dr hab. inż. Jerzy Zając, prof. PK
mgr inż. Grzegorz Chwajoł
Politechnika Krakowska

# WYKORZYSTANIE TECHNOLOGII WEB SERVICES DO BUDOWY ROZPROSZONEGO SYSTEMU STEROWANIA

*Współczesne technologie internetowe otwierają nowe możliwości w budowie otwartych, rekonfigurowalnych systemów sterowania. W pracy przedstawiono wybrane elementy wieloagentowego systemu sterowania AIM opracowanego w Politechnice Krakowskiej. Komunikacja pomiędzy rozproszonymi elementami systemu sterowania realizowana jest przy użyciu technologii Web services.*

## BUILDING DISRIBUTED CONTROL SYSTEM USING WEB SERVICES TECHNOLOGY

*Contemporary Internet technologies open new opportunities for building open, reconfigurable control systems. The paper presents selected elements of multiagent control system AIM developed at Cracow University of Technology. The communication among distributed elements of control system is performed by means of Web services technology*.

## 1. WPROWADZENIE

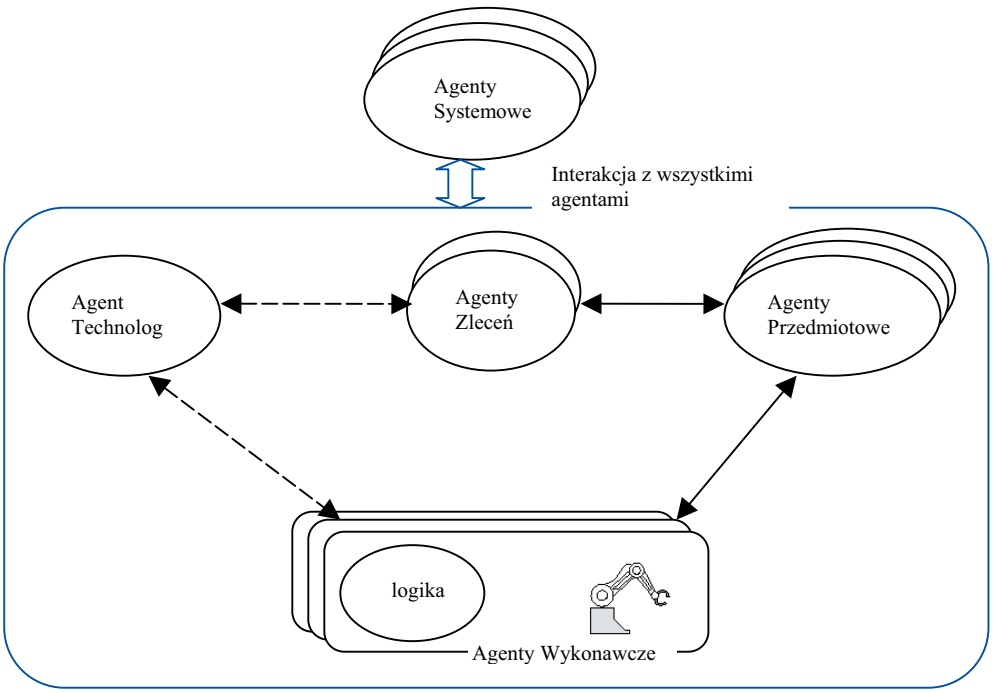
Otwartość i rekonfigurowalność to zasadnicze wymagania stawiane współczesnym systemom sterowania złożonych systemów produkcyjnych. Osiągnięcie tych wymagań jest obecnie możliwe dzięki rozwojowi technologii informacyjnych, zwłaszcza w zakresie standardów komunikacyjnych, wśród których dominują technologie internetowe, a także szeroka dostępność profesjonalnych pakietów aplikacji typu open-source gwarantujących niezależność na wciąż częściowo zmonopolizowanym rynku IT.

Szczególną rolę w budowie systemów sterowania odgrywają aktualnie technologie wieloagentowe. Wychodzą one naprzeciw dominującym trendom rozpraszania decyzji polegającym na przybliżaniu decydenta do miejsca decyzji, co docelowo prowadzić będzie do powstawania inteligenych urządzeń działających w systemach produkcyjnych zgodnie z zasadą *plug and play* [3]. Takie działanie wiąże się jednak z ograniczeniem dostępu do informacji o charakterze systemowym, co powoduje, że lokalny decydent musi uzyskiwać dodatkowe informacje, aby móc podejmować decyzje uwzględniające realizację celów systemowych. Odbywa się to w praktyce na dwa sposoby. Albo wykorzystuje się elementy wspólnego środowiska albo też stosuje bezpośrednią komunikację międzyagentową. Elementami wspólnego środowiska w systemie wieloagentowym są zazwyczaj scentralizowane lub rozproszone tablice ogłoszeń. Na takich tablicach agenty umieszczają informację o realizowanych procesach, zgłaszają swoje zapotrzebowania na zasoby itp. Są to więc miejsca, przez które odbywa się kontakt agentów - klientów poszukujących określonych usług i agentów – usługodawców, usługi te oferujących. Najbardziej znanym przykładem narzędzia umożliwiającego bezpośrednią komunikację międzyagentową jest protokół kontraktu sieciowego CNP (ang. Contract Net Protocol) [2]. Jest to protokół prowadzenia negocjacji wykorzystujący mechanizmy rynkowe. Istnieje również wiele różnych „mutacji" tego protokołu. Protokół ten jest często wykorzystywany także w procesach rozproszonego sterowania produkcją.

Zastosowanie technologii wieloagentowych do budowy rozproszonych systemów sterowania, przyjmując jednocześnie jako platformę implementacyjną technologię Web services, jedną ze znanych technologii internetowych, otwiera nowe możliwości budowy otwartych, rekonfigurowalnych systemów sterowania. W pracy przedstawiono ogólną strukturę wieloagentowego systemu sterowania AIM (*Agents Integrated Manufacturing*) opracowanego w Politechnice Krakowskiej i omówiono elementy technologii Web services wykorzystane do jego budowy.

## 2. WIELOAGENTOWY SYSTEM STEROWANIA AIM

Wieloagentowy system sterowania AIM zbudowany jest czterech typów uniwersalnych agentów oraz dwóch typów agentów dedykowanych. Agenty uniwersalne to agent zlecenia, agent systemowy, agent wykonawczy i agent przedmiotowy.




Rys. 1. Ogólna struktura wieloagentowego systemu AIM [5]

Agent zlecenia reprezentuje w systemie wykonywany typ produktu oraz przechowuje informacje dotyczące zamówienia, takie jak liczba sztuk produktu, termin i koszt jego wykonania oraz dokumentację konstrukcyjną zawierającą m.in. model CAD. Agent ten tworzony jest po wpłynięciu zlecenia, a usuwany albo po zakończeniu jego realizacji w przypadku przyjęcia zlecenia, lub też bezpośrednio po odrzuceniu zlecenia w przypadku jego nieprzyjęcia. Agent systemowy odpowiedzialny jest za działania o charakterze systemowym, tj. administrację systemu, nadzór i rejestracje agentów, a ponadto gromadzi informacje o bieżącym stanie systemu. Agent ten jest również aktywny w całym okresie działania systemu. Agent wykonawczy reprezentuje w systemie AIM fizyczne urządzenie takie jak maszyna, robot, magazyn itp. Agent ten odgrywa zasadniczą rolę w procesie decyzyjnym. Tworzony jest wraz z włączeniem (uaktywnieniem) fizycznego urządzenia, które reprezentuje, a usuwany z systemu po wyłączeniu tego urządzenia. Agent przedmiotowy reprezentuje zadanie wykonania pojedynczego produktu. Posiada informacje dotyczące marszrut technologicznych jego wykonania, a ponadto dysponuje bieżącymi informacjami o aktualnym stanie zaawansowania realiza-

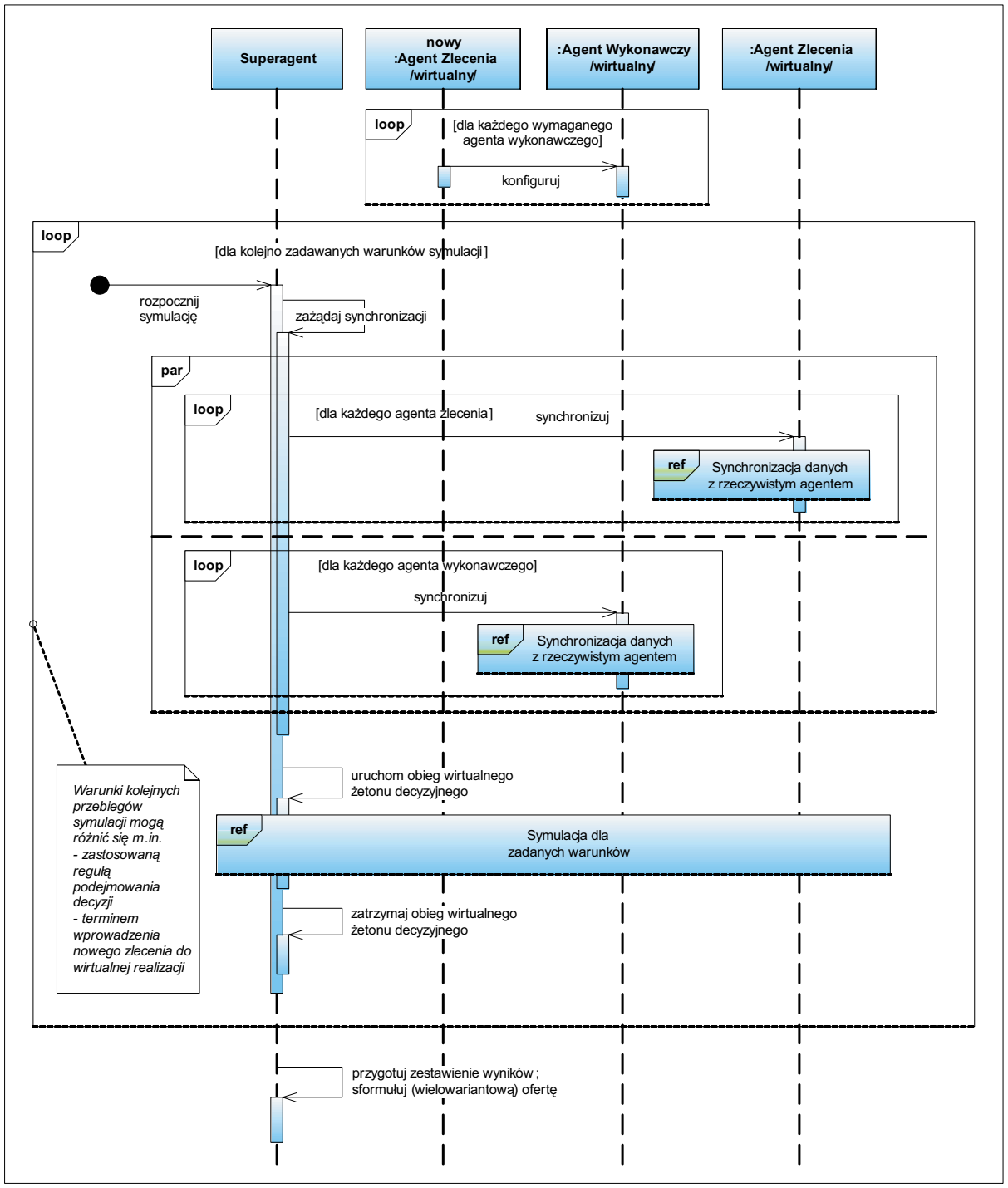
cji procesu. Agent ten tworzony jest w momencie wprowadzenia półfabrykatu do magazynu, a usuwany po wykonaniu gotowego produktu. Linie przerywane łączące agenty na rys. 1 wskazują na współdziałanie w procesie przygotowywania procesów technologicznych obróbki, natomiast linie ciągłe oznaczają współdziałanie w procesie sterowania produkcją.

Dwa typy agentów dedykowanych tworzą agent technolog i agent dostosowujący (pominięty na rys. 1). Agent technolog jest wyposażony w ekspercką wiedzę z zakresu projektowania procesów technologicznych, a jego zadaniem jest opracowywanie i modyfikacja procesów technologicznych dla przyjmowanych zamówień. Agent ten jest aktywny w całym okresie działania systemu AIM. Rola agentów dostosowujących sprowadza się najogólniej mówiąc do zapewnienia współpracy pomiędzy agentami wykonawczymi a sterownikami urządzeń.

Jedną z najciekawszych cech systemu AIM jest wprowadzenie mechanizmu symulacyjnego wykorzystującego tzw. wirtualne procesy wytwórcze [4]. Trzon omawianego mechanizmu stanowią wirtualne kopie (klony) agentów biorących udział w procesie sterowania operatywnego tj. wirtualne agenty wykonawcze oraz przedmiotowe, a także wirtualne agenty zleceń. Wymienione agenty współdziałają z ich rzeczywistymi odpowiednikami, a także w ograniczonym zakresie z pozostałymi elementami tworzącymi system sterowania. Za koordynację wirtualnych procesów odpowiedzialny jest tzw. superagent, tj. określony, uprzywilejowany agent wchodzący w skład grupy agentów systemowych, które w systemie pełnią role administracyjne i monitorujące. Jednym z zasadniczych celów zastosowania wspomnianego mechanizmu symulacyjnego jest weryfikacja możliwości przyjęcia nowego zlecenia.

Rozpoczęcie wirtualnego procesu wytwórczego w trybie nowego zlecenia poprzedzone jest w pierwszej kolejności weryfikacją możliwości technologicznych realizacji zamówienia. W przypadku pozytywnego wyniku weryfikacji, w następnej kolejności opracowany zostaje proces technologiczny. W efekcie, dla poszczególnych zasobów wytwórczych sporządzone zostają zbiory dodatkowych czynności elementarnych niezbędnych do realizacji rozpatrywanego zlecenia. Dopóki jednak warunki realizacji zamówienia nie zostaną zaakceptowane przez zleceniodawcę, dane te nie są przesyłane do agentów wykonawczych reprezentujących rzeczywiste zasoby wytwórcze. Są one jednak niezbędne w celu przeprowadzenia procesu symulacji. Jego rozpoczęcie wiąże się zatem z wcześniejszym dostarczeniem danych o nowych czynnościach oraz konfiguracją wszystkich agentów wirtualnie reprezentujących zasoby wytwórcze wymagane przez proces technologiczny. W celu uzyskania wielowariantowej oferty symulacje przeprowadzane są dla różnych terminów wprowadzenia zlecenia do realizacji. Uruchomienie mechanizmu symulacyjnego realizowane jest przez operatora przyjmującego zlecenie i każdorazowo rozpoczyna się od synchronizacji danych pomiędzy rzeczywistymi i wirtualnymi agentami wykonawczymi oraz agentami zleceń. Następnie inicjowany zostaje obieg wirtualnego żetonu decyzyjnego przyznającego uprawnienia decyzyjne poszczególnym agentom, co w konsekwencji pozwala na rozpoczęcie głównego etapu działań, jakim jest symulacja realizacji procesów wytwórczych. Algorytm działań symulujących sterowanie w podsystemie wirtualnym jest analogiczny do tego, który określa funkcjonowanie sterowania rzeczywistego, z jednym wyjątkiem dotyczącym sposobu sygnalizacji zakończenia poszczególnych czynności elementarnych. W systemie rzeczywistym zdarzenia te są sygnalizowane przez układy sterujące urządzeń biorących udział w realizacji czynności, zaś w procesach wirtualnych zakończenie poszczególnych czynności następuje po upływie oszacowanych uprzednio czasów ich trwania, przy czym czasy te są proporcjonalnie skrócone w stosunku do ich faktycznych wartości. Po zakończeniu realizacji wszystkich zadanych wa-

riantów symulacji przygotowywana jest oferta, która następnie zostaje przekazana zleceniodawcy. Istnieje także możliwość śledzenia przebiegu i częściowych wyników poszczególnych etapów symulacji w trakcie jej trwania. Proces symulacji w trybie nowego zlecenia zobrazowano na UML-owym diagramie przedstawionym na rys. 2.




Rys. 2. Symulacja w trybie nowego zlecenia

## 3. ZASTOSOWANIE TECHNOLOGII WEB SERVICES

Komunikacja pomiędzy rozproszonymi elementami systemu na wszystkich etapach sterowania, zarówno w trybie rzeczywistym jak i wirtualnym, realizowana jest przy użyciu technologii Web services [1]. Poszczególne kroki wymagane w procesie projektowania i implementacji mechanizmów wymiany informacji wykorzystujących powyższą technologię zostaną przedstawione w dalszej części rozdziału na przykładzie funkcji *startSimulation* wywoływanej na agencie systemowym, za pomocą której inicjowany jest proces symulacji opisanej w rozdziale 2.

Jednym z podejść do projektowania mechanizmów komunikacyjnych opartych na technologii Web services jest rozpoczęcie ich budowy od przygotowania dokumentu WSDL (ang. Web Services Description Language) opisującego w szczegółowy sposób funkcjonowanie mechanizmów. Podejście takie zapewnia najpełniejszą kontrolę nad regułami wymiany informacji i jest ono rekomendowane. Listing 1 przedstawia dokument (zgodny z wersją 1.1 standardu WSDL) definiujący reguły wymiany komunikatów w czasie wywoływania funkcji *startSimulation*, która jest metodą typu *void* (nie zwraca żadnego rezultatu swego działania) i przyjmuje jeden parametr określający tryb symulacji. Dokument ten zbudowany jest z pięciu zasadniczych typów elementów: *types*, *message*, *portType*, *binding* oraz *service*, które wchodzą w skład elementu nadrzędnego *definitions*. Wszystkie te elementy zdefiniowane są w przestrzeni nazw „*http://schemas.xmlsoap.org/wsdl/*" (ich nazwy poprzedza przedrostek *wsdl*, który jest niejako „skrótem" pełnej nazwy wspomnianej przestrzeni, co zostało określone poprzez atrybut: „*xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/*"). Znaczenie poszczególnych elementów jest następujące:

- element *types* zawiera deklaracje struktur danych wykorzystanych do definicji komunikatów. Struktury takie opisywane są najczęściej przy użyciu schematów XML (XML Schema). W omawianym przykładzie struktury te zostały zdefiniowane poprzez dwa elementy: *startSimulation* zawierający jeden parametr typu *string* o nazwie *workMode* oraz *startSimulationResponse* nie zwracający żadnego rezultatu. Omówiona wyżej symulacja w trybie nowego zlecenia realizowana jest dla parametru *workMode* przyjmującego wartość *virtual.*
- elementy *message* definiują komunikaty przesyłane w czasie wymiany informacji. W omawianym przykładzie określono dwa takie elementy: *startSimulationInputMessage* oraz *startSimulationOuputMessage*, a także powiązano z nimi definicje wprowadzono uprzednio w ramach elementu *types*.
- element *portType* tworzy interfejs zawierający określony zbiór udostępnianych operacji. Każda z operacji zdefiniowana jest w elemencie potomnym *operation*. W analizowanym, uproszczonym przykładzie element *portType* definiuje jedną operację o nazwie *startSimulation*. W odpowiadającym wymienionej operacji elemencie *operation* zdefiniowano z kolei który z określonych uprzednio komunikatów jest komunikatem wejściowym, a który wyjściowym. Taki układ komunikatów tworzy tzw. wzorzec wymiany komunikatów typu „żądanie-odpowiedź".
- element *binding* definiuje powiązanie elementu *portType* z konkretnym protokołem. W przedstawionym dokumencie WSDL element *binding* dokonuje powiązania w standardzie SOAP elementu o nazwie *SystemAgentServicePortType* z protokołem transportowym HTTP. Istotne znaczenie odgrywają także atrybuty *style* oraz *use*, które określają typ wiązania mający w efekcie wpływ na ostateczną postać przesyłanych pakietów danych.

Zdefiniowany w przykładzie typ *document/literal* jest najpopularniejszą spośród stosowanych kombinacją, która równocześnie jest rekomendowana w celu zapewnienia najwyższego poziomu interoperacyjności.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="SystemAgentService"
  targetNamespace="http://m6.mech.pk.edu.pl/SystemAgentService.wsdl"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://m6.mech.pk.edu.pl/SystemAgentService.wsdl"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsd1="http://m6.mech.pk.edu.pl/SystemAgentService.xsd1">
  <wsdl:types>
    <xsd:schema
      elementFormDefault="qualified"
      targetNamespace="http://m6.mech.pk.edu.pl/SystemAgentService.xsd1"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsd1="http://m6.mech.pk.edu.pl/SystemAgentService.xsd1">
      <xsd:complexType name="void">
        <xsd:sequence />
      </xsd:complexType>
      <xsd:element name="startSimulation">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="workMode" type="xsd:string" nillable="true" minOccurs="0" />
          </xsd:sequence>
        </xsd:complexType>
   </xsd:element>
      <xsd:element name="startSimulationResponse" type="xsd1:void" />
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="startSimulationInputMessage">
    <wsdl:part element="xsd1:startSimulation" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="startSimulationOutputMessage">
    <wsdl:part element="xsd1:startSimulationResponse" name="parameters"/>
  </wsdl:message>
  <wsdl:portType name="SystemAgentServicePortType">
    <wsdl:operation name="startSimulation">
      <wsdl:input message="tns:startSimulationInputMessage"/>
      <wsdl:output message="tns:startSimulationOutputMessage"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="SystemAgentServiceBinding" type="tns:SystemAgentServicePortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="startSimulation">
      <soap:operation soapAction="SystemAgentServicePortType#startSimulation"/>
        <wsdl:input>
          <soap:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="SystemAgentService">
    <wsdl:port binding="tns:SystemAgentServiceBinding" name="SystemAgentServicePort">
      <soap:address location="http://10.1.1.11:8080/axis2/services/SystemAgentService"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Listing 1. Dokument WSDL dla usługi SystemAgentService

- element service określa lokalizacje, pod jakimi osiągalne będą każde ze zdefiniowanych uprzednio wiązań. W prezentowanym przykładzie usługa o nazwie *SystemAgentService* udostępnia zdefiniowane wiązanie pod adresem:

  *http://10.1.1.11:8080/axis2/services/SystemAgentService*. Z tej informacji korzystać może oprogramowanie klienckie dedykowane danej usłudze. Warto jednak zauważyć, że wiele implementacji aplikacji klienckich umożliwia podawanie lokalizacji usług niezależnie od informacji zawartych w omawianym elemencie service, co bywa w określonych sytuacjach przydatne.

Na podstawie tak przygotowanego dokumentu WSDL można następnie za pomocą określonych narzędzi (np. WSDL2Java w środowisku Apache Axis2 lub wsdl.exe dostępnym w Microsoft .NET Framework) wygenerować kod pośredniczący (dla strony serwerowej lub klienckiej), który w dalszej kolejności należy zintegrować z właściwym kodem implementującym logikę agenta bądź wykorzystać w zewnętrznym oprogramowaniu klienckim.

Listing 2 przedstawia treść wymienianych w celu inicjalizacji symulacji XML-owych pakietów danych, których struktura wiąże się w sposób ścisły z definicją zawartą w dokumencie WSDL. Warto zwrócić uwagę, iż w przedstawionej sytuacji, mimo że wywoływana funkcja jest typu void, w odpowiedzi zwracany jest XML-owy pakiet. Wynika to z faktu zastosowania w definicji operacji zawartej w dokumencie WSDL wzorca wymiany komunikatów typu „żądanie-odpowiedź". Podejście takie pozwala w przeciwieństwie do potencjalnego zastosowania wzorca jednokierunkowego stwierdzić czy funkcja zakończyła swe działanie poprawnie.

***Zapytanie:***

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:sys="http://m6.mech.pk.edu.pl/SystemAgentService.xsd1">
  <soapenv:Header/>
  <soapenv:Body>
    <sys:startSimulation>
      <sys:workMode>virtual</sys:workMode>
    </sys:startSimulation>
  </soapenv:Body>
</soapenv:Envelope>
```

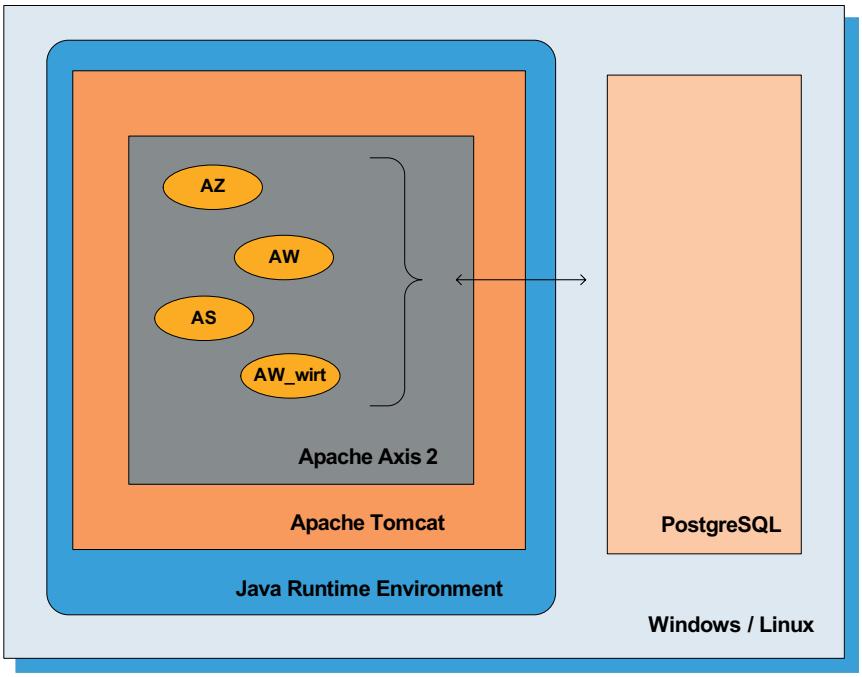
***Odpowiedź:***

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <startSimulationResponse xmlns="http://m6.mech.pk.edu.pl/SystemAgentService.xsd1"/>
  </soapenv:Body>
</soapenv:Envelope>
```

Listing 2. Dane w XML-owej postaci przesyłane w czasie uruchamiania symulacji

## 4. PLATFORMA IMPLEMENTACYJNA

Implementacja warstwy logiki systemu sterowania AIM została dokonana przy użyciu języka Java. Mechanizmy komunikacyjne wykorzystujące technologię Web services opracowano w oparciu o silnik Web services - Apache Axis2/Java. Aby poszczególne agenty wchodzące w skład systemu sterowania mogły być dostępne dla siebie oraz dla zewnętrznych aplikacji klienckich w postaci usług Web services, oprogramowanie agentów wraz z silnikiem Axis2 umieszczono w kontenerze aplikacji WWW, którego rolę pełni w omawianym środowisku serwer Apache Tomcat. Zarówno Axis2 jak i Tomcat należą do grupy aplikacji typu open source. Wykorzystywanym serwerem bazodanowym jest PostgreSQL, który w zależności od konfiguracji systemu może być uruchamiany centralnie lub też na każdej stacji roboczej. System sterowania AIM może być uruchamiany w środowisku MS Windows lub Linux, a także na innych systemach operacyjnych, dla których istnieje implementacja maszyny wirtualnej Java. Struktura środowiska uruchomieniowego została przedstawiona na rys. 3.




Rys. 3. Środowisko uruchomieniowe systemu AIM

Kliencka aplikacja do zarządzania systemem sterowania AIM została również zaimplementowana przy użyciu języka Java oraz silnika Apache Axis2/Java. Do stworzenia graficznego interfejsu użytkownika aplikacji wykorzystano bibliotekę Qt Jambi. Całość komunikacji pomiędzy aplikacją a systemem AIM realizowana jest w oparciu o technologię Web services. Listing 3 zawiera przykładowy kod implementujący żądanie rozpoczęcia symulacji. Na rys. 4 przedstawiono zrzut głównego okna aplikacji.

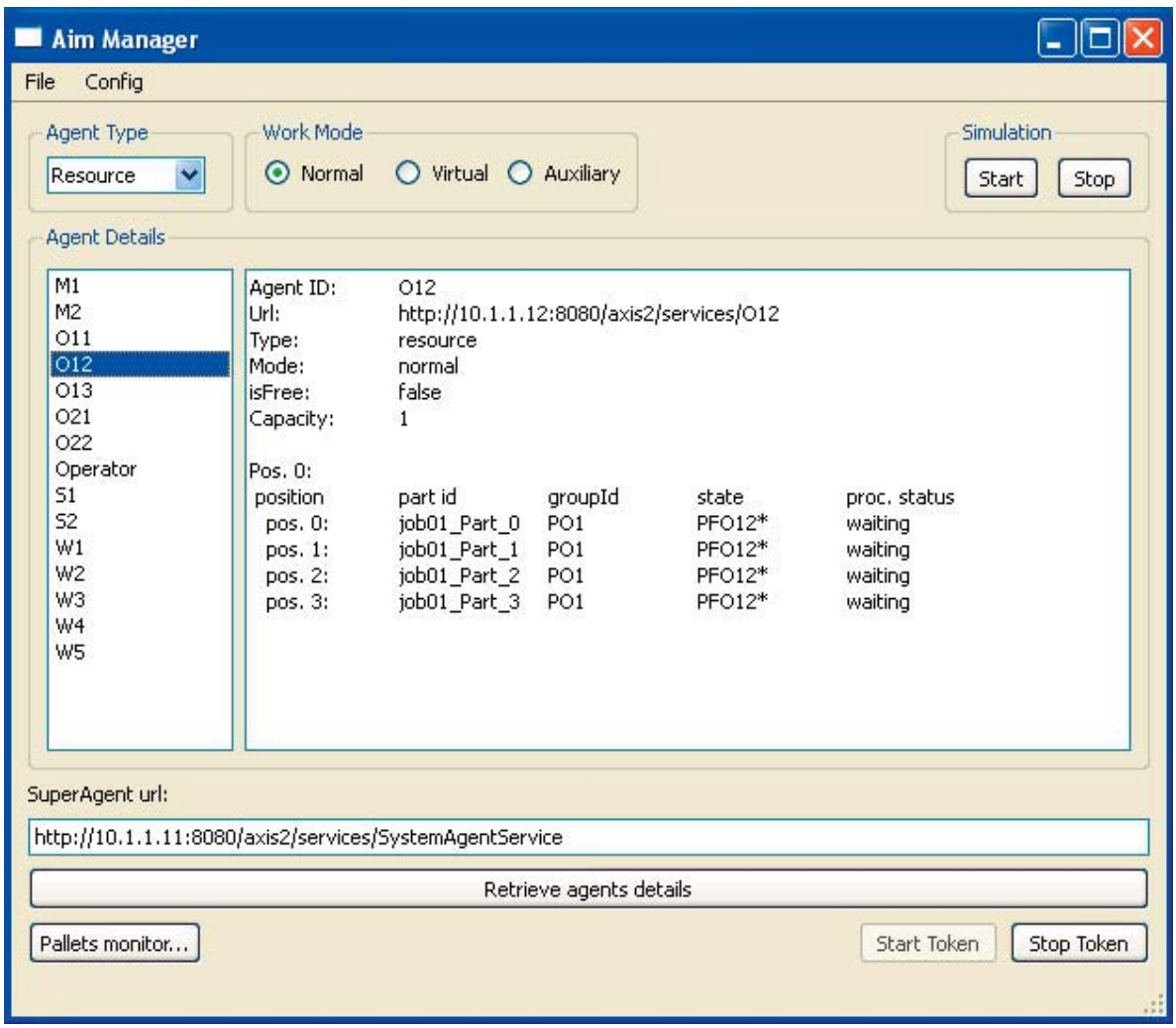
```
public void pushButtonStartSimulation() {
 try {
    // stwórz obiekt odpowiadający elementowi startSimulation zdefiniowanemu w WSDL
    StartSimulation req = StartSimulation.Factory.newInstance();
    req.setWorkMode(WorkMode.VIRTUAL.toString());

    // stwórz obiekt reprezentujący wysyłany komunikat
    StartSimulationDocument reqDoc = StartSimulationDocument.Factory.newInstance();
    reqDoc.setStartSimulation(req);

    // wywołaj metodę uruchamiającą symulację, przekaż jej obiekt z komunikatem
    SystemAgentServiceStub stub = new SystemAgentServiceStub(m_superAgentUrl);
    stub.startSimulation(reqDoc);

 } catch(RemoteException e) {
    Utils.showError("Error while starting simulation: " + e.getMessage());
 }
}
```

Listing 3. Kod implementujący żądanie rozpoczęcia symulacji (Java)




Rys. 4. Zrzut głównego okna aplikacji zarządzającej systemem sterowania AIM

Wykorzystując jeden z zasadniczych atutów technologii Web services, jakim jest relatywnie duży stopień niezależności od stosowanych języków programowania i platform systemowych można w łatwy sposób komunikować się z poszczególnymi elementami systemu AIM z poziomu modułów programowych zaimplementowanych nie tylko w języku Java. Listing 4 zawiera przykładowy kod implementujący żądanie rozpoczęcia symulacji napisany w języku JavaScript z wykorzystaniem skryptu WSRequest wchodzącego w skład oprogramowania WSO2 Web Services Framework/AJAX.

```
var req = new WSRequest();
var reqOptions = { useSOAP : true };
var url = "http://10.1.1.11:8080/axis2/services/SystemAgentService";
req.open(reqOptions, url, true);
var message = "<startSimulation xmlns=\"http://m6.mech.pk.edu.pl/SystemAgentService.xsd1\" />";
req.send(message);
```

Listing 4. Kod implementujący żądanie rozpoczęcia symulacji (JavaScript)

## 5. PODSUMOWANIE

Znacząca liczba współczesnych przemysłowych rozwiązań układów sterowania oferuje możliwość komunikacji za pomocą protokołów TCP/IP. Umożliwia to ich podłączanie bezpośrednio do Internetu lub zakładowego intranetu. Oznacza to także, iż producenci urządzeń automatyki przemysłowej widzą nowe możliwości dzięki wykorzystaniu potęgi Internetu. Jest to trend obserwowalny w bardzo wielu obszarach działalności człowieka. Stąd, wykorzystanie technologii internetowych w budowie rozproszonych systemów sterowania należy uznać za krok we właściwym kierunku. Technologie internetowe oprócz akceptowanego globalnie standardu komunikacyjnego jakim są protokoły TCP/IP stanowiły bazę do powstania znaczącej i ciągle rosnącej liczby profesjonalnych aplikacji typu open source, które pozwalają na budowę systemów oprogramowania niezależnych od sprzętu czy systemu operacyjnego.

## 6. LITERATURA

[1] FRYŹLEWICZ Z., SALAMON A., Podstawy architektury i technologii usług XML sieci WEB. Wydawnictwo Naukowe PWN SA, Warszawa 2008, ISBN 978-83-01-15371-7

[2] SMITH R.G., The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. IEEE Trans. on Computers, Vol. C-29, No. 12, 1980, s. 1104-1113

[3] ZAJĄC J., Rozproszone sterowanie zautomatyzowanymi systemami wytwarzania. Monografia 288, Seria Mechanika. Wydawnictwo Politechniki Krakowskiej, Kraków 2003.

[4] ZAJĄC J., CHWAJOŁ G., Wykorzystanie wirtualnych procesów sterowania produkcją do wspomagania decyzji w zautomatyzowanych systemach wytwarzania. Problemy Robotyki Tom II, Red. K. Tchoń, C. Zieliński. Oficyna Wydawnicza Politechniki Warszawskiej 2008, s. 625-634.

[5] ZAJĄC J., CHWAJOŁ G., KMIECIK A., Integracja projektowania procesów i sterowania produkcją w zautomatyzowanych systemach wytwarzania. Pomiary Automatyka Robotyka, 2/2007.