Symulacja w Badaniach i Rozwoju
Vol. 2, No. 3/2011

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

Adaptowalny integrator symulatorów różnej rozdzielczości w architekturze HLA

1 Wstęp

Podstawowym celem prac w zakresie integracji systemów symulacyjnych różnej rozdzielczości w oprogramowaniu Runtime Infrastructure standardu High Level Architecture (HLA) jest umożliwienie prowadzenia wielorozdzielczej symulacji rozproszonej w spójnej czasoprzestrzeni modelowanego pola działań bojowych. Dekompozycja złożonych systemów symulacyjnych zmusiła projektantów do poszukania wydajnego narzędzia potrafiącego sprawnie zarządzać wymianą danych między rozproszonymi komponentami. Architektura HLA jest znanym standardem, powstałym by sprostać rosnącym wymaganiom osób odpowiedzialnych za przygotowanie eksperymentów symulacyjnych. Ideą HLA jest, aby aplikacje programowe (symulatory komputerowe) mogły porozumiewać się między sobą niezależnie od tego, na jakiej platformie zostały osadzone. HLA daje możliwość łączenia wielu oddzielnych komputerowych symulacji, które stanowią re-używalne komponenty, w jedną większą i bardziej rozbudowaną całość. Kluczowe pojęcia nieodzownie związane z terminem eksperymentu symulacyjnego w tej architekturze to:

6 Federacja – złożony system symulacyjny;
7 Federat – pojedynczy symulator wchodzący w skład federacji;
8 RTI (ang. Runtime infrastructure) – warstwa nadzorująca przebieg symulacji, odpowiedzialna za zarządzanie komunikacją w obrębie eksperymentu;
9 FOM (ang. Federation Object Model) – model danych opisujący wymieniane dane.

HLA jest architekturą warstwową opartą na zdarzeniach. Warstwą najniższą jest RTI, zgodnie z założeniami, że każda warstwa dostarcza usług warstwie znajdującej się powyżej RTI dostarcza usługi federacji. Cały mechanizm takiej komunikacji jest zaszyty głęboko wewnątrz RTI i nie jest widoczny z poziomu federatów. Taki zabieg zapewnia komfort wprowadzania zmian, umożliwia niezależną rozbudowę symulatorów, pozwala w teorii na łączenie aplikacji różniących się pod względem architektury i zasad działania. Specyfikacja danych symulacyjnych opisuje strukturę każdego obiektu i interakcji wymienianej w trakcie trwania eksperymentu.

- Interakcje – są to dane nietrwałe, zwykle oznaczające wystąpienie jakiegoś zjawiska lub zdarzenia, wysyłane poprzez RTI do jednego lub wielu zainteresowanych federatów;

- Obiekty – odnoszą się do symulowanych persystentnych encji, które są w kręgu zainteresowań innych federatów.

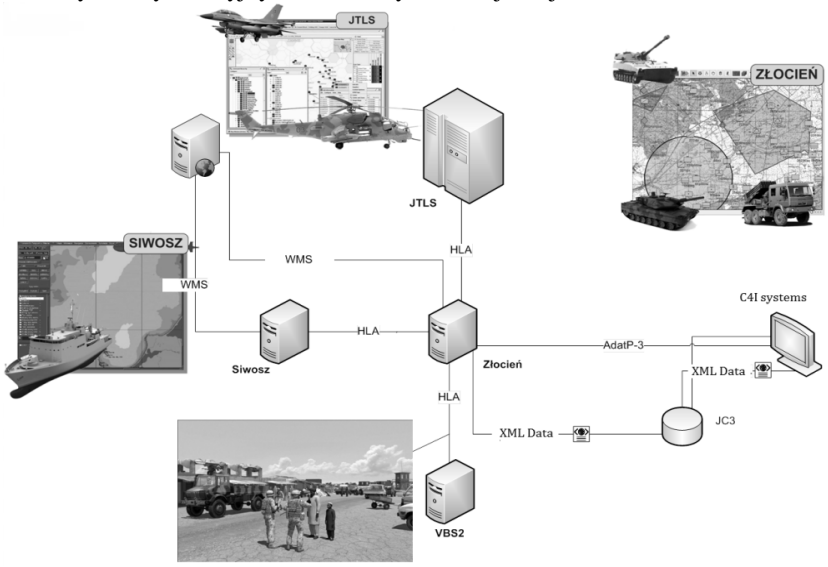
Plik FOM z opisem struktury tych danych stanowi wykaz akceptowalnych i obsługiwanych komunikatów, których zrozumienie i obsługa jest wymagane od każdego federata chcącego dołączyć, jako pełnoprawny uczestnik symulacji. Wiele współczesnych systemów rozproszonych, używanych do szkolenia z zakresu prowadzenia działań bojowych, pozwala prowadzić eksperymenty z wykorzystaniem jednego lub kilku systemów symulacyjnych. Przy okazji integracji odrębnych środowisk, projektanci zostaną skonfrontowani z problemem modeli o odrębnych strukturach i rozdzielczościach danych.

2 Środowisko symulacji rozproszonej

Łączenie symulatorów (dwóch lub więcej) polega na ich przystosowaniu do pracy w środowisku symulacji rozproszonej, w ramach której dane udostępniane przez jeden symulator nie zawsze muszą być rozumiane z poziomu pozostałych. Problem sprowadza się do stworzeniu wspólnego wirtualnego środowiska skupiającego aplikację na różnych poziomach szczegółowości.

Idea adaptowalnego integratora zrodziła się podczas badań nad skonstruowaniem wspólnego środowiska rozproszonego dla trzech symulatorów o różnym poziomach szczegółowości. W ich skład wchodzą:

- System symulacyjnego wspomagania szkolenia operacyjnego: „Złocień”;
- System symulacyjny działań połączonych: JTLS (Joint Theater Level Simulation);
- System symulacyjny działań marynarki wojennej: "Siwosz".



Rys. 1. Środowisko symulacji rozproszonej: „Złocień”, „Siwosz”, „JTLS”
Fig. 1. Distributed simulation environment: „Zlocien”, „Siwosz”, „JTLS”

Wymienione systemy są interaktywnymi narzędziami symulacyjnymi pozwalającymi na modelowanie i przeprowadzanie symulacji działań wojsk lądowych, morskich i powietrznych w ramach ćwiczeń wspomaganych komputerowo CAX (ang. Computer Assisted Exercises). Celem połączenia trzech symulatorów jest umożliwienie dowódcom doskonalenia umiejętności w niespotykanej jak dotąd skali. Każdy z powyższych symulatorów osobno dostarcza kilka unikatowych cech, jednak z racji swojego przeznaczenia mogą nieść kilka ograniczeń, których likwidacja możliwa jest przez ich wzajemne uzupełnianie. Operatorzy w ramach każdego systemu mają do czynienia z różnymi szczeblami dowodzenia, idealna sytuacja ma miejsce, gdy odbiorcom oddane jest narzędzie, pozwalające na dowodzenie pojedynczym żołnierzem. W rozpatrywanym systemie symulatorów taka sytuacja nie miała miejsca, jednak na potrzeby wizualizacji działań na dużym poziomie szczegółowości rozpatrywany był pomysł użycia konsoli zobrazowania w postaci symulatora pola walki VBS2 (ang. Virtual Battlespace Simulator).

Z każdym systemem dostarczane są narzędzia wspomagające przygotowanie ćwiczeń (poczynając od opracowania scenariusza działań), umożliwiające przeprowadzenie ćwiczenia w oparciu o scenariusz oraz pozwalające na analizę wyników symulacji. Ich celem jest usprawnić późniejszy proces planowania i prowadzenia ćwiczeń przy użyciu rzeczywistych środków oraz przyczynić się do zwiększenia jakości szkolenia dowódców. Modele walki zaimplementowane w systemach symulacyjnych uwzględniają różne struktury jednostek i różne działania. W ramach każdego symulatora może być mowa m.in. o różnym wyposażeniu, odwzorowaniu sił i środków, systemie walki, zasięgu prowadzenia ognia, potencjale jednostek. Takie zróżnicowanie opisu wirtualnego pola walki jest dużą przeszkodą w opracowaniu wspólnego środowiska dla symulacji rozproszonej z uwzględnieniem wspomnianych symulatorów. Umożliwienie współpracy w ramach jednego środowiska wymaga przede wszystkim ujednolicenia modelu danych symulacyjnych. W sytuacji jednak, gdy jeden symulator ze względu na poziom zagregowania, wynikający z poziomu szczegółowości symulowanego pola walki, nie pozwala na użycie konkretnego typu danych, konieczne jest użycie mechanizmów agregacji i deagregacji danych.

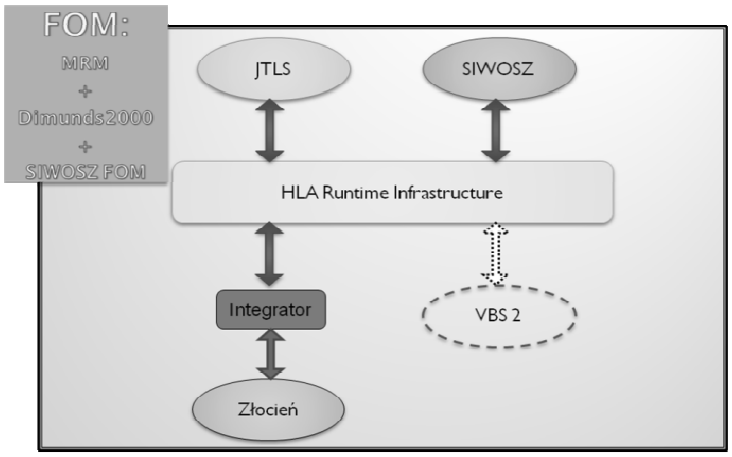
3 Zagadnienia agregacji i deagregacji w połączonych systemach symulacyjnych

Wielorozdzielczość dotyczy problemów o różnej skali i złożoności a tym samym na wielu poziomach szczegółowości odwzorowania. Bezpośrednim powodem stosowania agregacji są potrzeby informacyjne o zróżnicowanej szczegółowości na różnych poziomach decyzyjnych. Integracja danych w połączonym systemie symulacyjnym oznacza umożliwienie komunikacji między komponentami składowymi symulacji. Należy dostarczyć takie narzędzia, aby przetworzone dane z modeli wysokiej rozdzielczości były zrozumiałe i akceptowalne przez wszystkich uczestników symulacji, ze szczególnym uwzględnieniem symulatorów niskiej rozdzielczości. Oczywiście o poprawny transfer danych od systemu niskiej rozdzielczości do docelowego (o wysokiej) również trzeba zadbać. Integrator symulatorów symulacyjnych musi pozwolić na wydajną integrację nie tylko omawianych w rozdziale wcześniejszym symulatorów, powinien również zostać zaprojektowany tak, aby pozwolić na łatwą jego rozbudowę i wsparcie dla innych systemów – jeśli zajdzie taka potrzeba.

Systemy symulacji wielorozdzielczej mają zazwyczaj strukturę hierarchiczną. Objawia się ona tym, że symulator na wyższym poziomie korzysta z danych wytworzonych przez symulator poziomu niższego. Jeśli symulatorem najniższego poziomu uznamy symulator czołgu, to szereg danych potrzebnych do zasymulowania jego działania nie jest istotny z punktu widzenia dowódcy wojsk pancernych pracującego na stanowisku wyposażonym w symulator taktyczny wojsk pancernych. To, co może interesować dowódcę, to potencjał całej dywizji pojazdów, bez potrzeby rozróżnienia pojedynczego czołgu. Aby to wymaganie spełnić, potrzebny jest sprawny mechanizm potrafiący zebrać dane od każdego symulatora i zastosować np. w kontekście całej dywizji, by odwzorować z największą dokładnością jej stan.

4 Adaptowalny integrator

Docelowe środowisko symulacyjne uwzględniające założenia standardu HLA przedstawione zostało na poniższym diagramie:

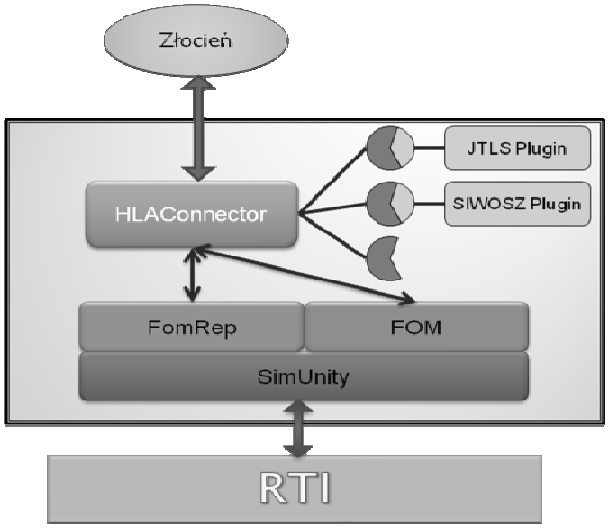


Rys. 2. Rozproszone środowisko symulacyjne
Fig. 2. Distributed simulation environment

Systemy JTLS, SIWOSZ oraz Złocień (w opracowaniu jest kwestia dołączenie symulatora VBS 2 do federacji) stanowią federatów. Każdy z nich działa na innym poziomie szczegółowości i posiada indywidualny model SOM (opisujący dane udostępniane oraz dane, którymi jest zainteresowany). Ich agregacja tworzy wielorozdzielczy model FOM, obowiązujący w całej federacji. Aby współpraca symulatorów była możliwa konieczne jest zapewnienie integracji na dwóch poziomach: technologicznym i modelowym. Zakres technologiczny obejmuje zapewnienie komunikacji w środowisku heterogenicznym. Każdy z systemów działa na innym serwerze (lub serwerach), te zaś pracują pod kontrolą różnych systemów operacyjnych, co wpływa na format danych przesyłanych w sieci. Integracja modelowa musi umożliwić wymianę danych o różnym poziomie szczegółowości. Oba aspekty integracji uwzględnia wytworzony integrator. Ma on postać komponentu programowego, który pośredniczy w komunikacji pomiędzy Złocieniem a RTI. Poniższy diagram prezentuje jego ogólną budowę – Rys. 3.

Integrator składa się z pięciu modułów, które posiadają określony zakres funkcjonalny a każdy z nich może być zmodyfikowany bez wpływu na pozostałe. Są to:

- Moduł SimUnity- odpowiada za komunikację z RTI. Udostępnia generyczne klasy repozytoriów stanów oraz mechanizmy kodowania i dekodowania danych do postaci sieciowej;
- Moduł Fom- uszczegóławia moduł SimUnity dla konkretnego modelu FOM. Zawiera implementację koderów i dekoderów danych dla wszystkich obiektów i interakcji;
- Moduł FomRep- uszczegóławia moduł SimUnity w zakresie repozytoriów stanów obiektów symulacyjnych;
- Moduł HlaConnector- odpowiada za współpracę symulatora Złocień z RTI przez wykorzystanie wyżej wymienionych modułów;
- Moduł wtyczek- wtyczki są komponentami udostępniającymi mechanizmy agregacji i deagregacji danych na potrzeby współpracy z symulatorem Złocień.



Rys. 3. Architektura integratora
Fig. 3. Integrator architecture

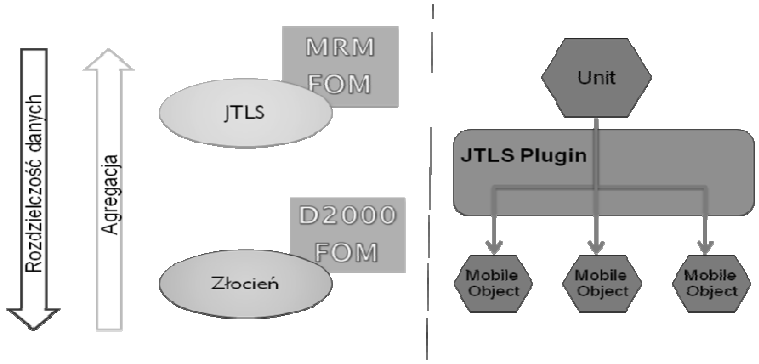
Moduł SimUnity zapewnia integrację technologiczną. To on odpowiada za dwukierunkową komunikację Złocienia z RTI a tym samym za konwersję danych z postaci sieciowej do zrozumiałej przez symulator (oraz w przeciwnym kierunku). Udostępnia także generyczne komponenty, w szczególności repozytoria stanów obiektów symulacyjnych, które używane są przez pozostałe moduły.

Integracja modelowa ma zapewnić zgodność z FOM federacji oraz umożliwić wymianę danych na różnych poziomach rozdzielczości. Moduł Fom jest programową implementacją FOM federacji. Jego zadaniem jest interpretacja wykrytych w trakcie symulacji obiektów oraz interakcji i mapowanie ich na obiekty, którymi operuje Złocień. Przykładowo jeśli symulowana przez niego dywizja wejdzie w kontakt z samolotem

symulowanym przez JTLS, zadaniem modułu Fom jest rozpoznanie typu obiektu JTLS i utworzenie jego odpowiedniej reprezentacji w Złocieniu. Stan obiektu symulacyjnego obsługiwanego przez federację przechowywany jest w module FomRep. Uszczegóławia on generyczne repozytoria stanów modułu SimUnity do konkretnego modelu FOM. Każdy z obiektów symulacyjnych, którym zainteresowany jest Złocień, posiada odzwierciedlenie stanu swoich atrybutów w dedykowanym repozytorium.

Opisane moduły zapewniają obsługę zdarzeń symulacyjnych, które pojawiają się w trakcie działania symulacji. Za ich współpracę ze Złocieniem odpowiada moduł HLAConnector. Stanowi on swego rodzaju *proxy* pomiędzy symulatorem a warstwą HLA. Zapewnia, że wszelkie zdarzenia wygenerowane przez federację będą zrozumiałe przez Złocienia a jego komunikaty zostaną dostarczone do RTI. Dla realizacji swoich zadań moduł HLAConnector wykorzystuje wtyczki (które stanowią wspomniany wcześniej moduł wtyczek). Są one oddzielnymi komponentami programowymi (technicznie mającymi postać bibliotek DLL), które odpowiadają za agregację i deagregację danych. HlaConnector decyduje, czy konieczna jest agregacja lub deagregacja danych. Jeśli tak, to są one przekazywane do odpowiedniej wtyczki. Następnie aktualizowany jest stan obiektu w repozytorium lub tworzona jest interakcja. Odpowiada za to moduł FomRep. Przed aktualizacją stanu obiektu w federacji lub wysłaniem interakcji dane są kodowane do postaci sieciowej przez moduł Fom. Następnie moduł SimUnity przekazuje dane do federacji. Komunikaty wysłane do Złocienia przechodzą przez te same warstwy, jednakże w przeciwnym kierunku.

Poniższy diagram przedstawia koncepcję wykorzystania wtyczek:



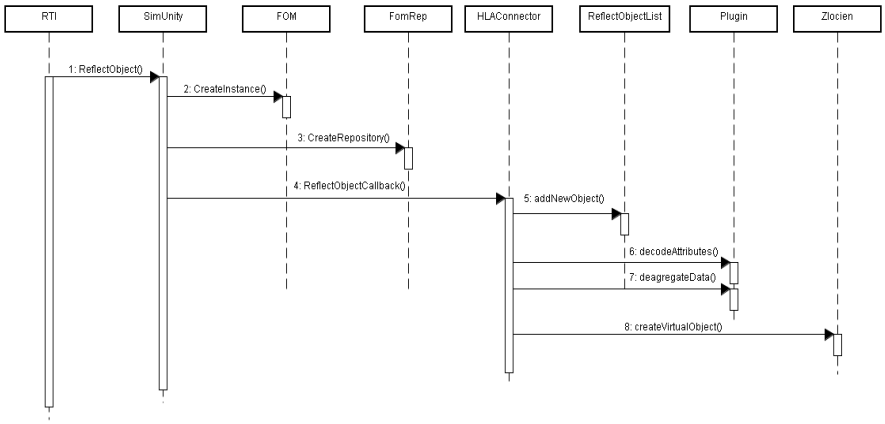
Rys. 4. Idea obsługi wielorozdzielczych danych przy użyciu wtyczek
Fig. 4. Idea of multiresolution data service by plug-in components

Rozpatrując systemy JTLS oraz Złocień jako federatów można stwierdzić, że pierwszy z nich posiada SOM zgodny z modelem MRM (Multi-Resolution Modeling), natomiast SOM drugiego z nich pokrywa się z modelem Dimunds2000. Oba opisują rzeczywistość na różnym poziomie szczegółowości (w tym ujęciu Złocień jest symulatorem wysokiej rozdzielczości, natomiast JTLS niskiej). W modelu MRM podstawowym obiektem jest Unit, który może reprezentować całą armię, natomiast elementarnym obiektem modelu Dimunds2000 jest MobileObject, który w Złocieniu odpowiada pojedynczej kompanii.

Zmiana stanu obiektu w JTLS pociąga za sobą konieczność deagregacji danych do poziomu zrozumiałego przez Złocienia – jest to zadanie wtyczek. Na przedstawionym diagramie jeden obiekt Unit jest dekomponowany do trzech obiektów MobileObject. Algorytmy odpowiedzialne za tę transformację zawarte są we wtyczce JTLSPlugin.

Zastosowanie architektury wtyczek zapewnia łatwą konfigurowalność integratora. W zależności od potrzeb można go wyposażyć w różne algorytmy agregacji/deagregacji danych. Pozwala to na uczestniczenie Złocenia w różnych federacjach, gdzie może być systemem zarówno wysokiej jak i niskiej rozdzielczości w stosunku do innych uczestników symulacji. Poza wtyczkami na łatwość adoptowania rozwiązania wpływa zastosowanie budowy modułowej. Zmiana modelu FOM wymaga jedynie wymiany modułów Fom oraz FomRep.

Ze względu na złożoną budowę integratora warto przedstawić proces wymiany danych uwzględniający jego poszczególne moduły. Przykładowy diagram przedstawia proces odkrycia nowego obiektu w federacji przez Złocienia – Rys. 5.



Rys. 5. Odkrycie nowego obiektu w federacji
Fig. 5. New object discovery

Kiedy w federacji zarejestrowany zostanie nowy obiekt, RTI informuje o tym wszystkich zainteresowanych federatów (deklarujących zainteresowanie, czyli tzw. subskrypcję). Komunikat *ReflectObject* dociera do modułu SimUnity, który uruchamia metody modułów Fom oraz FomRep. Pierwszy z nich powołuje instancję obiektu, będącego reprezentacją wykrytego obiektu (komunikat *CreateInstance*). Następnie tworzone jest przez moduł FomRep repozytorium stanu tego obiektu (komunikat *CreateRepository*). Po tych operacjach moduł HLAConnector informowany jest o pojawieniu się nowego obiektu. Odbywa się to przy użyciu mechanizmu tzw. wywołania zwrotnego (ang. callback). Moduł HLAConnector rejestruje w module SimUnity funkcję, która ma zostać uruchomiona w przypadku zajścia zdarzenia odkrycia nowego obiektu. W rozpatrywanym przypadku jest to funkcja *ReflectObjectCallback*. Jej wywołanie powoduje dodanie instancji wykrytego obiektu do listy ReflectObjectList, która przechowuje wszystkie wykryte w federacji obiekty. Następnie uruchamiana jest odpowiednia wtyczka, która dekoduje atrybuty obiektu

i dokonuje ich deagregacji (odpowiednio komunikaty *decodeAttribbute* oraz *deagregateData*). Przygotowane w ten sposób dane są wykorzystywane do utworzenia nowego obiektu w Złocieniu (komunikat *createVirtualObject*).

5 Podsumowanie

W ramach prac skonstruowany został adaptowalny integrator symulatorów różnej rozdzielczości działających w standardzie HLA. Zapewnia on dwukierunkową komunikację pomiędzy Złocieniem a RTI, dokonując przy tym integracji na poziomie tak technologicznym jak i modelowym. Zapewnia współpracę symulatorów w środowisku heterogenicznym, gdzie dane przesyłane w sieci mogą mieć różną postać w zależności od systemu operacyjnego. Gwarantuje też, że wymieniane dane będą zrozumiałe przez współpracujące systemy, nawet jeśli nie były projektowane z myślą o działaniu w jednej federacji. Udało się to osiągnąć głównie dzięki zastosowaniu budowy modułowej. Integrator ma budowę warstwową składającą się z pięciu łatwo wymienialnych i konfigurowalnych komponentów. Zmiana modelu FOM federacji lub dołączenie do niej nowego systemu wymaga modyfikacji jedynie modułów Fom oraz FomRep i nie pociąga za sobą ingerencji w pozostałe komponenty. Problem różnej rozdzielczości wymienianych danych rozwiązany został poprzez zastosowanie architektury tzw. wtyczek. W założeniu każdy z symulatorów, które współpracują ze Złocieniem powinien posiadać dedykowaną wtyczkę, która dostarcza algorytmy agregacji i deagregacji danych do postaci zrozumiałej przez Złocień.

Literatura

1. Pierzchała D.: Metoda wymiany informacji między heterogenicznymi systemami symulacji oraz wspomagania decyzji w systemach reagowania kryzysowego. *Symulacja w badaniach i rozwoju*, red. Leon BOBROWSKI, Vol. 1 No. 2/2010, 2010
2. Pierzchała D.:, Integracja rozproszonych systemów symulacji oraz wspomagania decyzji. *Przegląd Telekomunikacyjny i Wiadomości Telekomunikacyjnych*, Zeszyt 8-9/2009, 2009
3. Pierzchała D.: Standardy symulacji rozproszonej. *Materiały konferencyjne XV Warsztatów Naukowych PTSK*, pp. 322–333, Warszawa, 2009
4. Pierzchała D.: Designing and testing method of distributed interactive simulators. *Proceedings of the 15th International Conference on Systems Science*, Wrocław, 2004
5. Antkiewicz R., Najgebauer A., Kulas W., Pierzchała D., Rulka J., Tarapata Z., Wantoch-Rekowski R.: Selected Problems of Designing and Using Deterministic and Stochastic Simulators for Military Trainings. *43rd Hawaii International Conference on System Sciences*, IEEE Computer Society, 5–8 January, Koloa, Kauai, Hawaii (USA), 2010

Streszczenie

W pracy opisano komponenty programowe, których zadaniem jest integracja symulatora Złocień z symulatorami działających w federacji zgodnej ze standardem HLA. Głównym założeniem była możliwość dostosowania do różnych modeli FOM oraz zapewnienie współpracy symulatorów operujących na danych o różnych

rozdzielczościach. Zaprojektowany integrator posiadają budowę modułową – moduły stanowią kolejne warstwy, przez które dane są przesyłane do/z RTI.

Zapewniona jest współpraca symulatorów w środowisku heterogenicznym, gdzie dane przesyłane w sieci mogą mieć różną postać w zależności od systemu operacyjnego. Gwarantuje się też, że wymieniane dane będą zrozumiałe przez współpracujące systemy nawet jeśli nie były projektowane z myślą o działaniu w jednej federacji.

The adaptive integration of multi-resolution simulations in the HLA architecture

Summary

The paper describes the program components dedicated to integrate the simulator Zlocien with simulators operating in the HLA federation. The main goal was the ability to adapt different models of FOM and ensure cooperation of simulators that operate on multiresolution data. Designed integrator has a modular structure – modules represent the layers, through which data is sent to / from the RTI.