Symulacja w Badaniach i Rozwoju
Vol. 1, No. 2/2010

Dariusz PIERZCHAŁA
Wojskowa Akademia Techniczna, Wydział Cybernetyki,
ul. Kaliskiego 2, 00-908 Warszawa
E-mail: dariusz.pierzchala@wat.edu.pl

Metoda wymiany informacji między heterogenicznymi systemami symulacji oraz wspomagania decyzji w systemach reagowania kryzysowego

1 Wprowadzenie

Świadome podejmowanie trafnych decyzji w sytuacji zagrożenia kryzysem zależy nie tylko od wiarygodności i terminowości posiadanej informacji, ale również od możliwości przewidywania rozwoju sytuacji oraz weryfikowania decyzji podejmowanych przez decydentów. Do tego celu służą symulatory programowe różnego typu, systemy wspomagania podejmowania decyzji oraz narzędzia prognostyczne. Są to systemy autonomiczne lub współpracujące poprzez różne protokoły – typowym dla symulacji rozproszonej jest High Level Architecture (HLA), ale stosuje się także standardy przemysłowe (CORBA, RMI, DCOM). Oczekiwania przyszłych użytkowników systemów symulacyjnych ewoluują w kierunku architektury usług rozproszonych SOA (ang. Service Oriented Architecture), najsilniej wspieranej przez konsorcjum W3C. Autonomiczne symulatory wybranych działań lub sytuacji nie są już wystarczające a wymierne korzyści można uzyskać tylko wówczas, gdy funkcjonalność wspomnianych narzędzi zostanie wykorzystana łącznie (synergicznie). Z punktu widzenia technologii informatycznych sprowadza się to do spójnej i zrozumiałej wymiany informacji między systemami lub aplikacjami. Problemem jest jednakże brak przystosowania do interoperacyjnej współpracy ze sobą systemów. Jest to źródło nowych lub rozszerzonych wymagań dla systemów modelowania i symulacji.

Kolejnym krokiem na drodze integracji heterogenicznych systemów rozproszonych w sieci miejskiej lub rozległej jest albo aktualizacja założeń i rozwiązań konstrukcyjnych (np. standardu HLA) albo opracowanie pomostów pomiędzy standardowymi protokołami. Uzasadnia to potrzebę opracowania metody wymiany informacji między heterogenicznymi systemami symulacji oraz wspomagania decyzji. Proponowana metoda dotyczy wymiany informacji pomiędzy architekturami HLA i SOA i zakłada wykorzystanie Usługi Sieciowej dostępnej poprzez sieć lokalną lub Internet. Usługa Sieciowa spełnia założenia architektury SOA a z drugiej strony komunikuje się z lokalnym oprogramowaniem RTI, które jest implementacją architektury HLA.

Praktyczne wyniki znajdą zastosowanie w projektach badawczych: PBZ-MNiSW-DBO-02/I/2007 pt. „Zaawansowane metody i techniki tworzenia świadomości sytuacyjnej w działaniach sieciocentrycznych" na potrzeby systemów reagowania kryzysowego oraz GD-WAT 675/2009 pt.: „Metoda wymiany informacji między heterogenicznymi

systemami symulacji oraz wspomagania decyzji w systemach reagowania kryzysowego o architekturze SOA".

2 Informatyczne wspomaganie decyzji

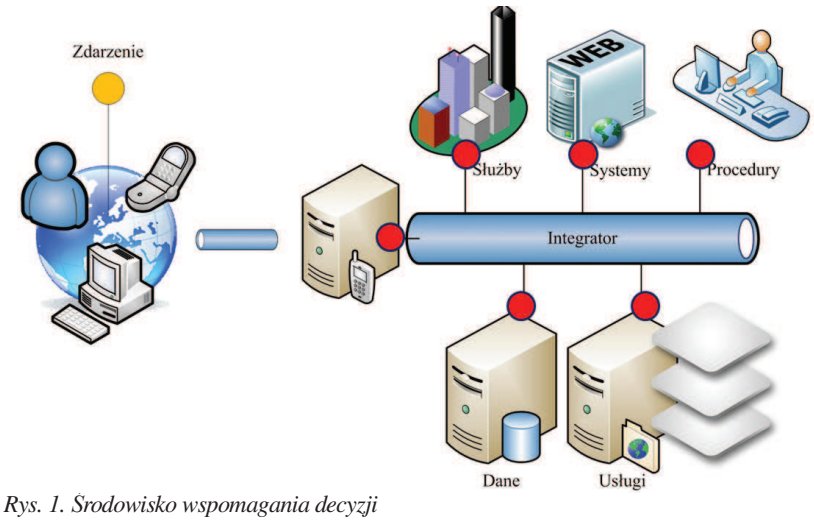
W powszechnym rozumieniu *decyzja* definiowana jest jako wybór, prowadzący do szczególnego działania. Natomiast *wspomaganie decyzji* to proces opracowania informacji niezbędnej do jej podjęcia – w podstawowym ujęciu zawiera następujące składowe [3]:

- listę możliwych do wyboru działań rozważanych w procesie decyzji;
- listę możliwych skutków wynikających z określonych działań;
- dane opisujące wszystkie możliwe kombinacje działań i ich skutków;
- oszacowania prawdopodobieństwa zaistnienia możliwych skutków;
- kryteria decyzji, które podpowiadają, jakie podjąć działania.

Zatem *informatyczne systemy wspomagania decyzji* to aplikacje lub usługi programowe, które dostarczają decydentowi odpowiednio przygotowanej informacji, stanowiącej podstawę do podjęcia decyzji, poprzez:

- gromadzenie, integrowanie i przetwarzanie dużej ilości danych;
- rozwiązywanie zadań optymalizacji, harmonogramowania, prognozowania, symulacji itp.;
- prezentowanie użytkownikowi danych podstawowych, agregatów, oszacowań i innych wyników (np. rozwiązań zadań optymalizacji);
- sterowanie przepływem czynności i obiegiem dokumentów w proceduralnych działaniach zgodnie z podjętymi decyzjami.

Typowe środowisko integrujące systemy wspomagania decyzji, ludzi i odpowiednie służby wykonawcze koncepcyjnie prezentuje Rys. 1 [5].



Rys. 1. Środowisko wspomagania decyzji
Fig. 1. Decision support environment

Realizacja powyższych zadań wymaga wielu złożonych podsystemów (modułów) informatycznych, w tym symulatorów i narzędzi prognostycznych. W obliczu zidentyfikowanych potrzeb i oczekiwań decydentów, wielkiej liczby użytkowanych systemów spadkowych (ang. legacy systems) oraz dynamicznie rozwijających się technologii i standardowych protokołów komunikacji niezbędna jest warstwa odpowiadająca za integrację rozproszonych podsystemów (modułów, komponentów oraz samodzielnych programów). Jej podstawowym celem jest zapewnienie wszystkim podsystemom możliwości działania oraz spójnej semantycznie wymiany komunikatów a decydentowi udostępnienie synergicznej funkcjonalności – większej, niż w przypadku niezależnego korzystania z poszczególnych podsystemów.

3 Integrator podsystemów wspomagania decyzji i symulacji

W skalowalnych systemach wspomagania decyzji, dla których konfiguracje można zestawiać w zależności od potrzeb użytkownika, integrowane są podsystemy różnych klas – w zarządzaniu kryzysowym są nimi sensory i narzędzia monitorujące, bazy danych, moduły prognozowania i symulacji oraz wspomagania decyzji, interfejsy użytkownika oraz zobrazowania mapowego, komponenty łączności telekomunikacyjnej. Każda z tych klas charakteryzowana jest innymi wymaganiami funkcjonalnymi a w efekcie różnymi interfejsami wymiany danych. Zakres integracji obejmuje zatem następujące grupy wymagań [6]:

- dynamiczna rejestracja usług podsystemów różnych klas;
- asynchroniczna i synchroniczna komunikacja pomiędzy składowymi systemu;
- reagowanie na powiadomienia (komunikaty) o zmianie stanu elementów otoczenia;
- cykliczna aktualizacja danych o stanie otoczenia oraz infrastruktury, np. sytuacji meteo;
- bezpośredni dostęp do informacji o zdarzeniach, stanie otoczenia oraz wiedzy eksperckiej;
- identyfikacja i klasyfikacja zdarzeń prostych oraz mieszanych;
- dobór usług oraz zarządzanie ich wywoływaniem w procesie obsługi zdarzeń;
- wyszukanie oraz analiza związków (możliwych następstw) – np. pożar może spowodować wybuch gazu i/lub wstrzymanie ruchu drogowego w pobliskim otoczeniu;
- konwersja wymienianych danych do ujednoliconych struktur komunikatów;
- archiwizowanie danych o stanach, zdarzeniach i sekwencjach działań;
- zapewnienie jakości usług.

Metody integracji wypracowane przez lata ewoluowały od operacji na plikach przez bazy danych aż do dynamicznych wywołań klient-serwer:

- wsadowa wymiana plików (nadawca eksportuje do pliku, odbiorca czyta z pliku) – lata 70-te;
- współdzielenie centralnej bazy danych – lata 80 - 90-te;
- zdalne wołanie procedur – lata 90-te;
- żądanie – odpowiedź – współcześnie.

Aktualne trendy oscylują wokół programowych rozwiązań określanych mianem magistrali danych ESB – (ang. Enterprise Service Bus). Do podkreślanych zalet należą:

- skalowanie,
- luźne powiązania,
- bezpieczeństwo,
- rutowanie komunikatów,
- transformacja danych.

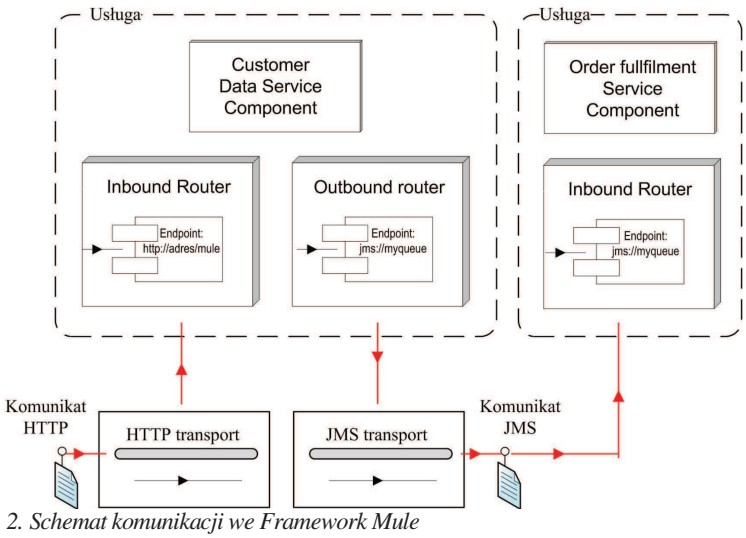
Za wyraźne wady wciąż muszą być uznawane:

- niska wydajność,
- problemy z komunikacją synchroniczną.

Wśród wielu rozwiązań ESB funkcjonujących na rynku systemów integracyjnych woróżnia się Framework Mule. Jest to platforma integracyjna klasy „open source", używana przez m.in.: HP, Sony, SyphonySoft, Deutsche Bank, Citi Bank. Realizuje koncepcję magistrali usług ESB poprzez następujące usługi:

- transformacji i rutowania komunikatów,
- centralnego zarządzania systemem rozproszonym,
- bezpieczeństwo i autoryzacja.

Udostępniono w niej kontenery usług, możliwa jest współpraca z popularnym frameworkiem Spring. Dzięki temu m.in. wspiera komponenty typu POJO. Adaptuje bogaty zestaw technologii i standardowych protokołów komunikacyjnych: HTTP, FTP, JDBC, XML, JMS, Axis, SOAP, Spring, BPEL i stosuje wzorce„Enterprise Integration Patterns". Jądrem Mule jest szyna komunikatów, przekazująca dane pomiędzy stronami za pośrednictwem szeregu definiowalnych usług (rys. 2).



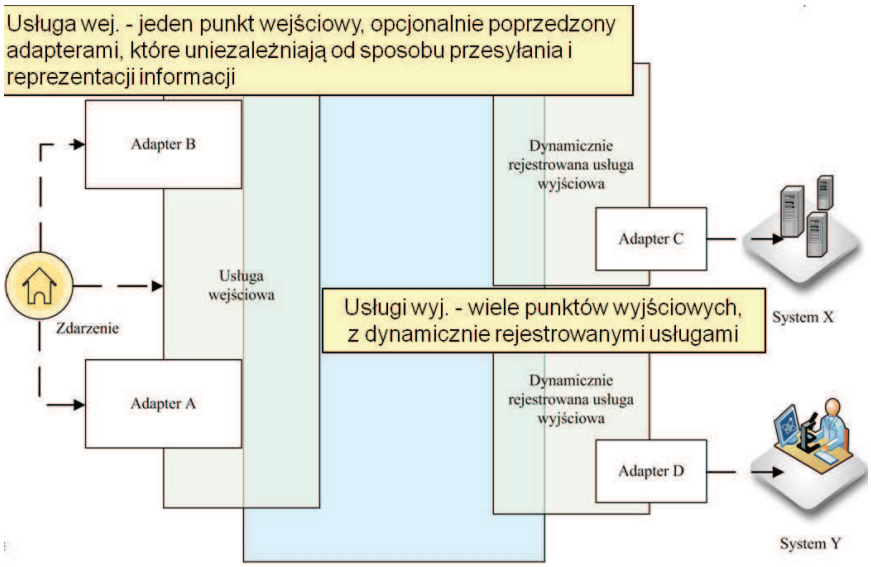
Rys. 2. Schemat komunikacji we Framework Mule

Fig. 2. Communication schema in Framework Mule

Kluczową rolę odgrywają usługi Customer Data Service Component, które implementują logikę przetwarzania komunikatów, odizolowaną od zadań związanych z otrzymywaniem i wysyłaniem danych. Zadania obsługi wejścia (filtrowanie, agregowanie, kolejkowanie) i wyjścia (adresowanie, wysyłanie) przypisane są komponentom Inbound Router oraz Outbound Router.

W architekturze modułu integratora komponenty Framework Mule zastosowano w następujący sposób:

- WorkflowManager – realizowany przez opracowany od podstaw komponent CustomCrisisInboundChainRouter (odpowiednik Mule Inbound Router);
- Message Router – realizowany przez komponent Mule Client (służy do podłączenia do uruchomionych w ramach platformy usług);
- klasy obiektów związane z przetwarzaniem komunikatów, tj.: MessageReceiver, MessageProcessor, MessageTransformator, MessageDispatcher bezpośrednio odpowiadają klasom proponowanym przez Framework Mule.



Rys. 3. Komunikacja z systemami zewnętrznymi
Fig. 3. Communication with external systems

W proponowanym rozwiązaniu, które wykorzystuje kluczowe elementy Mule, przepływ informacji ze źródła do odbiorcy obejmuje system w opisanym dalej schemacie. Informacja o zdarzeniu (zmianie stanu elementu otoczenia lub infrastruktury krytycznej), pochodząca z sensorów lub innych źródeł, trafia do modułu integratora, gdzie podlega analizie, przetworzeniu i obsłudze. Schemat obsługi otrzymanych danych wiąże w sekwencji przepływu prac (ang. workflow) następujące czynności:

- identyfikacja (rozpoznanie) – odebranie informacji o zdarzeniu, określenie typu, wyszukiwanie dodatkowych danych zależnych od typu zdarzenia (np. stan pogody, opis terenu, charakterystyka rozmieszczenia źródeł zagrożenia) oraz zapamiętanych historycznych opisów podobnych sytuacji;
- korelacja – wykrywanie związków czasowo–przestrzennych pomiędzy zidentyfikowanym zdarzeniem a innymi, które wystąpiły w pobliżu – efektem mogą być związki przyczynowo – skutkowe zdarzeń;
- przewidywanie – predykcja możliwych zdarzeń wtórnych, będących skutkiem (negatywnym następstwem) zidentyfikowanego zdarzenia, m.in. na podstawie macierzy incydencji zdarzeń definiującej ich wzajemny wpływ;
- dedukcja – ekspercki dobór usług programowych dla procedury reagowania na zdarzenie na podstawie informacji o zdarzeniu, stanie otoczenia, możliwych skutkach oraz zapamiętanych w systemie realizacjach podobnych sytuacji;
- weryfikacja – weryfikacja dobranych usług procedur reagowania;
- wykonanie – wywołanie usług, np. przekazanie poleceń do jednostek ratownictwa, wysłanie odpowiedzi do systemów źródłowych, wypracowanie wariantów decyzji.

Komunikacja z systemami zewnętrznymi wobec modułu integratora odbywa się za pośrednictwem portów wejściowych oraz wyjściowych. System posiada jeden port wejściowy, opcjonalnie poprzedzony adapterami oraz wiele portów wyjściowych, odpowiadających różnym systemom lub usługom zewnętrznym. Zadaniem adapterów jest konwersja danych napływających do integratora na ujednolicony format (zaadaptowano standard TSO promowany przez konsorcjum OASIS). Punkty wyjściowe do zewnętrznych systemów są usługami dynamicznie rejestrowanymi w integratorze. Alternatywą jest dynamiczna rejestracja adapterów konwertujących dane z systemów zewnętrznych na przyjęty format.

Jakość usług (ang. QoS – Quality of Service) rozumiana jest przede wszystkim jako zabezpieczenie przed utratą integralności danych przetwarzanych w module integratora na skutek niedostępności usług zewnętrznych lub awarii samego integratora. W tym celu, na każdym etapie przetwarzania, opis zdarzenia oraz stan przepływu pracy zapisywane są w bazie danych. Kolejnym aspektem QoS jest transakcyjność – każdy napływający do systemu komunikat może powodować rozpoczęcie nowej transakcji. W przypadku niepowodzenia przetwarzania informacji, wszystkie wcześniejsze czynności zostaną wycofane a system będzie w stanie sprzed otrzymania komunikatu.

4 Usługi symulacyjne

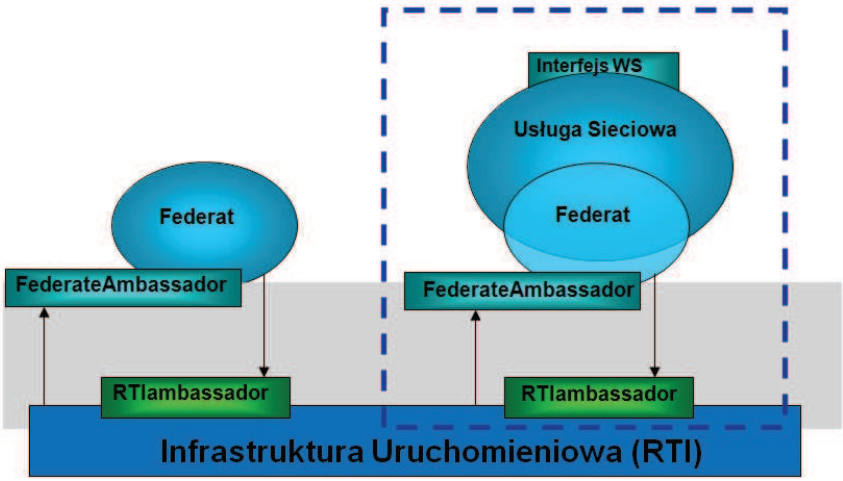
Zadania symulacyjne, których wyniki wykorzystane są przez decydentów do weryfikacji decyzji, prognozowania przebiegu zjawisk lub symulacyjnej oceny wariantów, mogą być realizowane w symulatorach autonomicznych lub rozproszonych. Ewolucja technik eksperymentowania z użyciem symulatorów implikuje różne podejścia do projektowania usług symulacyjnych. Przedmiotem zainteresowania w opracowaniu jest symulacja konstruktywna, bazująca na formalnych modelach i realizowana z udziałem operatorów. Docelowo eksperyment symulacyjny może być realizowany jako [4]:

- sekwencyjny (ang. serial algorithms),

- równoległy (ang. parallel discrete event simulation),
- rozproszony (ang. parallel and distributed simulation),
- interaktywny i rozproszony (ang. distributed interactive simulation).

Komunikację między elementami środowiska symulacyjnego zapewnia najczęściej standard High Level Architecture (w wersji wojskowej DoD HLA 1.3 a w cywilnej IEEE 1516). Środowiska symulacyjne zgodne z HLA, czyli Federacje, złożone są z federatów, wymieniających pomiędzy sobą dane: interakcje lub uaktualnienia atrybutów obiektów. Warunkiem koniecznym w procesie wymiany danych jest utworzenie dla nich specyficznych deklaracji w dedykowanych obiektowych modelach federacji (ang. FOM – Federation Object Model).

Rdzeniem rozwiązania integrującego Federacje symulacyjne z systemami wspomagania decyzji jest usługa sieciowa, która z jednej strony spełnia założenia SOA a z drugiej komunikuje się z oprogramowaniem RTI, które implementuje standard HLA. Koncepcję metody oraz podstawowe komponenty podsystemu prezentuje rys. 4 [7]. Dla zachowania stanowości usługi oraz zapewnienia dostępu do współdzielonych zmiennych, zdalny dostęp do RTI realizowany jest przez jedną usługę sieciową, która dynamicznie tworzy instancje klas, umożliwiające komunikację z RTI oraz przetwarzanie modelu FOM.



Rys. 4. Pomost HLA-WebService.
Fig. 4. HLA-WebService gateway.

Interfejs usługi, opisany plikiem WSDL, jest dostępny zdalnie. Usługa jest autonomiczna i tym samym realizuje zasadę reużywalności. Dzięki możliwościom wywoływania funkcji federatów usługa posiada kontrolę nad tym, co dzieje się w RTI a zarazem udostępnia na zewnątrz swój interfejs, odpowiadający zdefiniowanym wymaganiom funkcjonalnym, podzielonym na pięć grup:

- zarządzające (federacjami, deklaracjami, obiektami), np. CreateFederation;
- przetwarzające FOM, np. GetObjects;

- pośredniczące w symulacji, np. RegisterObject, UpdateAttributeValues, AdvanceTime;
- generujące szablony klas, np. FunctionCallClassGen;
- pomocnicze, np. UploadFEDFile.

5 Podsumowanie

Podstawowym celem integracji jest, z punktu widzenia użytkownika, uzyskanie jednolitego systemu, choć złożonego z rozproszonych i heterogenicznych podsystemów. Integracja systemów musi być zatem zrealizowana na wszystkich poziomach: technicznym, syntaktycznym oraz semantycznym. Nakład pracy na budowę warstwy integrującej systemy jest dodatkowym kosztem, a zyski osiąga się dopiero wówczas, gdy integracji poddane jest wiele systemów. Integracja ułatwia tworzenie aplikacji, co wynika z uniezależnienia od stosowanych technologii, sprzętu i protokołów.

Zaprezentowane rozwiązanie umożliwia wykorzystanie systemów nowoprojektowanych oraz spadkowych. Otwarta architektura systemu opartego na usługach gwarantuje możliwość rozszerzania jego funkcjonalności, dynamiczne rekonfigurowanie oraz wdrażanie systemu etapami. Framework Mule jest wystarczający do realizacji tak rozumianego integratora. Jednakże nietrywialne są zagadnienia dynamicznego przekształcanie typów danych, zapewnienia QoS przy jednocześnie wystarczającej efektywności (utrwalanie danych, transakcyjność a czas obsługi komunikatu) oraz ekspercki dobór właściwych usług sieciowych a następnie ich wyszukanie.

Literatura

1. Simulation Interoperability Standards Committee – *Grupa standardów IEEE Std 1516.1- 1516.4*, ISBN 0-7381-2621-7, ISBN 0-7381-2623-3, ISBN 0-7381-2619-5, ISBN 0-7381-3584-4, 2000-2004
2. Kuhl F., Weatherly D., Dahman J.: *Creating Computer Simulation Systems- An Introduction To The High Level Architecture*, PH PTR, ISBN 0-13-022511-8, 1999
3. Najgebauer A.: *Informatyczne systemy wspomagania decyzji w sytuacjach konfliktowych. Modele, metody i środowiska symulacji interaktywnej*, ISBN 83-908620-6-9, WAT, 1999
4. Pierzchała D.: *Metoda wspomagania projektowania i badania właściwości środowisk interaktywnej symulacji rozproszonej do realizacji gier operacyjnych, WAT*, Warszawa, 2002 (rozprawa doktorska)
5. Pierzchała D., Niewiński K., Ząbecki B., Zdziarski P.: *Integracja rozproszonych systemów prognozowania, symulacji i wspomagania decyzji*, rozdział w monografii A. Najgebauer (red.), Modele zagrożeń aglomeracji miejskiej wraz z systemem zarządzania kryzysowego na przykładzie m. st. Warszawy, ISBN 978-83-61486-22-0, Warszawa, 2009
6. Pierzchała D.: *Integracja rozproszonych systemów symulacji oraz wspomagania decyzji*, Przegląd Telekomunikacyjny i Wiadomości Telekomunikacyjnych, Zeszyt 8-9/2009, ISSN 1230-3496, 2009
7. Pierzchała D., Plewa R.: *Metoda wymiany informacji między systemami symulacji o architekturze HLA z wykorzystaniem mechanizmów SOA w systemach wspomagania decyzji*, materiały konferencji *Wsparcie teleinformatyczne dowództw w działaniach wojsk lądowych*, AON, 2008

Streszczenie

Opracowanie prezentuje propozycję integratora heterogenicznych systemów rozproszonych wspomagania decyzji i symulacji, w tym zgodnych ze standardem HLA. Proponowana metoda zakłada wykorzystanie Usługi Sieciowej dostępnej poprzez sieć lokalną lub Internet. Jako platformę integracyjną wykorzystano Framework Mule, realizujący koncepcję magistrali ESB.

The method of data exchanging between simulation and decision support modules in crisis management systems

Summary

The article presents the framework and method of data exchanging between simulation and decision support modules in crisis management systems. The attention is focused on IEEE 1516 (High Level Architecture - HLA). This work also supports reusability of legacy simulations by new direction: HLA integrated with SOA (Service Oriented Architecture). The framework Mule has been used as an integrator.