Symulacja w Badaniach i Rozwoju
Vol. 7, No. 1-2/2016

Dariusz PIERZCHAŁA, Stanisław SKRZYPECKI
Wojskowa Akademia Techniczna, Wydział Cybernetyki,
Instytut Systemów Informatycznych
ul. Gen. S. Kaliskiego 2, 00-908 Warszawa
E–mail: dariusz.pierzchala@wat.edu.pl, stanislaw.skrzypecki@wat.edu.pl

Wieloagentowa i wielorozdzielcza symulacja rozproszona DisSim – VBS

1 Wstęp

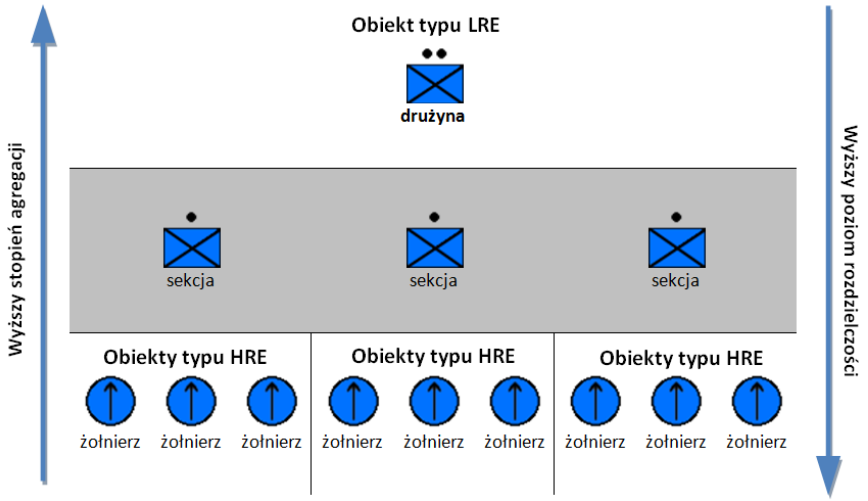
Istotnym wsparciem na współczesnym pola walki, bez którego nie funkcjonują współczesne siły zbrojne, są różne technologie, a w ramach tego najnowocześniejsze systemy informatyczne. W zakresie wspomagania planowania, dowodzenia oraz szkolenia istotną rolę odgrywają komputerowe systemy symulacyjne. Pozwalają m.in. na wielokryterialną analizę scenariuszy działań oraz utrzymanie świadomości sytuacyjnej dowódcy wzbogaconej o wiedzę „what-if".

Rozwiązania systemowe bazujące na symulatorach mają znacznie szerszy kontekst zastosowania, m. in. w: dziedzinie zarządzania kryzysowego, ekonomii, naukach, inżynieryjnych oraz szeroko rozpowszechnionej rozrywce growej. Jednakże to właśnie systemy symulacyjne do wsparcia działań bojowych są najbardziej zaawansowanymi rozwiązaniami modelowymi i technologicznymi - rozproszonymi, wielorozdzielczymi i wieloagentowymi. Symulacja rozproszona (ang. *distributed simulation*) pozwala na integrację różnych symulatorów oraz pracę wielu osób w ramach jednego eksperymentu symulacyjnego. Modelowanie wielorozdzielcze (ang. *multiresolution modelling* – MRM) wspiera odwzorowanie hierarchii poziomów dowodzenia, z odzwierciedlaniem zróżnicowanych poziomów szczegółowości symulowanych obiektów i grup. Natomiast idea autonomicznych obiektów, posiadających dedykowane modele symulacyjne zachowania oraz zdolność podejmowania autonomicznych decyzji (są tzw. agenty), których przykładami mogą być zarówno żołnierz jak i pluton lub dywizja, stanowi sedno systemów wieloagentowych (ang. m*ulti-agent systems* – M.A.S.).

Artykuł stanowi podsumowanie prac nad modułowym środowiskiem rozproszonego eksperymentu symulacyjnego, odwzorowującego działania bojowe na poziomie taktycznym z wykorzystaniem dwu systemów: wysokorozdzielczej symulacji wirtualnej *Virtual Battlespace 3* (VBS3) oraz autorskiego programowego pakietu DisSim (zrealizowanego w języku Java) do konstruktywnej symulacji dyskretno-zdarzeniowej. W efekcie prac powstał zestaw narzędzi do konstrukcji sieciowych środowisk działań, oparty na komunikacji wykorzystującej protokoły DIS i HLA oraz dający generyczne rozwiązanie dla agregacji/deagregacji danych pomiędzy modelami na różnych poziomach rozdzielczości odwzorowania symulowanego świata.

2 Modelowanie wieloagentowe i wielorozdzielcze w symulacji rozproszonej

W modelowaniu wielorozdzielczym konstruowana jest rodzina modeli, pokrywająca pojedyncze obiekty i ich grupowe agregaty, celem wykorzystania na różnych poziomach szczegółowości. Wielorozdzielcze modele umożliwiają kontrolowanie detali i uproszczeń w odwzorowaniu stanu a przez to również ilości przetwarzanych danych. Stopień odwzorowania szczegółów modelowanej rzeczywistości nazywany jest poziomem rozdzielczości. Pokrewnym terminem, oznaczającym stopień pogrupowania (agregowania) obiektów, jest poziom agregacji. Obiekty o wysokim poziomie agregacji nazywamy obiektami typu LRE (ang. *low-resolution entity*), natomiast na wysokim poziomie rozdzielczości lokują się obiekty typu HRE (ang. *high-resolution entity*). Obiekty programowe o niższym poziomie rozdzielczości, odwzorowujące stany grup obiektów wyższej rozdzielczości, nazywane są agregatami. Proces transformujący stan grupy obiektów do stanu pojedynczego agregatu nazywany jest agregacją, natomiast proces odwrotny (od zagregowanego obiektu do grupy obiektów) nazywany jest deagregacją – Rys. 1. Obydwa te procesy skutkują zmianą poziomu rozdzielczości (tym samym poziomu agregacji) i wymagają do tego adekwatnie zdefiniowanych reguł i zasad agregacji/deagregacji. Istnieje szereg problemów, które należy pokonać w tworzeniu algorytmów transformacji stanu obiektów, należą do nich m.in. brak spójności, asymetryczność i nieefektywność przekształceń.

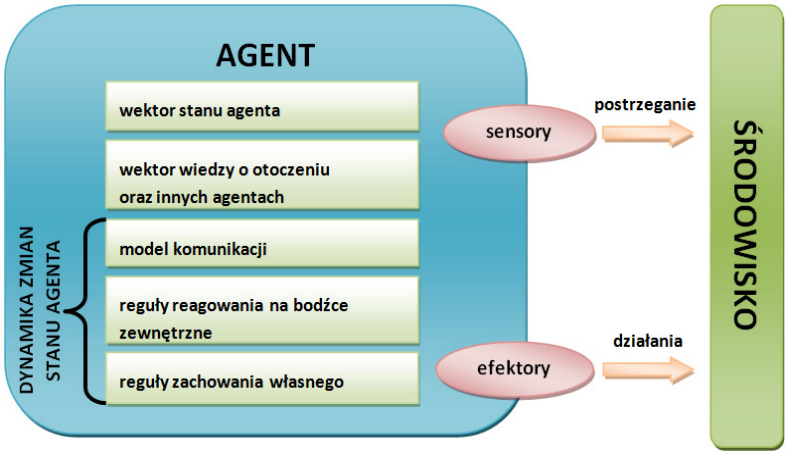


Rys. 1. Istota modelowania wielorozdzielczego
Fig. 1. The idea of multiresolution modelling

Problemy agregacji i deagregacji oraz opracowane dla nich metody modelowania wielorozdzielczego mają swoją historię – możemy wydzielić ich dwie zasadnicze grupy: VRM (ang. *Variable Resolution Modelling*) oraz CRMC (ang. *Cross Resolution Modelling Connection*). Podstawową różnicą pomiędzy obiema metodami jest to,

że w przypadku VRM buduje się nowe modele w celu umożliwienia zmiany rozdzielczości, a w przypadku CRMC łączy się istniejące modele, zaprojektowane pierwotnie do autonomicznego wykorzystania. Do najpopularniejszych technik VRM należy zaliczyć: *Selected viewing, Alternative submodels* (*or model families*) oraz *Integrated hierarchical variable resolution* (*IHVR*). Natomiast grupę CRMC reprezentują przede wszystkim: *agregacje i deagregacje* oraz *Multiple Representation Entity* (*MRE*).

Systemy wieloagentowe (ang. *multi-agent systems – M.A.S.*) to systemy, w których podstawą działania jest istnienie inteligentnych programowych agentów (ang. *intelligent agents)*. Agent jest obiektem osadzonym w środowisku programowym i zdolnym do podejmowania i wykonywania autonomicznych decyzji w tym środowisku w celu osiągnięcia swoich zamierzeń. Zakresy inteligencji, percepcji i autonomii agentów są różne, co wynika nie tylko z zaawansowania zaimplementowanych algorytmów, ale również z samej koncepcji i przeznaczenia agenta. Może on reprezentować zarówno byt materialny (osobowy lub bezosobowy), jak i niematerialny (konceptualny). Pomimo dużego podobieństwa do obiektów (w paradygmacie programowania obiektowo-zorientowanego: obiekt - enkapsuluje stan, jest zdolny do wykonywania metod na tym stanie i do komunikacji poprzez przekazywanie wiadomości), agenty różnią się od klasycznych obiektów przede wszystkim stopniem autonomiczności. Najlepiej różnicę pomiędzy agentami a obiektami oddaje stwierdzenie: obiekt wykonuje działania - ponieważ *może*, agent wykonuje działania - ponieważ *chce*.



Rys. 2. Koncepcja agenta programowego
Fig. 2. The concept of software agent

W modelu agenta można przyjąć następujące cechy i funkcje (Rys. 2):

- wektor stanu agenta (ang. *belief about itself*);

- wektor wiedzy o otoczeniu (ang. *belief about environment*) oraz innych agentach (ang. *belief about agents*);
- model komunikacji;
- reguły reagowania na bodźce zewnętrzne;
- reguły zachowania własnego (myślenie i planowanie działań).

Trzy ostatnie składowe reprezentują w modelu agenta dynamikę zmian stanu (na tle czasu) i tym samym są przedmiotem algorytmizacji w modelu symulacyjnym agenta.

Współczesne komputerowe symulatory programowe budowane są jako systemy rozproszone (ang. *distributed simulation*), co pozwala na skalowalne eksperymenty symulacyjne wykonywane z użyciem wielu programów zintegrowanych w jednym środowisku. Podstawowym celem, a historycznie biorąc pierwszym, stosowania symulacji rozproszonej jest przyspieszenie obliczeń poprzez zrównoleglenie zadań oraz pozbycie się ograniczeń pamięciowych co do wielkości przetwarzanych modeli. Od pewnego czasu równie ważne jest odwzorowanie naturalnego rozproszenia (dyslokacji) symulowanych obiektów oraz ćwiczących osób, odwzorowanie rzeczywistych zależności i hierarchii pomiędzy nimi, możliwość integracji heterogenicznych platform sprzętowo-programowych, adaptacja istniejących systemów (ang. *legacy*) oraz dynamiczna rekonfigurowalność środowiska.

Najczęściej wykorzystywanymi protokołami komunikacji i synchronizacji w środowiskach symulacji rozproszonej są DIS i HLA. Założeniem protokołu DIS (ang. *Distributed Interactive Simulation*) jest wymiana wiadomości zwanych w tym standardzie PDU (ang. *Protocol Data Unit*). Przenoszą one w czasie rzeczywistym informacje na temat zdarzeń symulacyjnych zapisanych w zdefiniowanym precyzyjnie formacie oraz o zawartości zgodnej ze standardowymi słownikami. DIS został zapisany w zestawie dokumentów organizacji IEEE 1278. Obecnie najnowszą wersją protokołu DIS jest wersja nr 7 przedstawiona w roku 2012 – jej specyfikacja zawiera 72 typy komunikatów PDU. Z kolei HLA (ang. *High Level Architecture*), czyli Architektura Wysokiego Poziomu, została przyjęta przez gremia międzynarodowe w standardach: IEEE-1516 oraz standard NATO (STANAG 4603). Podstawowym przeznaczeniem HLA jest wsparcie komunikacji i zarządzania czasem oraz danymi w rozproszonych eksperymentach symulacyjnych we wszystkich reżimach czasowych. Wspomnieć należy o kilku istotnych różnicach pomiędzy DIS i HLA:

- do poprawnego funkcjonowania DIS nie wymaga dodatkowego oprogramowania – w przypadku HLA niezbędne jest skonfigurowanie centralnego oprogramowania zwanego RTI (ang. *Runtime Infrastructure*);
- HLA umożliwia prowadzenie symulacji w czasie rzeczywistym, skalowanym oraz zdarzeniowym – DIS pracuje wyłącznie w reżimie czasu astronomicznego (zwanego również rzeczywistym);
- w komunikacji poprzez DIS stosuje się dokładnie opisane w standardzie PDU – dla federacji standardu HLA definiuje się każdorazowo własne modele wymiany danych SOM/FOM;
- HLA posiada mechanizmy zarządzania czasem i przesyłaniem danych – DIS opiera się na mechanizmach sieciowych;

- DIS działa w trybie broadcast co skutkuje dużym obciążeniem sieci – działanie w trybie *multicast* poprzez mechanizm subskrypcji wybranych obiektów i zdarzeń w HLA może znacząco ograniczyć liczbę przesyłanych komunikatów.

3 Symulacja zdarzeniowa w pakiecie DisSim

Dyskretną symulację zdarzeniową (ang. *Discrete Event Simulation* - DES) wyróżnia skokowy upływ czasu sterowany przyrostami wynikającymi z kolejno zaplanowanych chwil realizacji zdarzeń. Zdarzeniem *e* ze skończonego zbioru zdarzeń *E* nazywamy zaplanowaną algorytmicznie zmianę stanu obiektu/systemu w określonej chwili czasu symulacyjnego *t:*

$$e =< t, f_e^S >, e \in E, t \in T$$

Stanem systemu *S* nazywamy zestaw wartości atrybutów obiektu modelowanego systemu w dowolnej chwili *t* czasu symulacyjnego, zdefiniowany jako czwórka uporządkowana:

$$S =< o, a, v, t >, o \in O, a \in A_c, v \in V_a^c, t \in T$$

$O = \{o = \{id, c\}, c \in C^0\}$ – zbiór obiektów klasy *c* identyfikowanych przez *id*, o wartości niepowtarzalnej w zbiorze obiektów.

C^0 – niepusty zbiór klas modelowanych obiektów.

A_c – niepusty zbiór atrybutów określonych dla klasy obiektów $c \in C^0$.

V_a^c – zbiór dopuszczalnych wartości atrybutu $a \in A_c$ klasy obiektów $c \in C^0$.

$t \in T$ – czas symulacyjny, zmienna: $T \subset R_+ \cup \{0\}$ *z* podzbioru przeliczalnego, tzn. $(\exists t_i, t_k)\ (\neg \exists t_j)\ (t_i < t_j < t_k)$ – czyli istnieją dwa takie momenty, że nie istnieje żaden inny między nimi – oznacza to dyskretny upływ czasu.

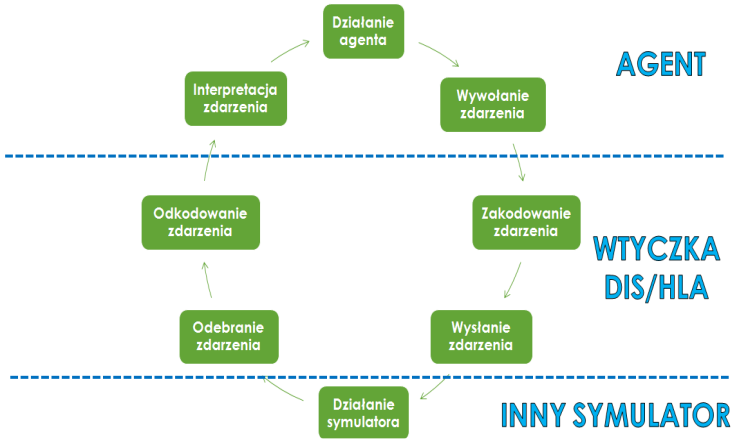
Funkcja zmiany stanu f_e^S wyznacza stan $s \in S$, w jakim znajdzie się system w chwili *t* po zajściu zdarzenia *e:*

$$f_e^S: T \times S \rightarrow S$$

DisSim jest programowym pakietem zrealizowanym w języku Java do programowania właśnie konstruktywnej symulacji zdarzeniowej. Obsługuje sekwencyjnie zdarzenia planowane w odniesieniu do wielu obiektów symulacyjnych (*BasicSimEntity*) a przechowywane we wspólnym kalendarzu zdarzeń (*BasicSimStateChange*). Umożliwia dynamiczne konfigurowanie obiektów poprzez język skryptowy oraz pozwala na rozszerzanie programu za pomocą wtyczek (*Plugins*). Przesyłanie komunikatów realizowane jest za pośrednictwem specjalizowanej szyny zdarzeń. Pakiet zawiera szereg klas pomocniczych umożliwiających: generowanie liczb i procesów pseudolosowych wg zadanych rozkładów (*RandomGen*), monitorowanie zmian i gromadzenie wartości przyjmowanych przez wskazane zmienne w czasie symulacyjnym (*MonitoredVar*), a także wnioskowanie statystyczne na zgromadzonych przez monitory szeregach czasowych (*Statistics*).

4 Zintegrowane środowisko symulacji DisSim – VBS

Zintegrowane środowisko symulacyjne DisSim – VBS umożliwia wykonywanie eksperymentów symulacyjnych na różnych poziomach dowodzenia, odwzorowuje wielopoziomowo modelowaną rzeczywistość oraz zapewnia interoperacyjność i integrację heterogenicznych symulatorów programowych poprzez protokoły DIS/ HLA. Koncepcja komunikacji pomiędzy symulatorami zakłada istnienie kilku warstw o wydzielonych odpowiedzialnościach – pierwsza na poziomie agenta, a następnie na poziomie wtyczki DIS/HLA. Przyjęta struktura warstw umożliwia rozdzielenie algorytmów odpowiedzialnych za zachowanie agentów programowych od funkcji komunikacji z systemami zewnętrznymi – Rys. 3. Obiekty warstwy wtyczki zajmują się wysyłaniem, odbieraniem, kodowaniem i dekodowaniem komunikatów przy użyciu odpowiedniego protokołu. Natomiast w warstwie agenta zakodowane są modele symulacyjne pojedynczych obiektów, ich agregatów oraz otoczenia.

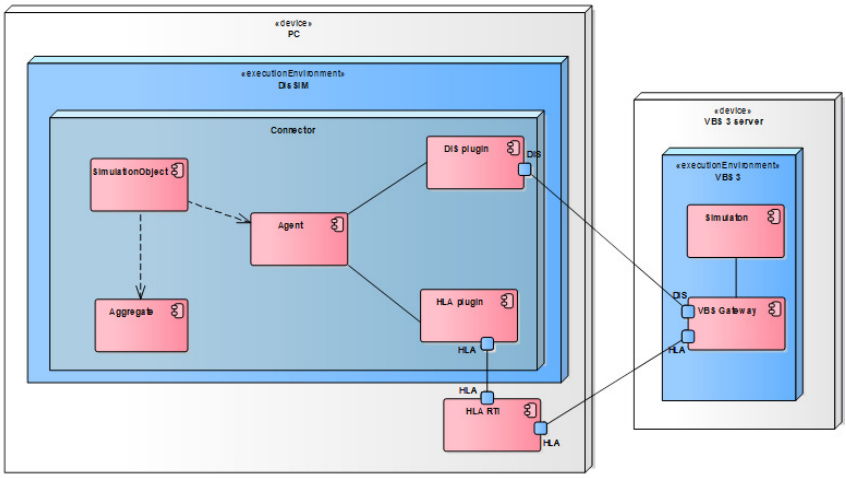


Rys. 3. Warstwy programowe zintegrowanego środowiska

Fig. 3. Software layers of the integrated environment

Każdy symulowany agent powołany na bazie *BasicSimEntity* jest samokonfigurowalny do zadanego poziomu rozdzielczości i agregacji, przez co może dostosować się do wymaganego poziomu dowodzenia. To przyporządkowanie do jednego z poziomów wymaga opracowania dla agenta stosownych reguł agregacji oraz deagregacji do co najmniej jednego sąsiedniego poziomu, na którym znajduje się inny agregat. Reguły te pozwolą na propagowanie zachodzących zmian w celu utrzymania spójności modelu symulacyjnego na każdym poziomie rozdzielczości. Przyjęto, że obiekty o najwyższej rozdzielczości (nieagregowane) to agregaty poziomu L_0, zwiększanie się poziomu dowodzenia odpowiada zwiększaniu się stopnia agregacji obiektów (np. żołnierz – agregat poziomu L_0, sekcja – agregat poziomu L_1, drużyna – agregat poziomu L_2). Agregaty poziomów skrajnych (najniższy i najwyższy) realizują tylko jednokierunkową komunikację (połączoną odpowiednio z agregacją lub deagregacją).

Każdy obiekt symulacyjny *BasicSimEntity* może być agregatem albo agentem poziomu pojedynczego obiektu. Zaimplementowane w kodzie agenta mechanizmy programowe umożliwiają wysyłanie zdarzeń odpowiadających zmianom stanu symulacji (*BasicSimStateChange*) do DIS lub HLA poprzez odpowiednie wtyczki: *DIS plugin*, *HLA plugin*. Zaimplementowane zgodnie z modelem RPR FOM wtyczki realizują dalej procedurę komunikacji z zewnętrznymi systemami o modelu wymiany danych zgodnym z RPR FOM. W przypadku zrealizowanego przez autorów środowiska systemem takim jest symulator VBS3 – Rys. 4.

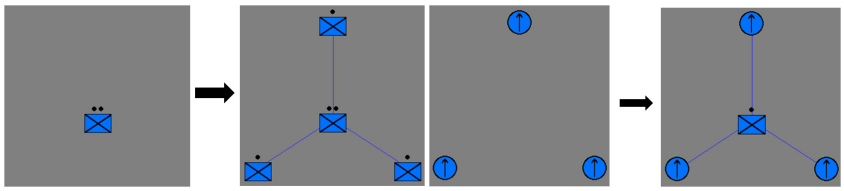


Rys. 4. Architektura komponentów zintegrowanego środowiska
Fig. 4. Component architecture of the integrated environment

Przyjęte rozwiązania projektowe pozwoliły na integrację i spójne współdziałanie symulatorów DisSim-VBS poprzez obydwa protokoły DIS oraz HLA dla jednego z modeli danych, tj. RPR FOM. Zastosowanie dedykowanych interfejsów programowych oraz klas abstrakcyjnych pozwala w minimalnym zakresie ingerować w kod w sytuacji, gdy środowisko będzie adaptowane do innego modelu wymiany danych. Zmiany dotyczyć będą obiektów symulowanych, które muszą rozszerzać klasę *BasicSimEntity* lub implementować interfejs *IAgentSubscriber*. Kolejnym krokiem będzie modyfikacja lub opracowanie nowych subkoderów pakietów PDU (do kodowania i dekodowania pakietów standardu DIS) oraz subuchwytów (dla protokołu HLA). Architektura pakietu DisSim umożliwia dołączenie ich jako wtyczek bez potrzeby głębokiego ingerowania w kod oprogramowania istniejącego.

Jako przykład działania zintegrowanego środowiska rozpatrzymy scenariusz, w którym zaplanowane są interakcje pomiędzy obiektami następujących poziomów dowodzenia: żołnierz – sekcja – drużyna. Wymaga to dedykowanych algorytmów propagacji zmian w odpowiednich chwilach czasu symulacyjnego, który płynie w sposób krokowy lub zdarzeniowy. Każdy agregat zadeklarował poziom agregacji, dla którego adekwatne reguły agregacji/deagregacji są dynamicznie doczytywane poprzez mechanizm

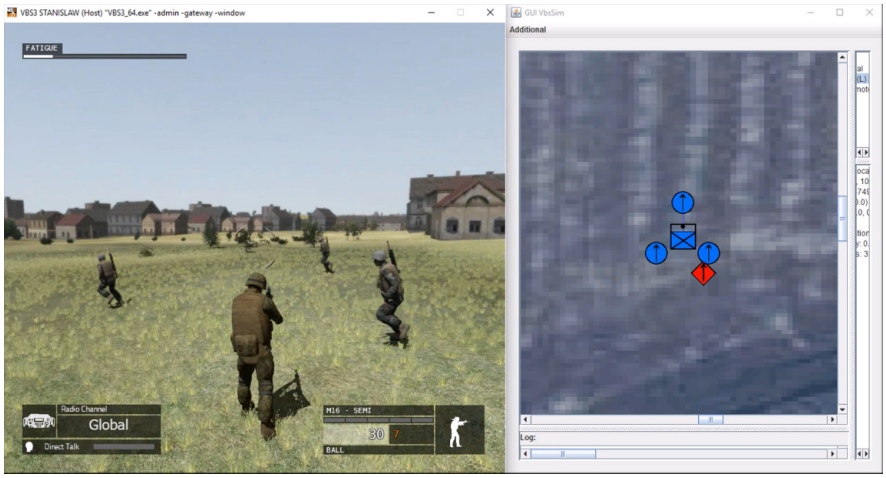
skryptów w języku Groovy. Wyniki testów, prezentujące efekt działania reguł, obrazuje rysunek 5.



Rys. 5. Testy agregacji i deagregacji

Fig. 5. Aggregation and deaggregation tests

W następnej fazie testowano poprawność komunikacji pomiędzy DisSim oraz VBS3 za pośrednictwem wtyczek dla standardów DIS/HLA. Zmiany stanu agentów w symulatorze konstruktywnym wykorzystującym DisSim skutkowały adekwatną zmianą stanu obiektów wysokiej rozdzielczości w VBS3. Potwierdzona została również poprawność komunikacji poprzez obie wtyczki programowe – takie same zdarzenia przekazane zostały przy użyciu wtyczki DIS oraz HLA.



Rys. 6. Testy komunikacji poprzez wtyczki programowe do DIS/HLA

Fig. 6. Communication tests using HLA/DIS plugin connectors

5 Podsumowanie

Modelowanie i symulacja komputerowa przeżywają swój renesans dzięki technologiom, które wniosły możliwość rozproszonego i wielorozdzielczego eksperymentowania z użyciem nowych oraz już stosowanych symulatorów. Skuteczne i adekwatne modelowanie wielorozdzielcze wymaga jednak wielu dedykowanych algorytmów agregacji oraz deagregacji, które można oprzeć na klasycznych metodach lub zastosować systemy regułowe, ontologie, zbiory przybliżone, sieci neuronowe, etc.

Natomiast integracja rozproszonych systemów w jedno spójne w sensie czasu i stanu środowisko wymaga dostosowania wewnętrznych modeli stanu symulatora do modeli wymiany danych wynikających ze słowników standardu DIS lub przyjętych referencyjnie modeli FOM w HLA.

Literatura

1. Chlebicki K., Dyk M., Pierzchała D.: Adaptowalny integrator symulatorów różnej rozdzielczości w architekturze HLA. *Symulacja w Badaniach i Rozwoju*, Vol. 2, No. 3/2011, 2011
2. Dyk M., Pierzchała D., Najgebauer A.: Modelowanie i symulacja sieci IoT w symulatorze SenseSIM. *Symulacja w Badaniach i Rozwoju*, Vol. 6, No. 1/2015, 2015
3. Pierzchała, D.: Symulacja komputerowa – od procedury do chmury. *Projektowanie systemów informatycznych: modele i metody*. WAT, 2014
4. Skrzypecki S.: *Środowisko wieloagentowej i wielorozdzielczej symulacji rozproszonej działań bojowych*, praca magisterska WAT, 2016

Streszczenie

Praca prezentuje zagadnienia wielorozdzielczej (ang. MRM – *multi-resolution modelling*) symulacji rozproszonej (DIS, HLA) w oparciu o modelowanie wieloagentowe (ang. *multi-agent modelling*), wykorzystanej w domenie wsparcia działań bojowych na poziomie taktycznym. Zamieszczono propozycję zestawu narzędzi wspomagających budowę zintegrowanego środowiska złożonego z pakietu symulacji zdarzeniowej DisSim oraz VBS3.

Słowa kluczowe:

Multi-agent and multi-resolution distributed simulation DisSim – VBS

Summary

In the paper issues of multi-agent and multi-resolution modelling in distributed simulation (HLA, DIS) for supporting combat actions on the tactical level are presented. Moreover, it contains the proposition of a set of tools suited for constructing integrated environment based on package DisSim and VBS3..

Keywords: