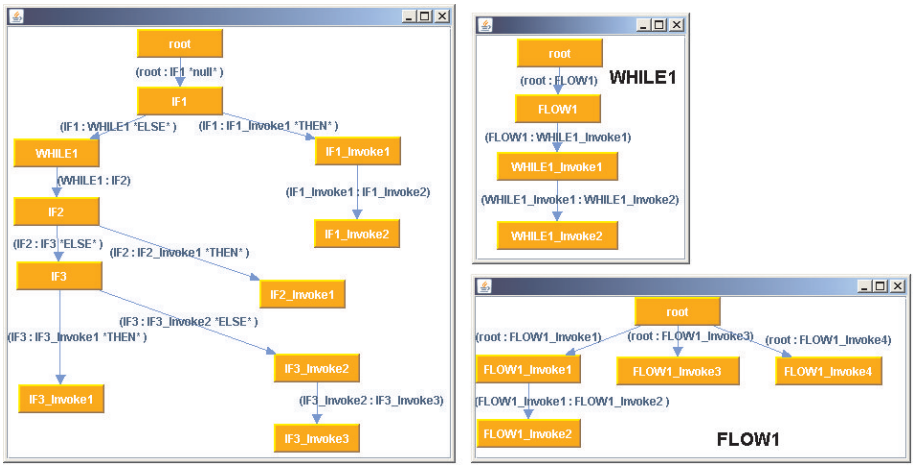
Rysunek 6 przedstawia API struktury grafowej oraz klasę implementującą algorytm jej konwersji.



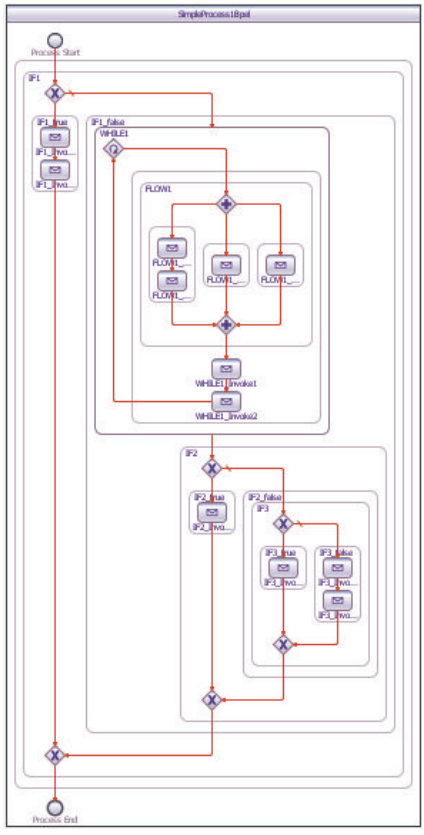
Rys. 5. Hierarchia klas wierzchołków Skierowanego Grafu Kompozycji Serwisów



Rys. 6. API Skierowanego Grafu Kompozycji Serwisów oraz klasa algorytmu konwersji



Rys. 7. Przykładowy Skierowany Graf Kompozycji Serwisów



Rys. 8. Wizualizacja wynikowego planu kompozycji serwisów

4.1. Przykładowy wynik pracy systemu informatycznego

Na rysunku 7 jest przedstawiony Skierowany Graf Kompozycji Serwisów (razem z grafami zagnieżdżonymi), który został poddany algorytmowi konwersji. Widać, że graf jest strukturą rozbudowaną i posiada wywołania serwisów i instrukcje warunkowe: *IF*, *WHILE*, *FLOW*, przy czym instrukcja *FLOW* jest zagnieżdżona w instrukcji *WHILE*.

W wyniku konwersji przedstawionego grafu powstał plan kompozycji serwisów w języku BPEL (rys. 8 przedstawia jego wizualizację schematyczną). Należy dodać, iż wynikowy plan kompozycji okazał się całkowicie zgodny z planem pierwotnym, który na początku posłużył jako wzorzec dla utworzenia Skierowanego Grafu Kompozycji Serwisów z rysunku 7.

5. Podsumowanie

Proponowana metoda skupia się na ułatwieniu procesu edycji planów kompozycji serwisów i pozwala poszerzyć możliwości tworzenia i edycji planów w języku BPEL za pomocą użycia Skierowanego Grafu Kompozycji Serwisów oraz przedstawionego algorytmu konwersji.

Mimo tego, że przedstawiona metoda nie uwzględnia całkowitego zbioru możliwości języku BPEL, jest ona jak najbardziej akceptowalna do użycia i może być łatwo poszerzona o takie elementy jak nowe typy wierzchołków, zmienne oraz inne cechy języka BPEL.

Obecnie algorytm jest częścią eksperymentalnego oprogramowania, które na celu ma implementację hybrydowego podejścia do kompozycji serwisów [1].

Literatura

[1] Belava L., *Koncepcja hybrydowej kompozycji usług w środowisku SOA.* Automatyka (półrocznik AGH), 13/2, 2009, 189–197.

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

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

[4] Belava L., *Concept of Hybrid Service Composition In SOA Environment with Agent Enrichment*. Proceedings of the 2010 Second Forum of Young Researcher, 2010, 117–122.

[5] Kavantzas N., Burdett D., Barreto C., Ritzinger G., Fletcher T., *Web Services Choreography Description Language Version 1.0.* http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/ 2005.

[6] Dean M., Connolly D., Harmelen F., Hendler J., Horrocks I., McGuinness D. L., *OWL Web Ontology Language 1.0.* Reference http://www.w3.org/TR/2002/WD-owl-ref-20020729/ 2004.

[7] Martin D., Burstein M., Hobbs J., Lassila O., McDermott D., McIlraith S., Narayanan S., *OWL-S: Semantic Markup for Web Services*. http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/ 2004.

[8] Ankolekar A., Burstein M., Hobbs J., Lassila O., Martin D., *DAML-S:Web Service Description for the Semantic Web.* Proceedings of the International Semantic Web Conference (ISWC) 2002, 348–363.

[9] Hamadi R., Benatallah B., *Petri Net-based model for Web Service Composition.* Proceedings of the 14th Australasian database conference on Database technologies, 2003, 191–200.

[10] Sirin E., Parsia B., Wu D., Hendler J., Nau D., *HTN planning for Web Service composition using SHOP2.* Web Semantics: Science, Services and Agents on the World Wide Web, 2004, 4, 377–396.

[11] Li. B., Zhou Y., Pang J., *Model-driven Automatic Generation of Verified BPEL Code for Web Service Composition*. Proceedings of 16th Asia-Pacific Software Engineering Conference, 2009, 355–362.

[12] Ionita C., Costan A., Cristea V., *Automatic Generation of Functional Workflows Using a Semantic Specification*. 2010 International Conference on Complex, Intelligent and Software Intensive Systems, 496–501.

[13] Díaz G., Cambronero M.E., Pardo J.J. , Valero V., Cuartero F., *Automatic generation of Correct Web Services Choreographies and Orchestrations with Model Checking Techniques*. Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services, 2006.

[14] Mohanty H., Chenthati D., Vaddi S., *Automatic Generation of BPEL and WSDL from FSM models of Web Services*. Advanced Computing and Communications, 2006, 440–444.

[15] Ning G., Zhu Y., Lu T., Wang F., *BPELGEN: An Algorithm of Automatically Converting from Web Services Composition Plan to BPEL4WS*. Pervasive Computing and Applications, 2007.

[16] Gui Z., Wu H., Wang Z., *A Data Dependency Relationship Directed Graph and Block Structures Based Abstract Geospatial Information Service Chain Model*. Proceedings Fourth International Conference on Networked Computing and Advanced Information Management, 2, 2008, 21–27.