



POLITECHNIKA WARSZAWSKA
Wydział Elektroniki i Technik Informacyjnych
Instytut Telekomunikacji

PRACA DYPLOMOWA INŻYNIERSKA

Marcin Maciorowski

**Tworzenie narzędzi wspomagających projektowanie BPEL
w architekturze SOA**

Praca wykonana pod kierunkiem
dra inż. Andrzeja Ratkowskiego

.....
Ocena pracy

.....
Podpis Przewodniczącego Komisji

Warszawa, 2014

Życiorys

Urodziłem się 22 września 1988 roku w Radzynie Podlaskim. W 2004 roku rozpocząłem naukę w I Liceum Ogólnokształcącym w Radzynie Podlaskim, gdzie uczęszczałem do klasy o profilu matematyczno-fizyczno-informatycznym. Po uzyskaniu świadectwa dojrzałości w 2007 roku, rozpocząłem studia na Politechnice Warszawskiej na Wydziale Elektroniki i Technik Informatycznych. W trakcie studiów wybrałem specjalizację Systemy informacyjno-decyzyjne prowadzoną przez Instytut Automatyki i Informatyki Stosowanej.

Marcin Maciorowski

Streszczenie

BP EL.

Abstract

Implementation of (Font TNRoman 12 Normal).

This thesis includes the design and testing procedure of

Spis treści.

1.	Wstęp.....	5
2.	Cel pracy.....	6
3.	Układ pracy.....	6
4.	Opis języka BPEL.....	7
4.1.	Zmienna procesu BPEL (<i>Variable</i>).	7
4.2.	Instrukcja wywołania usługi (<i>Invoke</i>).	7
4.3.	Instrukcja przepisania danych (<i>Assign</i>).	7
5.	Wtyczka Eclipse BPEL Designer.	8
5.1.	Interfejs użytkownika.	8
5.1.1.	Edytor.	9
5.1.2.	Widoki.	10
5.2.	Przykładowy proces BPEL.	14
6.	Wtyczka generująca instrukcje kopiujące.	17
6.1.	Konfiguracja wtyczki (PDE).	20
6.2.	Transformacja procesu z postaci EMF do postaci grafu.	18
6.3.	Analizator grafu procesu.....	20
6.3.1.	Architektura. Error! Bookmark not defined.	
6.3.2.	Mechanizm dopasowania zmiennych.	20
6.4.	Graficzny interfejs użytkownika.....	20
7.	Testy.	22
8.	Podsumowanie.	23
8.1.	Napotkane problemy.....	23
8.2.	Możliwości rozwoju.	23
9.	Bibliografia.	24
10.	Załączniki.	25
10.1.	Płyta CD.....	25
10.2.	Instrukcja instalacji wtyczki BPELag (BPEL assign generator).	25

1. Wstęp.

=====

We wstępie znajdzie się ogólne rozwinięcie streszczenia, czego praca dotyczy, z czym czytelnik się zetknie w kolejnych rozdziałach. Opisany układ dokumentu.

=====

Co to jest proces biznesowy, co to jest web service. WSDL. WS-BPEL opisuje w podejściu zorientowanym na usługi działanie procesów biznesowych w instytucji.

2. Cel pracy.

Celem pracy jest napisanie wtyczki do zintegrowanego środowiska programistycznego jakim jest Eclipse, która umożliwi automatyczne uzupełnienie zaprojektowanego procesu BPEL o instrukcje kopiujące dane. Wtyczka ma za zadanie dokonać analizy procesu oraz na podstawie wyników analizy wygenerować instrukcje kopiujące. Dalej umożliwiać użytkownikowi/projektantowi przegląd wyników analizy oraz dodatkowo manualną edycję wszystkich uczestniczących w procesie elementów przepisania danych (*Assign*), akceptację lub rezygnację z zapisu zmian do projektowanego procesu. Analizowany proces BPEL oraz uczestniczące w procesie usługi wykorzystują tę samą konwencję nazewnictwa.

3. Układ pracy

Opis układu pracy. Powstanie na końcu, gdy zostaną napisane już wszystkie rozdziały.

4. Opis języka BPEL.

=====

Wstępny opis języka BPEL wprowadzający czytelnika w aspekty języka, których dotyczy niniejsza praca dyplomowa. Na pewno będą to: Struktura procesu, bloki assign, invoke.

=====

4.1. Zmienna procesu BPEL (*Variable*).

=====

Bla bla bla.

=====

4.2. Instrukcja wywołania usługi (*Invoke*).

=====

Bla bla bla.

=====

4.3. Instrukcja przepisania danych (*Assign*).

=====

Bla bla bla.

=====

5. Wtyczka Eclipse BPEL Designer.

Eclipse BPEL Designer jest wtyczką integrującą się ze zintegrowanym środowiskiem programistyczne (ang. *Integrated Development Environment – IDE*) Eclipse dostarczając własnej perspektywy wspomagającej projektowanie procesów BPEL. Wtyczka dostarcza wsparcie w definiowaniu, edytowaniu, instalacji oraz testowaniu i debuggowaniu procesów WS-BPEL 2.0, czyli języka do definiowania procesów biznesowych opartego o usługi sieciowe, dostarczonego przez konsorcjum OASIS.

Główne cechy wtyczki:

- *Designer* – edytor graficzny (oparty o GEF – Graphical Editing Framework) wprowadzający graficzne oznaczenia elementów procesu BPEL.
- *Model* – reprezentacja modelu BPEL (specyfikacja WS-BPEL 2.0) reprezentowana przez model oparty o EMF (Eclipse Modelling Framework).
- *Validation* – operujący na modelu EMF walidator informujący o błędach i ostrzeżeniach dotyczących procesu BPEL, wynikających ze specyfikacji.
- *Runtime Framework* – zestaw narzędzi umożliwiających instalację oraz wykonanie procesu BPEL.
- *Debug* – zestaw narzędzi umożliwiający śledzenie kolejnych kroków wykonywanego procesu oraz dostarczających obsługę przerw wywołania.

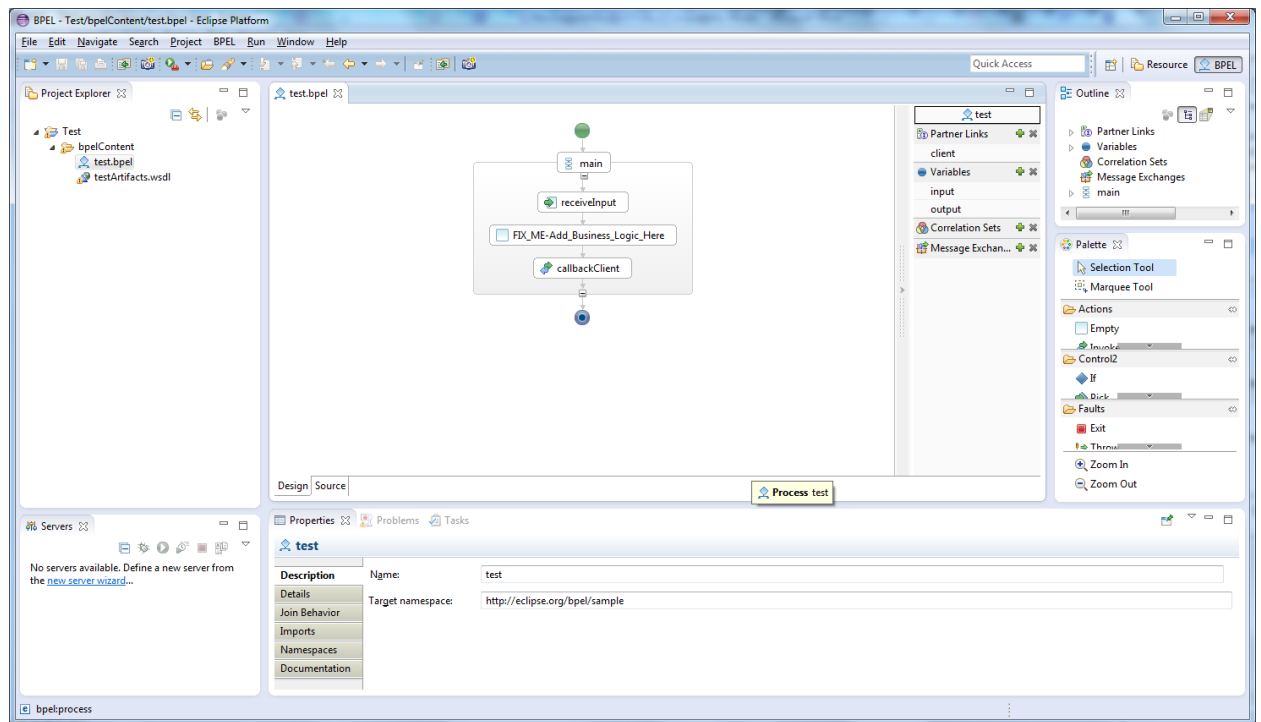
W niniejszej pracy wykorzystywana jest wtyczka Eclipse BPEL Designer w wersji 1.0.3.

5.1. Interfejs użytkownika.

Wtyczka Eclipse BPEL Designer dostarcza własnej perspektywy dodając:

- Okno edytora procesów BPEL (umożliwiający również edycję kodu BPEL).
- Widok *Palette* zawierający graficzne elementy będące reprezentacją znaczników języka BPEL biorących udział w procesie.
- Widok *Outline* - schemat procesu w postaci drzewa elementów.
- Widok *Properties* służący do modyfikacji konfiguracji poszczególnych elementów w procesie – specyficzny dla różnych elementów.

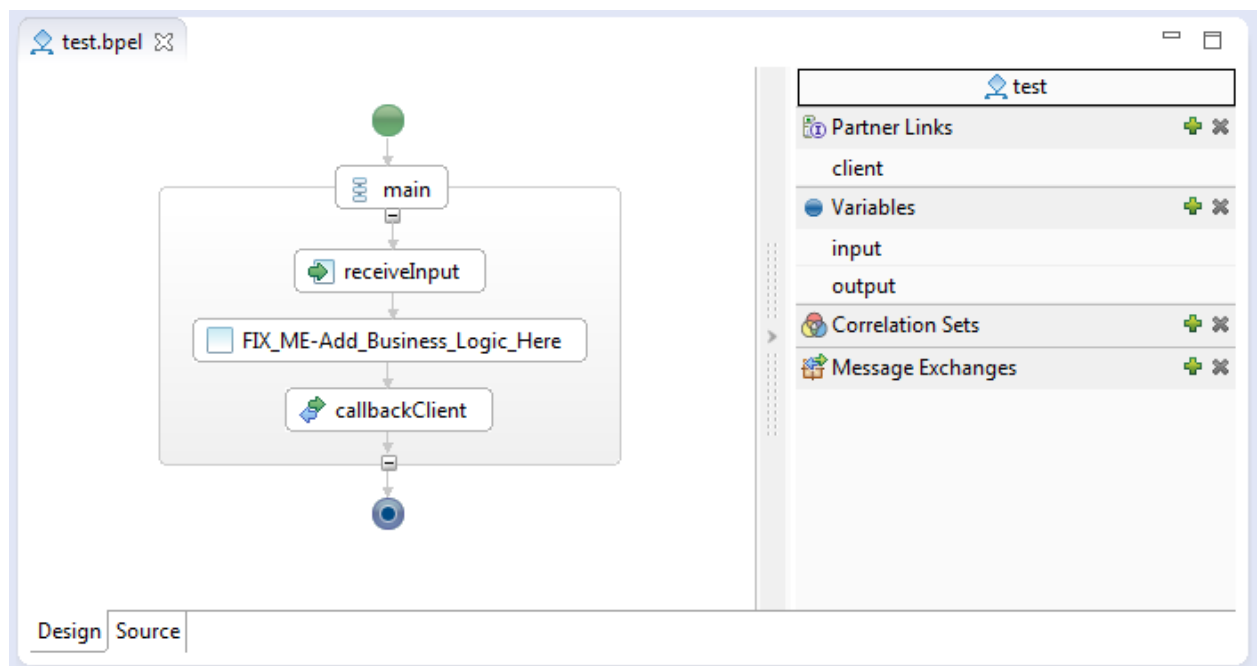
Na rys. 5.1 przedstawiono wygląd IDE Eclipse w perspektywie BPEL.



Rys. 5.1 UI wtyczki Eclipse BPEL Designer.

5.1.1. Edytor.

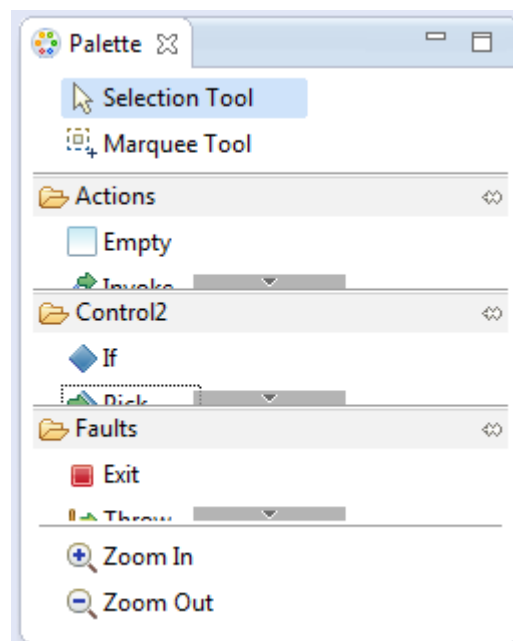
Edytor projektu umożliwia podgląd procesu oraz jego edycję przy użyciu edytora graficznego oraz standardowego edytora kodu BPEL. Edytor graficzny dostarcza możliwość modyfikacji reprezentacji graficznej procesu wizualizującej strukturę głównej sekwencji przebiegu przy użyciu dostarczonych przez wtyczkę Eclipse BPEL Designer graficznych reprezentacji aktywności (*Activity*) BPEL. Graficzny edytor umożliwia również edycję elementów wchodzących w skład procesu takich jak zmienne (*Variables*), połączenia z partnerami biorącymi udział w procesie (*Partner Links*), listę definicji korelacji (*Correlation Sets*) i definicje wiadomości używanych do komunikacji dostawca-konsument usługi (*Message Exchanges*). Ponadto proces może być również edytowany przy użyciu standardowego edytora kodu (zakładka *Source* na Rys. 5.2).



Rys. 5.2 Edytor procesu BPEL – reprezentacja graficzna.


5.1.2. Widoki.



















Jednym z widoków jakie dostarcza Eclipse BPEL Designer jest widok palety, zawierający elementy wykorzystywane do zbudowania struktury procesu w oknie edytora.




Rys. 5.3 Widok palety BPEL Designer

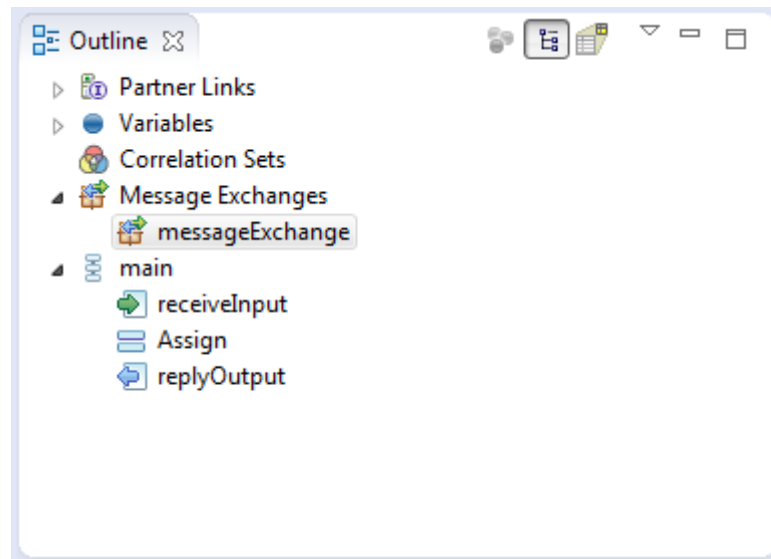
Tab. 5.1 Opis elementów palety BPEL Designer.

Element palety	Opis
 Empty	instrukcja pusta (<i><empty></i>)

 Invoke	wywołanie zewnętrznej usługi sieciowej (<i><invoke></i>)
 Receive	odebranie komunikatu wywołania procesu (<i><receive></i>)
 Reply	odpowiedź wysyłana do obiektu wywołującego process (<i><reply></i>)
 Opaque Activity	
 Assign	instrukcja przypisania wartości zmiennej (<i><assign></i>)
 Validate	walidacja wartości zmiennej na podstawie definicji typu (<i><validate></i>)
 If	warunkowy wybór jednej instrukcji do wykonania (<i><if></i>)
 Pick	oczekiwanie na nadejście wiadomości lub przekroczenie czasu oczekiwania (<i><pick></i>)
 While	pętla wykonywana dopóki warunek jest spełniony (<i><while></i>)
 For Each	pętla wykonywana określoną ilość razy (<i><forEach></i>)
 Repeat Until	pętla wykonywana do spełnienia warunku (<i><repeatUntil></i>)
 Wait	oczekiwanie przez określony czas lub do określonego momentu (<i><wait></i>)
 Sequence	zbiór instrukcji uruchamianych sekwencyjnie (<i><sequence></i>)
 Scope	definicja nowej aktywności (<i><scope></i>)
 Flow	zbiór instrukcji wykonywanych jednocześnie (<i><flow></i>)
 Exit	natychmiastowe zakończenie wykonywanego procesu (<i><exit></i>)
 Throw	instrukcja generująca wystąpienie błędu w procesie (<i><throw></i>)
 Rethrow	instrukcja generująca wystąpienie już obsługiwanego błędu (<i><rethrow></i>)
 Compensate	odwrócenie działania wszystkich zakończonych wewnętrznych instrukcji procesu (<i><compensate></i>)

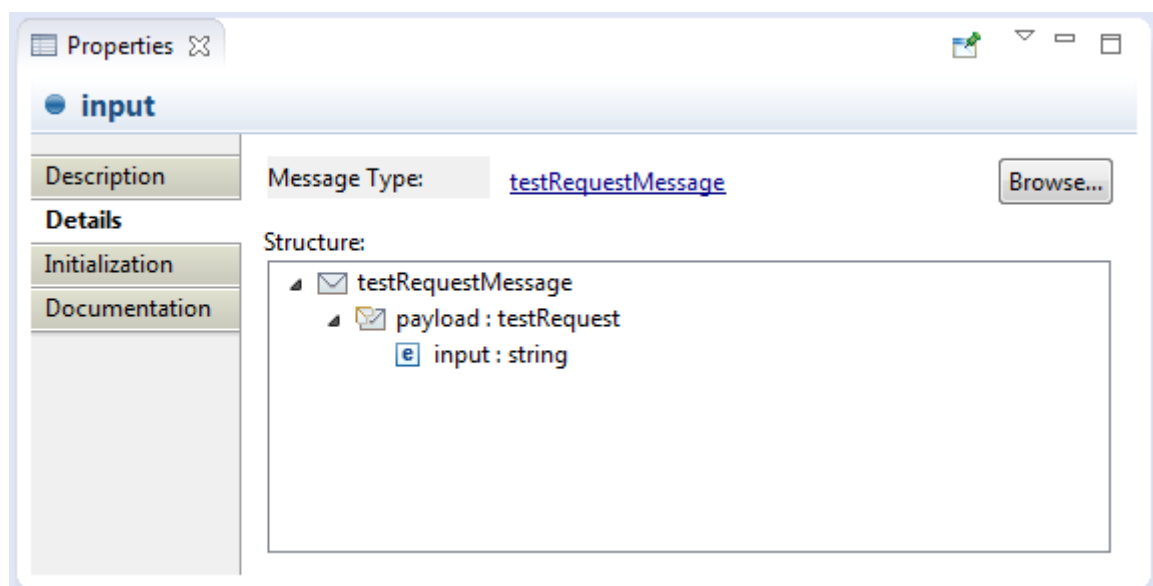
 CompensateScope	odwrócenie działania określonej wewnętrznej instrukcji procesu (<code><compensate></code>)
--	---

Kolejny widok jakim dysponuje Eclipse BPEL Designer jest schemat procesu widoczny na rysunku Rys. 5.4. Schemat przy użyciu struktury drzewiastej obrazuje strukturę zagnieżdżeń wszystkich elementów w całym procesie.



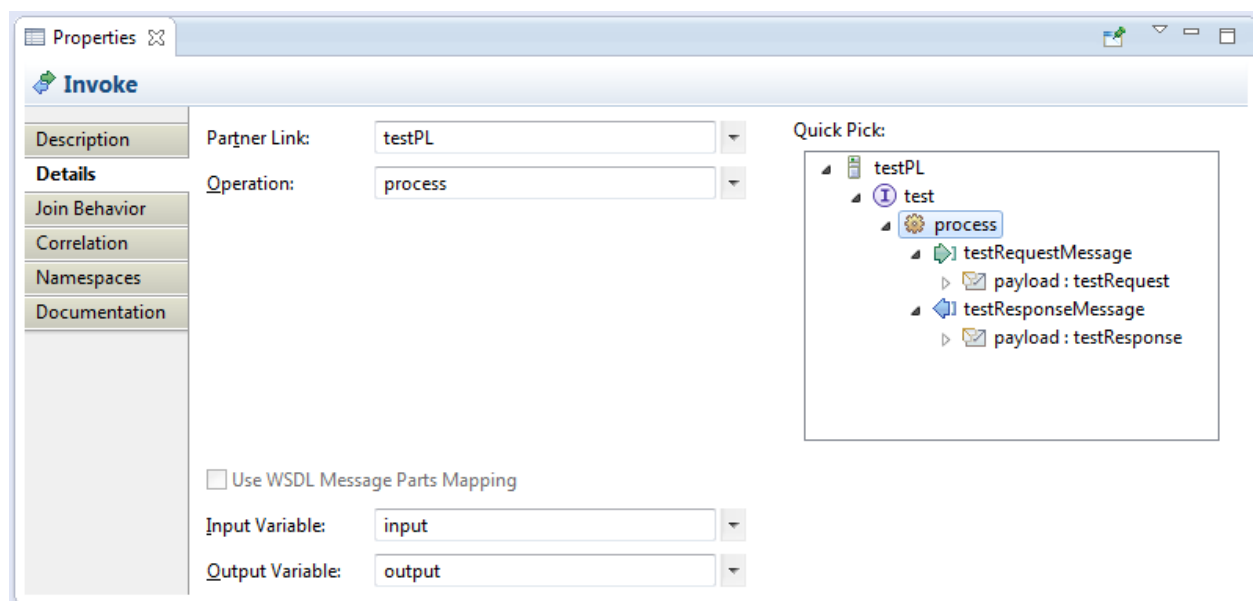
Rys. 5.4 Widok schematu procesu.

Widok konfiguracji (*Properties*) poszczególnych elementów procesu jest ściśle związany z rodzajem edytowanego elementu. Każdy widok konfiguracyjny zawiera elementy podwidoki wspólne dla wszystkich elementów procesu oraz podwidoki specyficzne dla aktualnie edytowanego procesu. Na rysunku Rys. 5.5 przedstawiono szczegółową konfigurację zmiennej procesu umożliwiającą określenie typu danych konfigurowanej zmiennej, który może być typem prostym lub złożonym. Po wyborze typu wyświetlana jest jego struktura, przy czym w przypadku wyboru typu prostego ogranicza się ona do jednego poziomu.



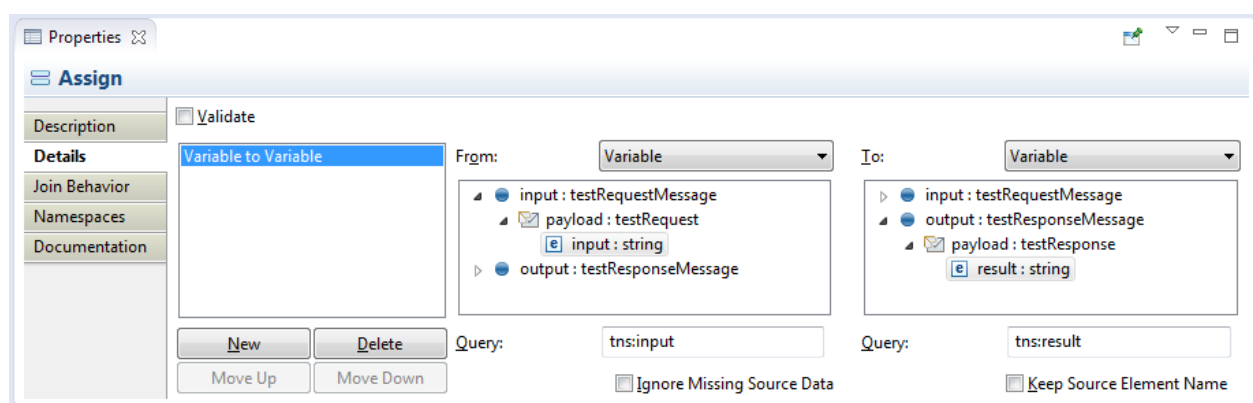
Rys. 5.5 Szczegółowa konfiguracja zmiennej procesu (*Variable*).

Na rysunku Rys. 5.6 przedstawiono widok szczegółowej konfiguracji instrukcji wywołania usługi zewnętrznej umożliwiającą zdefiniowanie partnera procesu oraz operacji wywołanej przez instrukcję. Dokonanie wyboru możliwe jest poprzez sekwencyjne określenie partnera procesu z listy rozwijanej, a następnie operacji – z listy operacji publikowanych przez usługę – z listy rozwijanych, lub przy użyciu sekcji szybkiego wyboru. Po określeniu operacji usługi zewnętrznej wywołanej w konfigurowanej instrukcji konieczne jest również określenie zmiennych procesu, które powinny zostać użyte do wywołania usługi (*Input Variable*) oraz do której zostanie zapisany wynik wywołania operacji usługi zewnętrznej (*Output Variable*).



Rys. 5.6 Szczegółowa konfiguracja instrukcji wywołania usługi zewnętrznej (*Invoke*).

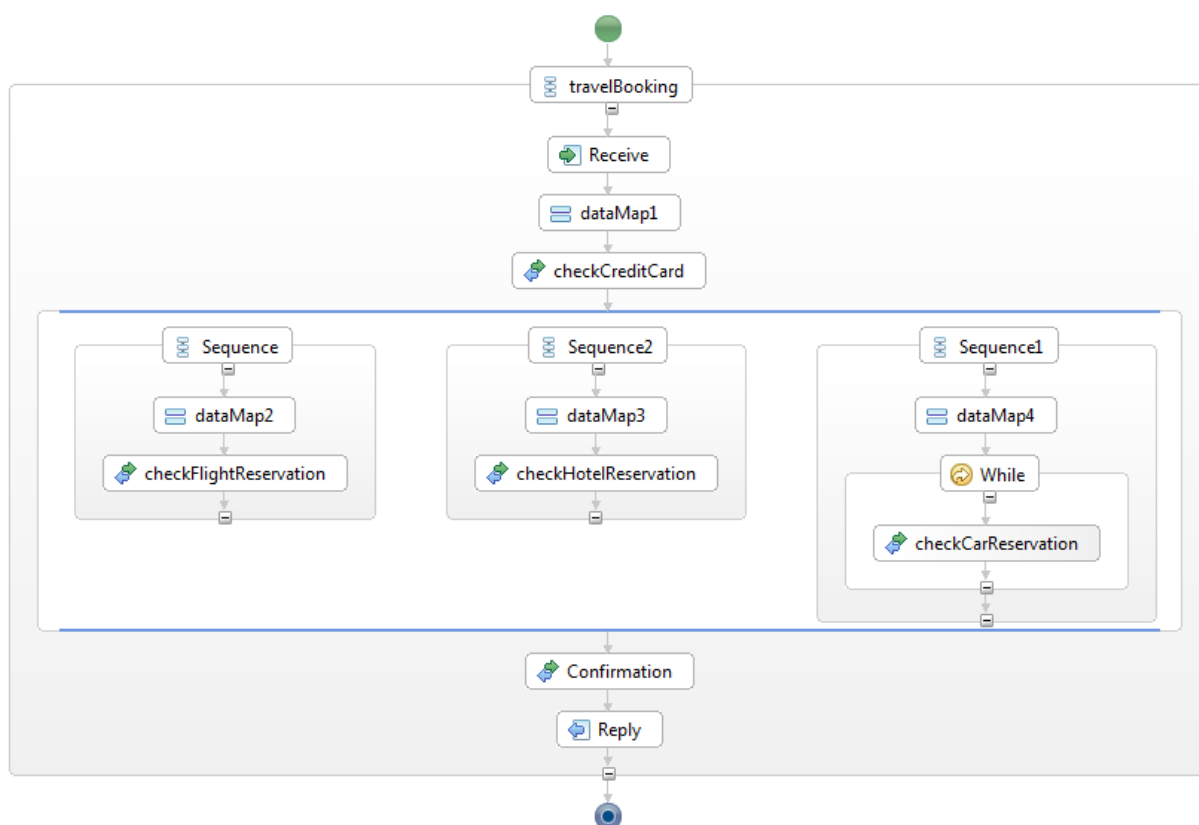
Na rysunku Rys. 5.6 przedstawiono widok szczegółowej konfiguracji instrukcji przepisania wartości zmiennej/zmiennych w procesie. Widok dostarcza możliwość tworzenia/usuwania/zmiany kolejności instrukcji kopiujących w instrukcji przepisania. Możliwe jest również określenie rodzaju elementu z którego oraz do którego wartość ma zostać skopiowana, a następnie określenie tych elementów. Na rysunku zaprezentowano instrukcję przepisania wartości z jedną instrukcją kopiującą. Wybrany typ elementu źródłowego to zmienna procesu, element źródłowy to zmienna typu prostego wchodząca w skład typu złożonego wybranej zmiennej procesu. Podobnie jest w przypadku elementu docelowego.



Rys. 5.6 Szczegółowa konfiguracja instrukcji przepisania wartości (*Assign*).

5.2. Przykładowy proces BPEL.

Na Rys. 5.1 przedstawiony został przykładowy proces BPEL utworzony przy użyciu Eclipse BPEL Designera, na podstawie procesu rezerwacji wycieczki [2]. W momencie wywołania procesu rezerwacji, zostają mu przekazane informacje dotyczące karty kredytowej, celu oraz okresie podróży. Poprzez wywołanie zewnętrznych usług następuje najpierw sprawdzenie dostępności środków – na karcie kredytowej, następnie równoległe rezerwacja lotu, hotelu oraz samochodu. Po zakończeniu równoległych przebiegów do konsumenta usługi trafia żądanie potwierdzenia rezerwacji, po którym zostaje wysłana informacja o poprawnym zakończeniu procesu.



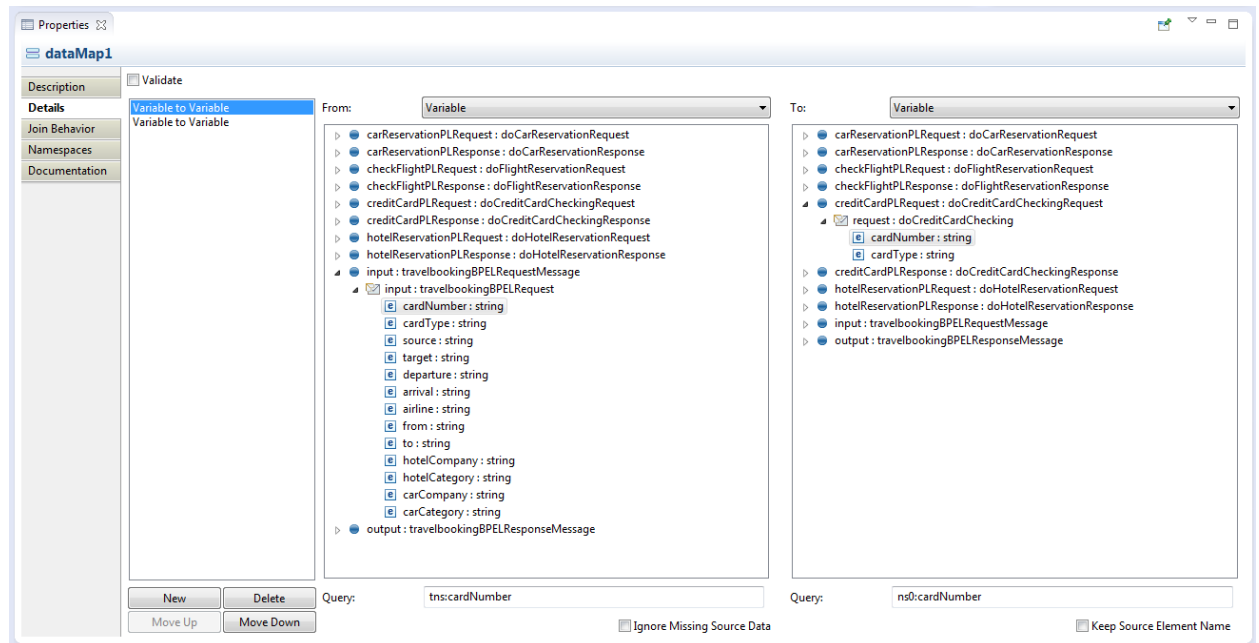
Rys. 5.1 Przykładowy proces BPEL utworzony w Eclipse BPEL Designer – *travelBooking*

W przedstawionym procesie występują trzy bloki przepisania danych (*Assign*):

- *dataMap1* – zawiera instrukcje kopiujące odpowiednie wartości wejściowe procesu do zmiennych będących elementami parametru wywołania usługi *checkCreditCard*.
- *dataMap2* – analogicznie do *dataMap1* dla usługi *checkFlightReservation*.
- *dataMap3* – analogicznie do *dataMap1* dla usługi *checkHotelReservation*.

- *dataMap4* – analogicznie do *dataMap1* dla usługi *checkCarReservation*.

Na Rys. X przedstawiona została konfiguracja instrukcji kopiujących dane na przykładzie bloku przepisywania danych *dataMap1*. Sekcja zawiera listę instrukcji kopiujących opisanych jako para typów (elementu źródłowego oraz elementu docelowego dla instrukcji kopiowania) oraz dwie listy zmiennych o zasięgu nie mniejszym niż aktualnie konfigurowany blok *Assign*. W obu listach *From* oraz *To* zaznaczono zmienne odpowiednio źródłowa i docelowa.



Rys. 3. Sekcja *Properties* bloku przepisywania danych *dataMap1*, zakładka z listą instrukcji kopiujących.

Na Rys. 4. Przedstawiono wygenerowany przez Eclipse BPEL Designer kod w języku BPEL odpowiadający konfiguracji przedstawionej na Rys. 3. dla bloku *dataMap1*.

```

69 <bpel:assign validate="no" name="dataMap1">
70   <bpel:copy>
71     <bpel:from part="input" variable="input">
72       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
73         <![CDATA[tns:cardNumber]]>
74       </bpel:query>
75     </bpel:from>
76     <bpel:to part="request" variable="creditCardPLRequest">
77       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
78         <![CDATA[ns0:cardNumber]]>
79       </bpel:query>
80     </bpel:to>
81   </bpel:copy>
82   <bpel:copy>
83     <bpel:from part="input" variable="input">
84       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
85         <![CDATA[tns:cardType]]>
86       </bpel:query>
87     </bpel:from>
88     <bpel:to part="request" variable="creditCardPLRequest">
89       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
90         <![CDATA[ns0:cardType]]>
91       </bpel:query>
92     </bpel:to>
93   </bpel:copy>
94 </bpel:assign>

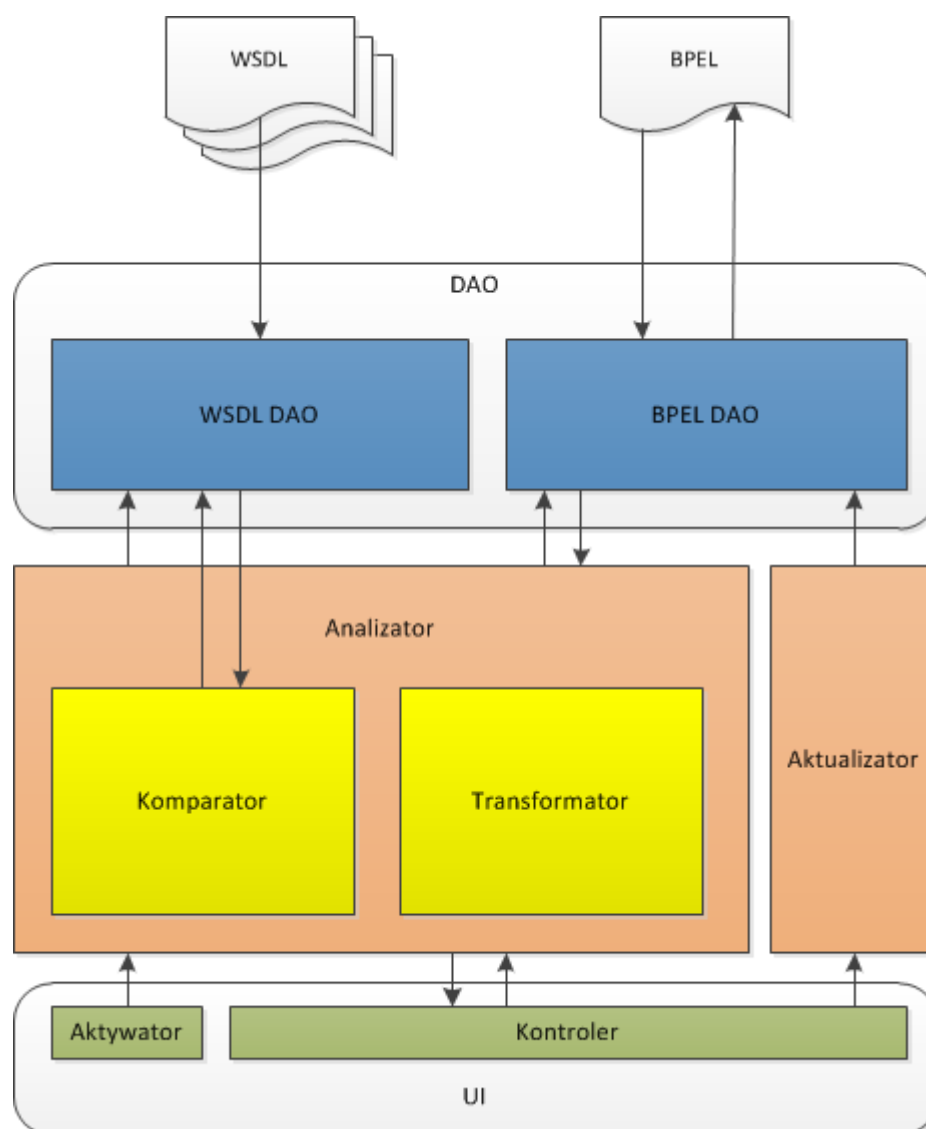
```

Rys. 4. Kod BPEL bloku *dataMap1*.

6. Wtyczka generująca instrukcje kopiujące.

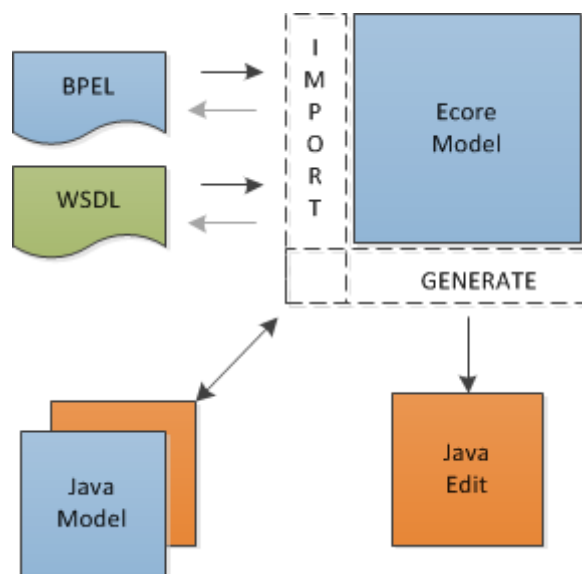
W ramach niniejszej pracy dyplomowej inżynierskiej została opracowana wtyczka generująca instrukcje kopiujące dla instrukcji przepisania wartości zmiennych w procesie BPEL. Wtyczka stanowi rozszerzenie IDE Eclipse bazując na wyniku analizy procesu przetransformowanego z postaci EMF – w jakiej znajduje się po wczytaniu – do postaci grafu. W podstawowym scenariuszu istnieje plik procesu BPEL, który zostaje wczytany przy użyciu obiektu wczytującego. Następnie wczytany proces zostaje przetransformowany do postaci grafu. Graf poddany zostaje analizie przeprowadzonej przez analizator. Wyniki analizy w postaci listy instrukcji kopiujących zmapowanych na konkretne instrukcje przepisania wartości zmiennych występujących w procesie zostają wykorzystane przez aktualizator do załadowania do procesu.

Na rysunku Rys. 6.1 przedstawiono model architektury omawianej wtyczki. Dostępem do danych – opublikowane pliki opisujące usługi sieciowe oraz proces biznesowy BPEL – zajmują się obiekty DAO (*Data Access Object*). Komponenty DAO zostały zastosowane w celu oddzielenia warstwy dostępu do danych od logiki znajdującej w Analizatorze oraz Aktualizatorze i warstwy prezentacji przedstawionych w modelu architektury.



Rys. 6.1 Model architektury wtyczki generującej instrukcje kopiujące.

Zaprezentowany schemat architektury przedstawia dwa obiekty dostępu do danych. BPEL DAO użyty jest do wczytania oraz zapisu procesu biznesowego z i do pliku zawierającego jego definicję (*.bpel) do modelu BPEL Designer przy użyciu modelu EMF. Drugim obiektem dostępu do danych jest WSDL DAO używany do wczytania plików definiujących opis usług sieciowych biorących udział w procesie BPEL. Na rysunku Rys. 6.2 przedstawiono schemat importu plików *.bpel oraz *.wsdl przy użyciu modeli EMF do modeli Java.

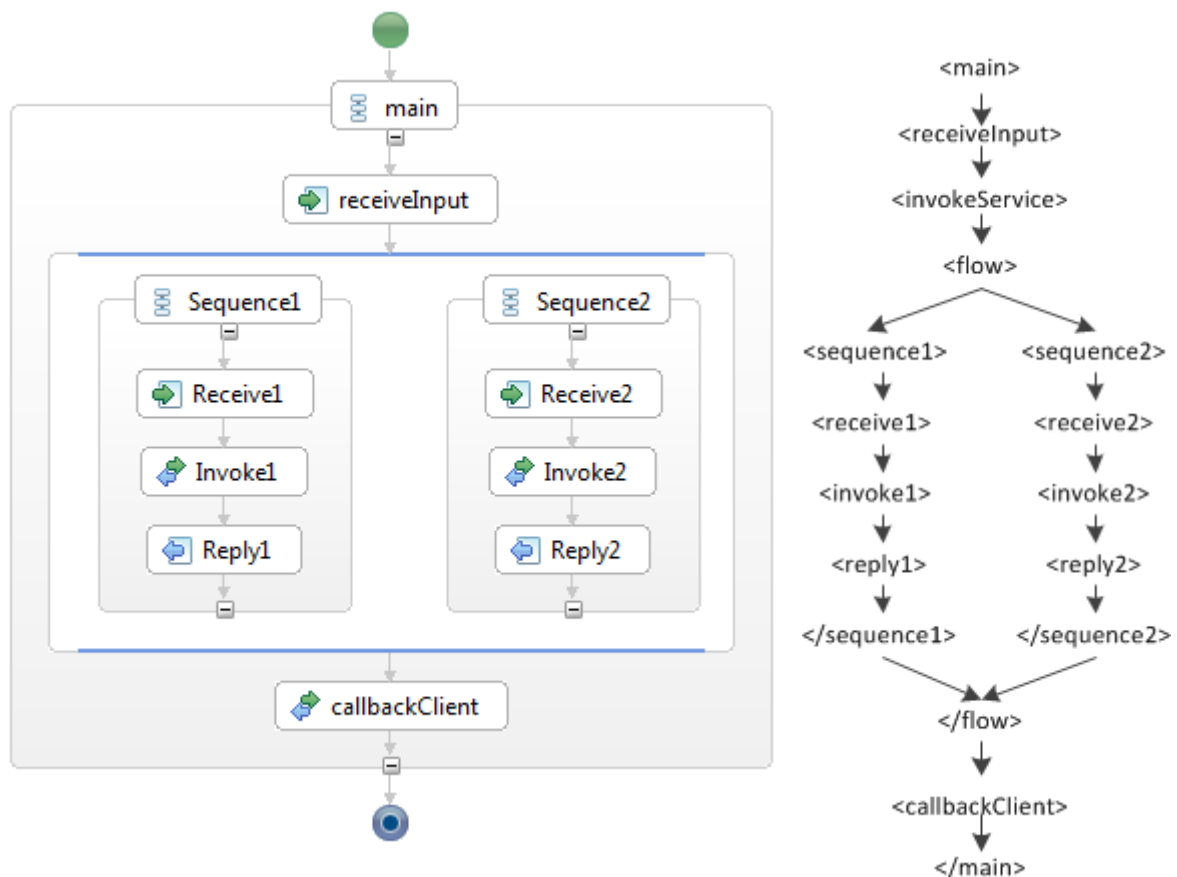


Rys. 6.2 Schemat importu przy użyciu Eclipse Modelling Framework (EMF) na podstawie [5]

Komunikacja z obiektem BPEL DAO przedstawiona na rysunku Rys. 6.1 odbywa się dwukierunkowo w przypadku Analizatora – wysłanie żądania wczytania procesu oraz w odpowiedzi odebranie wczytanego procesu w postaci obiektu EMF. W przypadku Aktualizatora komunikacja z obiektem BPEL DAO zachodzi jednokierunkowo – wysłanie żądania do zapisu procesu w postaci EMF do pliku. Komunikacja z obiektem WSDL DAO prezentowanym na rysunku Rys. 6.1 zachodzi jednokierunkowo z Analizatorem – wysłanie żądania do wczytania listy plików zawierających opis usług sieciowych. Dwukierunkowa komunikacja z WSDL DAO zachodzi natomiast z Komparatorem - jego zadaniem jest wyszukiwanie dopasowań pomiędzy zmiennymi procesu zarówno typów prostych jak i złożonych – polegając na wysłaniu żądania przesłania wczytanego do postaci EMF pliku WSDL oraz w odpowiedzi przekazanie obiektu.

6.1. Transformacja procesu do postaci grafu.

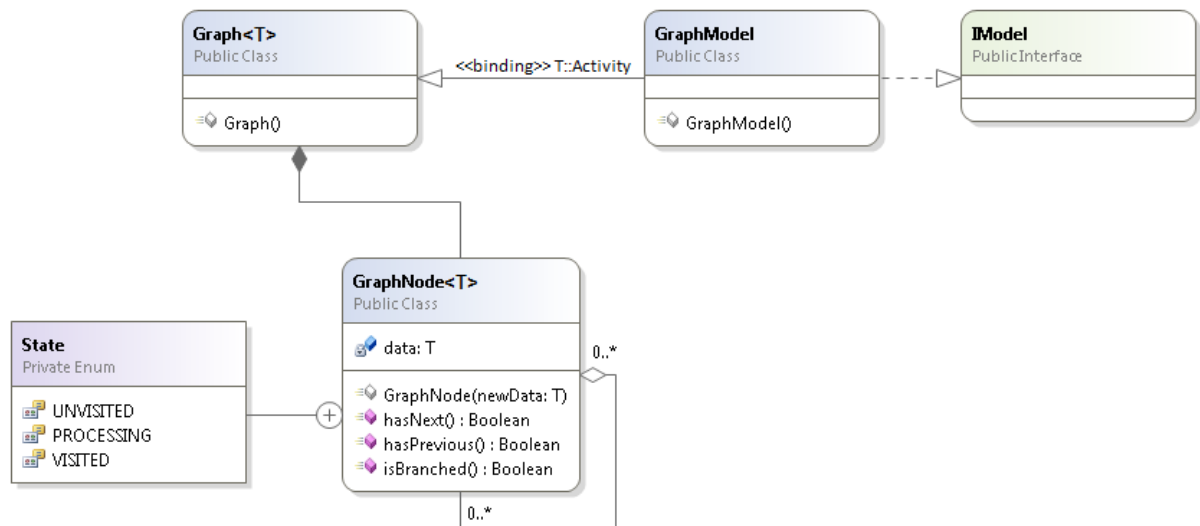
W ramach projektu wtyczki został zaprojektowany oraz zaimplementowany Transformator tworzący z procesu BPEL z modelu BPEL Designer graf aktywności głównego przebiegu procesu. Proces BPEL w notacji Eclipse BPEL Designer oraz jego odpowiednik w postaci grafu przedstawiono na rysunku Rys. 6.3.



Rys. 6.3 Proces BPEL z odpowiednikiem w postaci grafu na podstawie [6].

Model grafu reprezentuje strukturę procesu macierzystego, którego aktywności proste są reprezentowane jako pojedyncze węzły grafu. Aktywności złożone zostały zamodelowane w grafie jako pary węzłów – węzeł otwierający aktywność złożoną oraz węzeł zamykający. Model grafu jest tworzony na zasadzie iteracji po wszystkich elementach aktywności procesu BPEL i w zależności od złożoności elementu produkowany jest pojedynczy węzeł grafu lub ich para. Transformator trafiając na aktywność złożoną generuje węzły – otwierający oraz zamykający – po czym rekurencyjnie wywołuje metodę transformującą dla wszystkich aktywności znajdujących wewnątrz aktywności złożonej. W ten sposób powstaje model procesu w postaci grafu.

Zaprojektowany model grafu jest klasą implementującą interfejs modelu oraz dziedziczącą po klasie implementującej graf jak na diagramie klas przedstawionym na rysunku Rys. 6.4.



Rys. 6.4 Diagram klas modelu grafu.

Graf zaimplementowany na potrzeby projektu ze względu na specyfikę problemu – nie mają znaczenia odległości między wierzchołkami, a jedynie istnienie połączeń między nimi – posiada jedynie informację o połączeniach z innymi wierzchołkami. Graf reprezentowany jest przez klasę generyczną, czyli także reużywalną na potrzeby innych modeli bazujących na strukturze takiego grafu. Graf reprezentowany jest przez klasę zawierającą jedynie referencję węzła głównego grafu (*ang. root node*). Pojedynczy węzeł grafu zawiera obiekt modelowanego typu oraz stan węzła będący wartością enumeracyjną. Pojedynczy wierzchołek grafu może znajdować się w stanie nieodwiedzony (*UNVISITED*), procesowany (*PROCESSING*) oraz odwiedzony (*VISITED*). Klasa węzła grafu zawiera także dwie listy z referencjami do obiektów omawianej klasy. Pierwsza zawiera listę węzłów grafu bezpośrednio poprzedzających, natomiast druga, listę węzłów grafu będących bezpośrednimi następnikami dla danego wierzchołka.

Model właściwy grafu wykorzystywanego w dalszej części analizy rozszerza klasę generyczną *Graph<T>* z parametrem typu *org.eclipse.bpel.model.Activity*. W efekcie model złożony jest z węzłów zawierających jako element *data* aktywności składające się na transformowany proces BPEL.

6.2. Analiza grafu procesu.

Opis procesu analizy grafu procesu będącego wynikiem transformacji z postaci EMF.

6.2.1. Mechanizm dopasowania zmiennych.

6.3. Graficzny interfejs użytkownika.

Opis elementów graficznego interfejsu użytkownika dostarczonego pluginu.

6.4. Konfiguracja wtyczki (PDE).

=====

Opis konfiguracji wtyczki pod plugin Eclipse BPEL Designer – extension points.

=====

7. Testy.

=====				
Opis	przeprowadzonych	testów:	przebieg,	wyniki.
=====				

8. Podsumowanie.

=====

Ogólne	podsumowanie	projektu.
--------	--------------	-----------

=====

8.1. Napotkane problemy.

=====

Problemy,	z	którymi	stykano	się	podczas	realizacji	projektu.
-----------	---	---------	---------	-----	---------	------------	-----------

=====

8.2. Możliwości rozwoju.

=====

Dalsze perspektywy rozwoju projektu, jak można go rozwinąć, co można ulepszyć.
--

=====

9. Bibliografia.

- [1] <http://www.eclipse.org/bpel/>
- [2] Andrzej Ratkowski. Projektowanie transformacyjne procesów w architekturze usługowej, 2011.
- [3] <http://pic.dhe.ibm.com/infocenter/adiehelp/v5r1m1/index.jsp?topic=%2Fcom.ibm.etools.etc.bpel.doc%2Fsamples%2Ftravelbooking%2FtravelBooking.html>
- [4] <http://www.slideshare.net/milliger/eclipse-bpel-designer-presentation-765528>
- [5] Nick Boldt and Dave Steinber. Introduction to the Eclipse Modeling Framework, eclipseCON 2006.
- [6] Siran Chen. Extraction of BPEL Process Fragments in Eclipse BPEL Designer. Stuttgart, 2009.
- [7] Eric Freeman, Elisabeth Freeman, Kathy Sierra, Bert Bates. Head First. Design Patterns. 2004 O'Reilly Media, Inc.

10.Załączniki.

Bla bla bla.

10.1.Płyta CD.

Bla bla bla.

Dołączona płyta CD zawiera:

- Zestaw testowy przeznaczony dla kodera/dekoder
 - *.v – pliki poszczególnych modułów kodera/dekoder (Verilog)
 - *.mif – pliki inicjacyjne pamięci ROM modułów kodera/dekoder
 - Idcelp_encoder_tester.qar – archiwum projektu Quartus II zawierające zestaw testowy kodera
 - Idcelp_decoder_tester.qar – archiwum projektu Quartus II zawierające zestaw testowy dekodera
 - EncoderUSBReader.java – moduł programowy testera kodera odczytujący dane z portu USB i zapisujący je do pliku (Java)
 - DecoderUSBReader.java – moduł programowy testera dekodera odczytujący dane z portu USB i zapisujący je do pliku (Java)
 - jd2xx.jar – biblioteka procedur komunikacji z układem FT245BM (Java)
- Zestaw wektorów testowych wraz ze stanami wewnętrznymi
- Elektroniczną wersję pracy dyplomowej magisterskiej

10.2.Instrukcja instalacji wtyczki BPELag (BPEL assign generator).

Instrukcja