

Technika modelowania środowiska rzeczywistego wykorzystywana przy budowie edukacyjnych gier taktycznych

Jakub Radziulis, prof. Witold Hołubowicz, Rafał Renk

Uniwersytet im. Adama Mickiewicza w Poznaniu
e-mail: radziuli@amu.edu.pl, holub@amu.edu.pl, rrenk@amu.edu.pl

Andrzej Adamczyk

ITTI Sp. z o.o.
e-mail: andrzej.adamczyk@itti.com.pl

Streszczenie

Referat dokonuje przeglądu rodzajów gier ze szczególnym uwzględnieniem gier symulacyjnych. Analiza jest przeprowadzona w kontekście poszukiwań techniki ćwiczenia pracowników organizacji w zakresie postępowania według ustalonych procedur działania. Przedyskutowane są metody modelowania m.in. graf proceduralny, graf stanów oraz sieci Petriego (w tym kolorowane sieci Petriego). W dyskusji szczególną uwagę poświęcono możliwości tworzenia hierarchii modeli w poszczególnych metodach oraz realizacji komunikacji pomiędzy obiektami w ramach modelu. Przedyskutowano także przydatność poszczególnych metod w kontekście weryfikacji poprawności wykonania gry przez ich uczestników. Ponadto, zaprezentowano podstawowe techniki tworzenia modeli elementów środowiska w wybranej metodzie modelowania. Przedstawione treści poparte zostały faktycznymi przykładami modeli zbudowanych w trakcie tworzenia przykładowej gry taktycznej.

Informacja o autorze

por. referat „Klasyfikacja informacji i danych prawnie chronionych oraz wymagania dotyczące środków informatycznych przeznaczonych do ich przechowywania i przetwarzania”.

1. Wstęp

Problem jaką techniką zamodelować środowisko rzeczywiste pojawił się przy okazji pracy nad opracowaniem gry edukacyjnej, której celem było utrwalenie i weryfikacja wiedzy na temat procedur działania w firmie. Technika modelowania środowiska musiała zostać podporządkowana pewnym założeniom przyjętym na etapie tworzenia gry

Główne założenia towarzyszące tworzeniu gry, które miały wpływ na wybór techniki modelowania środowiska to:

- Model przedstawiający obiekty świata rzeczywistego (ich parametry i logikę działania) powinien być możliwie jak najmniej powiązany z kodem programu gry i wiązać się z nim za pomocą interfejsów. W ten sposób zmiany w modelu nie miałyby dużego wpływu na kod i nie wymagałyby dużej pracy programisty.
- Uczestnicy gry tzw. gracze zostali wcześniej przeszkoleni z zakresu czynności jakie powinni podjąć zgodnie z procedurami w określonych warunkach.
- Podczas gry symulowane jest środowisko w taki sposób aby powstały warunki umożliwiające podjęcie czynności lub działań opisanych w testowanych procedurach.
- Uczestnicy gry mają za zadanie działać zgodnie z treścią procedury, w zależności od podjętego działania przez uczestnika gry zmieniają się symulowane warunki środowiska co ma wpływ na działania innych graczy.
- Symulowane środowisko posiada elementy niezależne, które zmieniają się w określony (założony z góry) sposób mające wpływ na działanie podejmowane przez uczestników gry.
- W grze ma uczestniczyć wielu graczy w różnych lokalizacjach geograficznych a ich działania podlegać będą ocenie.
- Za sukces gry uznaje się sytuację, w której gracz postąpi zgodnie z zapisami w procedurze.

2. Przegląd rodzajów gier

Pojęcie gry można definiować na wiele sposobów. Na potrzeby artykułu przyjęto następującą definicję: „Gra to czynność o ustalonych zasadach, w której bierze zwykle kilka osób (rzadziej jedna), w celach rozrywkowych” [1]. Różnicą pomiędzy grą a rozrywką jest taka, że gra posiada ścisły zbiór zasad i czynności jakie w niej można wykonać, a rozrywka takowych zasad nie musi posiadać.

2.1. Rodzaje gier

Przed przystąpieniem do prac związanych z opracowaniem modelu przeprowadzono analizę i klasyfikację dostępnych na rynku rodzajów gier i ich przydatności w celach edukacyjnych pracowników. Przegląd ten miał służyć wyszukaniu wykorzystywanych w praktyce sposobów modelowania środowiska rzeczywistego w grach.

Istnieje wiele sposobów klasyfikacji rodzajów gier w zależności od różnych kryteriów. Poniżej przedstawiono zdaniem autorów najważniejsze klasyfikacje gier.

Klasyfikacja według zaangażowania zdolności graczy:

- Gry wykorzystujące zdolności graczy – angażujące zdolności fizyczne lub umysłowe uczestników np. gra w piłkę nożną, gry logiczne.

- Gry losowe – gry, w których o zwycięstwie obok zdolności decyduje w różnym stopniu los np. ruletka. Ponieważ wynik gry pomiędzy graczami jest losowy w grach tego typu nie wytwarza się prawie nigdy skomplikowana teoria gry.
- Gry strategiczne – gry, w których istnieją skomplikowane zasady a o zwycięstwie decyduje w dużej mierze przyjęta strategia gry. Bardzo przybliżonymi do gier strategicznych (a wręcz ich podgrupą) są gry taktyczne, w których o zwycięstwie decyduje wykorzystana taktyka.
- Gry symulacyjne – są to gry, które wykorzystują elementy wszystkich z wyżej wymienionych rodzajów gier, tzn. gier wykorzystujących zdolności, gier losowych i gier strategicznych. Przez ich połączenie tworzona jest skomplikowana struktura gry odwzorowująca (symulująca) elementy świata rzeczywistego.

Klasyfikacja według wykorzystywanych zasobów wprowadza podział na gry: komputerowe, planszowe, karciane, gry typu papier – ołówek, gry w kości, terenowe, itp.

Klasyfikacja według liczby uczestników rozgrywki dzieli gry na: jednoosobowe, dwuosobowe i wieloosobowe. Podział na gry dwuosobowe i wieloosobowe ma znaczenie dla twórców gry, ponieważ większość gier wieloosobowych umożliwia tworzenie koalicji graczy co wymaga innego podejścia do tworzenia gry niż w sytuacji gdy gra toczy się w trybie „jeden na jednego”. Istnieje jednakże dużo jest gier, w których dwu- lub wieloosobowość uczestników nie ma zasadniczego znaczenia ponieważ rozgrywka prowadzona jest na zasadzie „każdy z każdym”.

Klasyfikacja według relacji do świata rzeczywistego:

- Gry abstrakcyjne – przykładem może być gra przedstawiająca wojnę dwóch światów w dalekim kosmosie.
- Gry rzeczywiste – np. gry ekonomiczne i handlowe wykorzystujące dane rzeczywiste i działające zgodnie z regułami funkcjonowania rzeczywistego świata.
- Gry mieszane (rzeczywiste z elementami abstrakcji) – gry wykorzystujące dane i zachowanie modeli z rzeczywistego świata w sytuacjach abstrakcyjnych (fikcyjnych).

Klasyfikacja według celów jakie mają zostać osiągnięte podczas gry:

- Gry edukacyjne – gry, których celem jest nauka uczestników gry pewnej wiedzy.
- Gry rozrywkowe – gry, których celem jest przyjemność gry bez elementów edukacyjnych, może być wpleciony element rywalizacji np. wykonanie pewnych czynności w określonym czasie.
- Gry rywalizujące – gry, których celem jest wygrana z przeciwnikiem. W tym przypadku przeciwnikiem może być osoba rzeczywista lub symulowana przez grę.

Ze względu na dużą różnorodność gier i ogromną kreatywność ich twórców coraz więcej gier jest mieszanych tzn. łączy cechy poszczególnych kategorii gier w danym typie klasyfikacji. Przykładem jest gra edukacyjna, w której wprowadzony jest element rywalizacji pomiędzy dwiema grupami graczy.

W odniesieniu do przedstawionych klasyfikacji i założeń wyszczególnionych we wstępie artykułu grę, do której miano opracować modele można nazwać edukacyjną grą taktyczną, ponieważ jest grą edukacyjną i sprawdza taktykę działania – zgodność podejmowanych działań zgodnie z procedurą. Ponadto gra ta jest grą wieloosobową przeprowadzaną w środowisku komputerowym działającą w środowisku rzeczywistym z elementami abstrakcji.

2.2. Teoria gier

Innym znaczeniem gry jest model matematyczny zwany teorią gier [3].

„Teoria gier to dział matematyki zajmujący się badaniem optymalnego zachowania w przypadku konfliktu interesów” [1]. Według tej teorii gra została zdefiniowana jako „dowolna sytuacja konfliktowa” [1], natomiast gracza jako dowolnego uczestnika gry. Według teorii gier uczestnicy gry mają na celu uzyskać dla siebie jak najlepsze wyniki kosztem współgraczy gdzie graczem może być człowiek lub przedsiębiorstwo. Teoria gier ma zastosowanie do odzwierciedlenia sytuacji ekonomicznych, biologicznych, socjologicznych i informatycznych. Każdy gracz (zwany dalej stroną) wybiera pewną strategię postępowania po czym następuje konfrontacja z inną stroną. W zależności od tego czy strategia przeciwnika jest lepsza lub gorsza strona przegrywa lub wygrywa, tracąc lub zyskując wygraną (w teorii gier zwaną wypłatą).

Rozważana edukacyjna gra taktyczna zakłada współpracę grających w celu osiągnięcia celu (sukcesu). Poszukując w założeniach tej gry konfliktu dochodzi się do wniosku, że jedyny konflikt jaki w niej występuje to rywalizacja pomiędzy graczem (uczestnikiem symulacji) a środowiskiem symulowanym. Ponieważ cel gracza, którym jest wykonanie czynności zgodnie z zapisem procedury nie jest tożsamy z symulowanym środowiskiem, zastosowanie teorii gier w opisywanym przypadku nie ma sensu. Inną różnicą pomiędzy podejściem do gry przedstawionym w teorii gier a w niniejszym artykule jest to, że w grze taktycznej nie jest celem sprawdzenie kto zastosował lepszą strategię a weryfikacja czy gracz lub gracze podjęli działania z jasno określoną z góry strategią działania.

2.3. Przegląd stosowanych rodzajów gier w szkoleniu pracowników

Wykorzystanie gry jako sposobu edukacji nie jest tematem nowym, dlatego dokonano przeglądu oferty gier wykorzystywanych w celu przeszkolenia pracowników. Przeanalizowane przykłady takich właśnie gier są gry oferowane jako element szkolenia między innymi przez firmy: House of Skills (produkt: TANGO™) [3], CT Partners (produkt: Apollo 13 - an ITSM case experience) [4], DOOR (produkt: gra „piwna”) [5] i inne.

Celem tych gier jest nauka przez rywalizację. Uczestnicy gry po szkoleniu i zapoznaniu się z materiałem teoretycznym np. związanym z sposobem zarządzania firmą czy zarządzaniem środkami finansowymi, biorą udział w grze, w której rywalizują ze sobą wieloosobowe zespoły w tym kto lepiej wykorzysta nabytą podczas szkolenia wiedzę. Decyzje podejmowane przez graczy mają na celu sprawdzenie jak zachowałaby się firma, gdyby podjęto dane decyzje w sytuacji rzeczywistej. Przeznaczeniem tych gier poza nauką i przyswojeniem wiedzy przez wykorzystywanie jej w symulowanym środowisku jest także weryfikacja nabytych umiejętności oraz lepsze zrozumienie nabytej wiedzy.

Przegląd stosowanych sposobów realizacji gier wykorzystywanych w celu przeszkolenia pracowników wskazał cechy charakterystyczne dla tego rodzaju gier, którymi są:

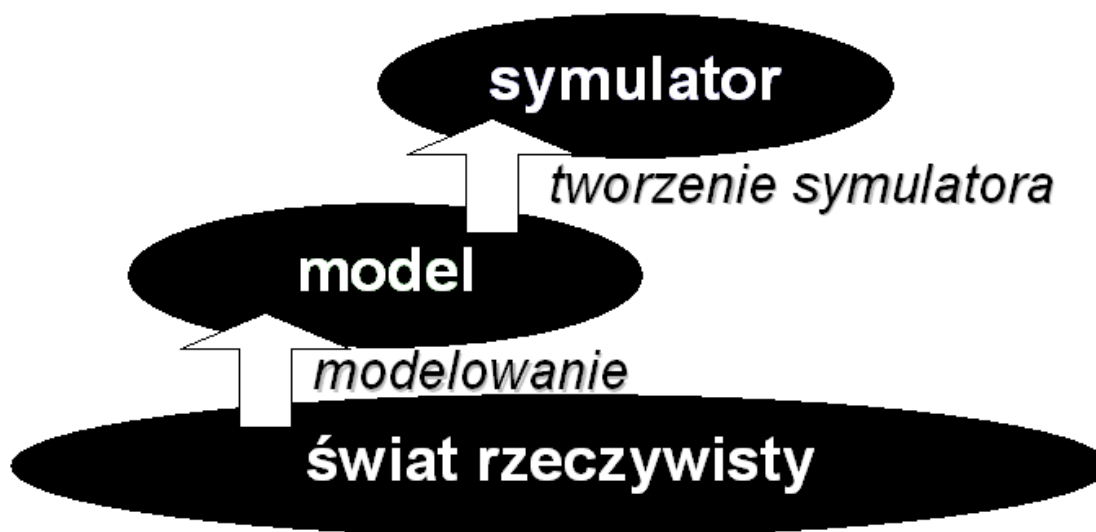
- gry realizowane są jako gry symulacyjne w środowisku rzeczywistym zakładające rywalizację pomiędzy sobą wieloosobowych zespołów.
- sposób realizacji szkoleń za pomocą gier realizowany jest głównie za pomocą dwóch technik:
 - gra planszowa z moderatorem (ołówki i papier)
 - gra komputerowa.

Przegląd możliwych gier nie wskazał możliwego do zastosowania modelu sprawdzającego działanie pracownika zgodnie z procedurami, ponieważ edukacyjna gra taktyczna nie zakłada elementu rywalizacji a edukacji. Nie znaleziono także przykładu gry weryfikującej postępowanie zgodnie z czynnościami opisanymi w procedurach.

3. Metody modelowania świata rzeczywistego

Tworzenie modelu środowiska rzeczywistego jest krokiem pośrednim pomiędzy światem rzeczywistym a stworzeniem kodu symulatora. Modelowanie funkcjonowania świata rzeczywistego oraz obiektów w nim występujących umożliwia lepsze jego zrozumienie. Pozwala to lepiej zasympulować poszczególne jego zachowania. Autorzy artykułu podczas przygotowania do wyboru techniki modelowania wyróżnili dwa podejścia do modelowania:

- Modelowanie obiektów i funkcjonowania świata rzeczywistego tylko w celu lepszego zrozumienia i opisanego jego parametrów – modele służą jedynie do określenia jakie parametry opisują świat rzeczywisty i jakie są relacje pomiędzy poszczególnymi jego elementami (określenie danych wejściowych dla symulatora). Modelowanie jest mniej skomplikowane (oparte na programach graficznych). Zmiany w modelu przenoszą się w trudne do wprowadzenia zmiany w kodzie programu.
- Modelowanie obiektów świata rzeczywistego w celu ich wykorzystania utworzonego modelu w symulatorze. Modele w dalszym ciągu opisują parametry świata rzeczywistego i jego zachowanie, jednakże są bezpośrednio wykorzystywane przez programistę w symulatorze za pomocą interfejsów pomiędzy modelem a kodem gry – niezbędne jest narzędzie modelowania umożliwiające współpracę z narzędziem developerskim kodu symulatora. Modelowanie świata rzeczywistego jest utrudnione przez wymóg dostosowania semantyki modelowania do zgodnej z interfejsem opracowanym przez programistę. Zmiany w budowie modeli łatwo przyswajalne dla symulatora, jako że modele nie są integralną częścią kodu



Rys. 1. Miejsce modelu w procesie tworzenia symulatora

Z punktu widzenia założeń przedstawionych we wstępie zdecydowano się na poszukiwanie techniki modelowania umożliwiającej realizację drugiego z przedstawionych powyżej podejść modelowania świata rzeczywistego.

3.1. Kryteria oceny metod

Aby porównać techniki modelowania określono następujące kryteria porównania poszczególnych technik:

- Stopień trudności tworzenia modeli elementów (obiektów) świata rzeczywistego.
- Stopień trudności weryfikacji modeli.

- Stopień komplikacji budowy modeli wraz ze wzrostem jego złożoności.
- Możliwość tworzenia hierarchii modeli.
- Sposób integracji modeli z symulatorem – możliwość powiązania utworzonego modelu z kodem symulatora.
- Komunikacja pomiędzy obiektami (elementami) w modelu.
- Współbieżność działania obiektów (elementów) w modelu.
- Sposób weryfikacji poprawności wykonywania działań przez uczestnika gry (ocena gracza).

Kryteria te umożliwiły porównanie poszczególnych technik modelowania i wyboru najlepszej z nich dla opracowania modelu, który mogłyby zostać później wykorzystany przez programistę w celu integracji z kodem symulatora.

Ocenie zostały poddane następujące techniki modelowania:

- Modelowanie z wykorzystaniem języka programowania [6].
- Modelowanie obiektów jako encji w bazie danych [7].
- Graf proceduralny (reprezentacja diagramu blokowego).
- Graf stanów (z rozważeniem wykorzystania łańcuchu Markowa).
- Diagramy UML (Unified Modeling Language) [8].
- Kolorowane sieci Petriego (z wykorzystaniem hierarchii) [9, 10,11].

3.2. Modelowanie z wykorzystaniem języka programowania

Pierwszą z analizowanych technik modelowania jest modelowanie obiektów świata rzeczywistego za pomocą języka(ów) programowania. Podczas modelowania obiekty tworzy się przez zredagowanie w odpowiednim języku programowania. Kod opisujący model może mieć postać jednego lub kilku plików komputerowych zawierający zapis modelu, informację o sygnałach wejściowych i wyjściowych.

Wykorzystanie tej techniki modelowania zostało odrzucone z następujących powodów:

- Tworzenie modeli obiektów jest ściśle powiązane z kodem symulatora – czego chcieliśmy uniknąć.
- Tworzenie modelu wymaga znajomości języka programowania co powoduje, że model zależy od platformy programistycznej wykorzystanej do budowania kodu symulatora (wymagana praca programisty a nie analityka).
- Dowolna zmiana w modelu powiązana jest ze zmianą w kodzie symulatora.
- Skomplikowane obiekty są opisane za pomocą wielu parametrów, co komplikuje przejrzystość modeli i ich weryfikację.
- Tekstowa reprezentacja modelu.
- Występują trudności przy tworzeniu hierarchii modeli przy wykorzystaniu języka programowania nie zorientowanego obiektowo.
- Weryfikacja poprawności wykonania działań przez gracza wymaga opracowania dodatkowego kodu przez programistę.

3.3. Modelowanie w bazie danych

W bazie danych można przedstawić elementy świata rzeczywistego w postaci obiektów, ich atrybutów i powiązania pomiędzy obiektami. Elementami takiego modelu są:

- Obiekty (encje) –rzeczy lub obiekty mające dla nas znaczenie, rzeczywiste lub wyobrażalne, o którym informacje muszą być znane lub przechowywane.
- Własności obiektów (atrybuty encji) – dowolny szczegół służący do kwalifikowania, identyfikowania, klasyfikowania, określania ilości lub wyrażania stanu encji lub reprezentuje cechę (własność) obiektu (encji).
- Związki między obiektami (encjami) – nazwane, istotne powiązania istniejące między dwiema encjami. (źródło) lub Związek odzwierciedlający semantycznie różne sprzężenia pomiędzy różnymi lub tymi samymi obiektami (encjami).

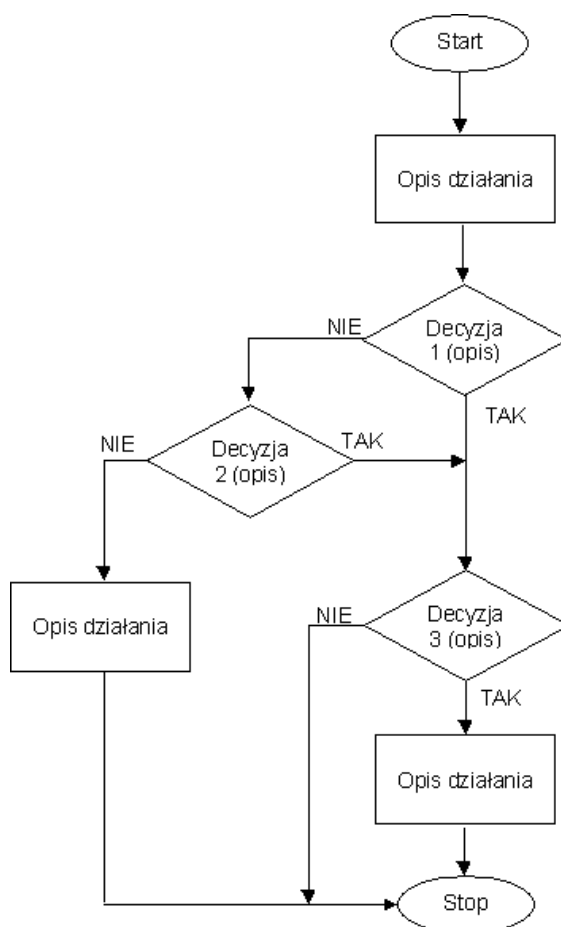
Taka technika pozwala na zamodelowanie obiektów i parametrów, które je opisują. Jednakże cała logika działania poszczególnych obiektów w modelu musi zostać opisana w kodzie symulatora. To powoduje, że model środowiska rzeczywistego jest wkomponowany w kod tworzony przez programistę. Modelowanie za pomocą obiektów w bazie danych jest uciążliwe w przypadku skomplikowanego modelu i trudne do weryfikacji. Komunikacja pomiędzy obiektami musi być realizowana za pomocą kodu programu. Bazy danych umożliwiają tworzenie hierarchii obiektów a weryfikacja działań uczestników gry jest możliwa przez wprowadzenie dodatkowych parametrów do poszczególnych obiektów. Wniosek: w bazie danych można odzwierciedlić świat rzeczywisty za pomocą obiektów i cech poszczególnych jego elementów za pomocą atrybutów, jednakże nie można zamodelować logiki działania elementów świata rzeczywistego.

3.4. Graf proceduralny

Graf proceduralny jest strukturą opartą na opisie algorytmów procedur. Jest to reprezentacja diagramu blokowego, którego poszczególne elementy zostały nazwane węzłami i opisane w następujący sposób:

- Węzeł początkowy (końcowy) opisujący działanie początkowe (końcowe) w grafie.
- Węzeł decyzyjny (przedstawiany w postaci rombu) opisujący decyzję binarną (tak lub nie), która może zostać podjęta w tym węźle.
- Węzeł opisujący działanie (prostokąt).

Opis działania jakie są wykonywane lub treść decyzji jaka ma zostać podjęta jest zawarta w opisie poszczególnych węzłów. Przykład grafu proceduralnego przedstawiono na rys.2.



Rys. 2. Przykład grafu proceduralnego

Próba wykorzystania grafu proceduralnego w celu modelowania świata rzeczywistego była naturalną konsekwencją tego, że procedury, których znajomość jest testowana podczas symulacji zostały opisane za pomocą diagramów blokowych. Zaletą tego podejścia jest duża intuicyjność i łatwość w tworzeniu oraz możliwość stosowania hierarchii, za pomocą której można znacznie uproszczyć budowę modelu.

Wadami grafu proceduralnego przy wykorzystaniu do modelowania środowiska rzeczywistego są:

- Konieczność określenia w dodatkowej strukturze danych opisu warunków wejściowych i wyjściowych dla poszczególnych grafów.
- Duży poziom skomplikowania grafu przy modelowaniu pomimo wykorzystania hierarchii.
- Bardzo duża trudność przy weryfikacji poprawności budowy modelu.
- Brak możliwości zamodelowania realizacji komunikacji asynchronicznej.
- Binarność przy podejmowaniu decyzji mocno komplikująca graf przy możliwości podjęcia wielu decyzji jednocześnie.

Przy analizie możliwości wykorzystania techniki modelowania środowiska rzeczywistego za pomocą tego rodzaju grafów napotkano na problemy, na które nie znaleziono rozwiązania. Wymienione problemy to:

- Połączenie modelu z kodem programu - poza reprezentacją graficzną umożliwiającą opis działania obiektów w modelu graf proceduralny nie daje innych możliwości. Parametry po-

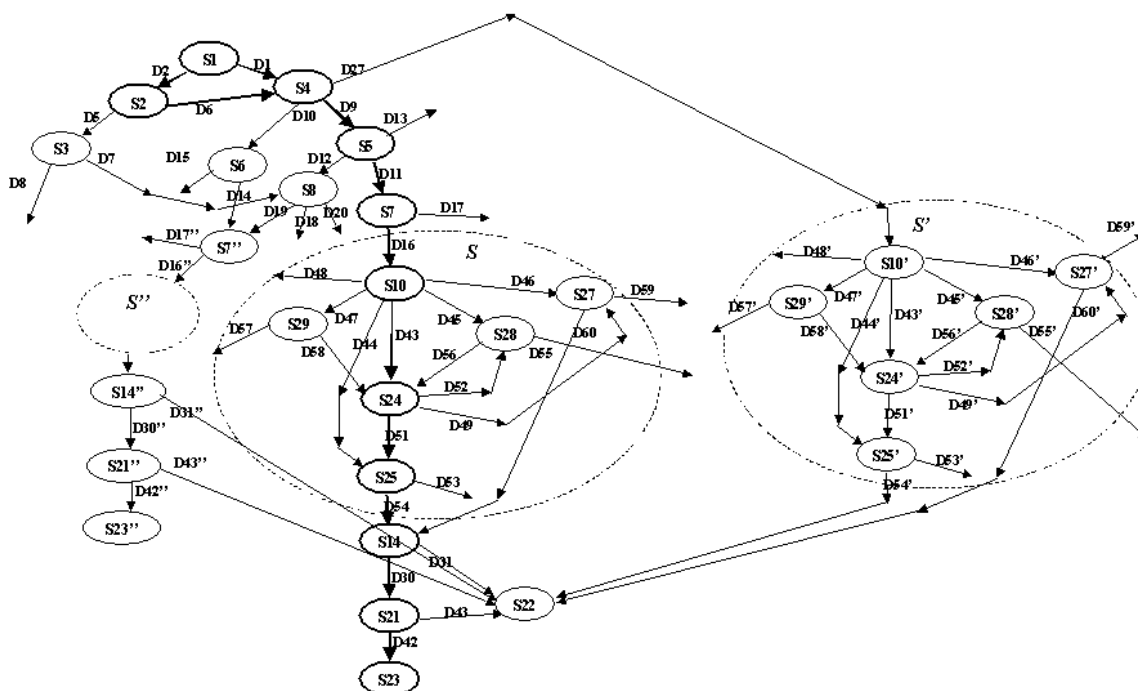
szczególnych obiektów muszą zostać opisane w odrębnej strukturze danych. Sama implementacja obiektu wymaga pracy programisty.

- Modelowanie współdziałania uczestników gry.
- Założona wielowątkowość gry, jak ją zamodelować w grafie proceduralnym.
- Komunikacja asynchroniczna jaka może mieć miejsce pomiędzy obiektami.
- Weryfikacja poprawności działania uczestników gry.

3.5. Graf stanów

Innym podejściem do przedstawienia środowiska rzeczywistego jest wykorzystanie w tym celu grafu stanów (maszyny stanów). Takie podejście wydaje się być naturalnym ponieważ symulator tak jak świat rzeczywisty będzie znajdował się w różnych stanach w zależności od działań podejmowanych przez uczestników gry.

Przykład grafu stanów przedstawiono na rys. 3.



Rys. 3. Przykład grafu stanów

Grafy stanów pozwalają na symulację zdarzeń i działań za pomocą maszyny stanów. Modelowanie za pomocą grafu stanów posiada bardzo prosty i intuicyjny „język notacji” (w postaci stanów i przejść między stanami), modele są łatwe do zaimplementowania w postaci symulatora komputerowego i są łatwo weryfikowalne (wystarczy testowe przejście wszystkimi ścieżkami grafu). Jednakże może powstać bardzo skomplikowany graf stanów (duża liczba możliwych stanów symulacji – duża przestrzeń stanów) w przypadku symulowania wielu działań jednocześnie czy uwzględnienia nietrywialnych możliwości reakcji uczestników symulacji.

Poza przedstawionymi cechami grafu stanów istnieje także konieczność odrębnego rozpatrywania tych samych stanów wynikających z realizacji procesów współbieżnych oraz występują duże problemy przy różnych sposobach zakańczania procesów współbieżnych. Jednym z najważniejszych wniosków jaki wysunięto po analizie przydatności grafów stanów do symulowania edukacyjnej gry taktycznej jest brak możliwości wykorzystania podejścia obiektowego przy budowaniu modelu (jednokrotne definiowanie sposobu zachowania zasobów,

modelu (jednokrotne definiowanie sposobu zachowania zasobów, elementów środowiska, osób itp.).

Model według którego toczyć się ma gra można opisać za pomocą:

- Stanów całej gry (a nie stanów konkretnych obiektów w grze).
- Przejść pomiędzy stanami (zmianom wartości zmiennych) które mogą występować na skutek:
 - obiektywnego zdarzenia symulowanego
 - zdarzenia spowodowanego dokonaniem wyboru przez uczestnika gry (np. podjęciem określonego działania)
 - zdarzeniem spowodowanym przez działalność symulowanych graczy
- Stany gry rozumiane są jako zbiór wartości wszystkich wykorzystywanych w grze zmiennych – zmiana wartości którejkolwiek zmiennej w grze powoduje zaistnienie nowego stanu.
- Symulacja zaczyna się i kończy określonym stanem gry.

Przy tworzeniu stanów istnieje potrzeba definicji parametrów związanych z danymi stanami oraz wartości jakie przyjmują w tych i pozostałych stanach. Poza utworzeniem samych grafów stanów do opisu modelu potrzebne jest utworzenie dodatkowych elementów jakimi są:

- Tabela zmiennych stanu i dopuszczalnych wartości (opis parametrów gry).
- Tabela kombinacji wartości zmiennych stanu (przedstawienie wartości parametrów związanych z każdym stanem).
- Macierz przejść stanów wykorzystywany w kodzie programu do określania przejść pomiędzy stanami.
- Tabela komunikatów związanych ze stanem adresowanych do poszczególnych ról.

Istotnym mankamentem poza złożonością modeli reprezentowanych w postaci grafu stanów jest potrzeba przechowywania struktur danych związanych z opisem modelu w dodatkowych plikach. Każdorazowa zmiana w grafie stanów wymaga zmian w powiązanych z danymi węzłami parametrów. Tworzenie hierarchii stanów związane jest z tworzeniem dodatkowych struktur danych i powiązań między nimi. Natomiast weryfikacja poprawności działania gracza jest bardzo uproszczona, ponieważ wystarczy sprawdzić czy działania gracza doprowadziły do przejść po odpowiednich stanach.

Podczas analizy techniki modelowania świata rzeczywistego za pomocą grafów stanów rozważano zastosowanie łańcuchów Markowa i ich reprezentacji w postaci grafu stanów z przejściami. Przejścia pomiędzy poszczególnymi stanami w diagramie Markowa są stochastyczne i określone przez prawdopodobieństwo ich wystąpienia. Tak więc diagramy Markowa są stochastyczne, co dyskryminuje ich wykorzystanie do modelowania obiektów, których zachowanie jest deterministyczne.

3.6. UML

Unified Modeling Language jest obiektowym językiem modelowania. Jest on oparty na pojęciach obiektowości, takich jak: klasy, obiekty, atrybuty, związki, agregacje, dziedziczenie, metody. UML pozwala wszechstronnie odwzorować modelowaną dziedzinę problemu lub założenia systemu informatycznego.

Podstawowym celem UML jest modelowanie systemów z użyciem pojęć obiektowych. UML zakłada wykorzystanie wielu rodzajów diagramów służących do opisu i zamodelowania analizowanego lub projektowanego systemu, z których każdy skupia się na innym aspekcie.

W notacji UML można wyróżnić następujące diagramy:

- Diagramy przypadków użycia – odwzorowują funkcję projektowanego systemu w sposób widziany przez jego przyszłych użytkowników.
- Diagramy klas – reprezentują klasyczne relacje encja – związek (opisane w punkcie 3.3.) z rozszerzeniami poprawiającymi czytelność diagramów.
- Diagramy przedstawiające dynamiczne własności systemu:
 - Diagramy sekwencji – przedstawiające kolejność przesyłanych komunikatów pomiędzy obiektami.
 - Diagramy współpracy – przedstawiające kolejność przesyłanych komunikatów pomiędzy obiektami wraz z odwzorowaniem statycznej struktury obiektów.
 - Diagramy stanów – przedstawiające stany i przejścia pomiędzy nimi.
 - Diagram aktywności – reprezentujący przepływ sterowania.
- Diagramy implementacyjne:
 - Diagramy komponentów.
 - Diagramy wdrożeniowe.

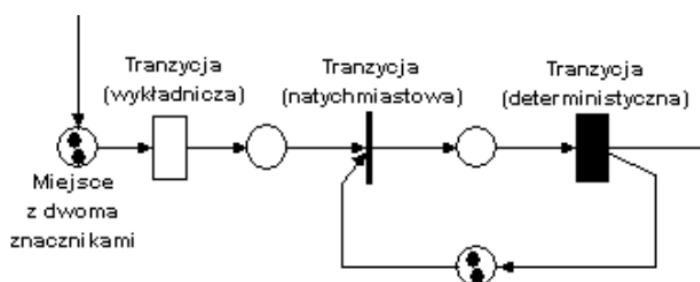
Po przeanalizowaniu możliwych do wykorzystania diagramów do modelowania środowiska rzeczywistego na potrzeby opracowywanej gry, okazało się, że można w tym celu wykorzystać diagramy klas, stanów i sekwencji lub aktywności. O ile diagramy klas i stanów są pewną modyfikacją do technik modelowania opisywanych w punktach 3.3. i 3.5., to diagramy sekwencji lub aktywności uzupełniają powyższe diagramy o możliwość przedstawienia działania i komunikacji pomiędzy obiektami.

Cechy jakie zdecydowały o wykluczeniu UML-a jako techniki modelowania środowisk rzeczywistego na potrzeby gry to:

- UML nadaje się bardzo dobrze do opisowego przedstawienia modelu za pomocą różnych diagramów co ułatwia zrozumienie działania i relacji pomiędzy obiektami. Nie można jednak w prosty sposób zintegrować utworzonych modeli z kodem programu.
- Narzędzia umożliwiające integracje modelu z kodem np. Rational Rose Enterprise wymagają od twórcy modelem znajomości języka programowania. W najprostszej sytuacji gdyby wykorzystano proste narzędzia graficzne do tworzenia modeli programista musiałby tworzyć modele w kodzie od początku.
- Każdorazowe zmiany w modelu wymaga zmian w kodzie programu.
- Opis modeli (parametrów, danych wejściowych i wyjściowych) w zależności od wykorzystanego narzędzia wymaga dodatkowej struktury danych.

3.7. Kolorowane sieci Petriego

Język modelowania za pomocą sieci Petriego został formalnie zdefiniowany w 1962 roku przez profesora Carla Adama Petri w ramach jego pracy doktorskiej pt. „Kommunikation mit Automaten”. Sieć Petriego jest to graf złożony z miejsc, tranzycji i krawędzi skierowanych, w którym poruszają się znaczniki. W każdym miejscu sieci może znajdować się wiele różnych znaczników. Wzbudzenie tranzycji w bieżącym znakowaniu sieci zależy od obecności znaczników w miejscach wejściowych (poprzedzających daną tranzycję w grafie). Wzbudzenie tranzycji powoduje przesunięcie znacznika z miejsca wejściowego tranzycji do miejsca wyjściowego. Przykład sieci Petriego przedstawiono na rys. 4.



Rys. 4. Przykład sieci Petriego

Od opracowania definicji powstało wiele wariantów i rozszerzeń oryginalnej sieci Petriego. Między innymi powstały kolorowane i hierarchiczne sieci Petriego. Oba te warianty zostały wykorzystane przy tworzeniu modelu świata rzeczywistego.

Kolorowana sieć Petriego (ang. coloured Petri net) jest wariantem sieci Petriego należącym do tzw. sieci Petriego wysokiego rzędu (ang. high level Petri nets). Jest to sieć, w której poruszające się znaczniki przenoszą wartości określonego typu. W danym miejscu sieci typ znajdujących się znaczników musi być zgodny z typem tego miejsca. Wzbudzenie tranzycji w bieżącym znakowaniu sieci zależy zarówno od obecności znaczników w miejscach wejściowych tranzycji, jak i od wartości przypisanych do tych znaczników. Zgodnie z tradycyjną terminologią kolorowanych sieci Petriego typ danych nazywa się paletą kolorów, a aktualna wartość przypisana do znacznika – kolorem. Znaczniki różnych kolorów, zgromadzone w pewnym miejscu sieci, tworzą kolekcję znaczników. Liczbę i kolor znaczników usuwanych w chwili wzbudzenia tranzycji definiują wyrażenia wejściowe, przypisane do jej łuków wejściowych. Liczbę i kolor znaczników tworzonych w wyniku wzbudzenia tranzycji definiują wyrażenia wyjściowe, przypisane do jej łuków wyjściowych. Dodatkowy warunek wzbudzenia określa przypisane do tranzycji wyrażenie, nazywane dozorem.

Hierarchiczna sieć Petriego (ang. hierarchical Petri net) jest wariantem sieci Petriego także należącym do tzw. sieci Petriego wysokiego rzędu (ang. high level Petri nets). Jest to sieć, w której pojedynczy element sieci może być opisany modelem w postaci odrębnej podsieci Petriego określającej jego zachowanie. Takim elementem może być sam znacznik. Modele na poszczególnych poziomach hierarchii mogą się ze sobą komunikować.

Sieci Petriego pozwalają na symulację zdarzeń i działań według formalnych zasad działania określonych dla tego rodzaju sieci. Modelowanie za pomocą sieci Petriego charakteryzuje się następującymi cechami:

- Zrozumiały graficzny „język notacji” (miejsca, tranzycje i przejścia między nimi).
- Łatwe do zaimplementowania w postaci symulatora komputerowego za pomocą narzędzi do tworzenia i symulacji kolorowanych sieci Petriego (wykorzystujących platformę java).
- Sposób zapisu bardziej zwarty niż graf stanów, ponieważ stanem symulacji nie jest już pojedyncze miejsce ale układ znaczników w określonych miejscach w modelach.
- Wymaga wieloprzebiegowej, złożonej weryfikacji analogicznie jak program komputerowy, (weryfikacja może być wspierana przez narzędzia informatyczne).
- Brak konieczności odrębnego rozpatrywania stanów wynikających z realizacji procesów współbieżnych – stan gry jest opisywany przez położenie znaczników we wszystkich obiektach i fragmentach obiektów modelu.
- Możliwość budowania modelu w sposób modułowy („obiektywny”), w którym zachowanie każdego uczestnika ćwiczenia i obiektu jest opisane za pomocą odrębnej sieci.
- Komunikacja za pomocą informacji przypisanej do znaczników.

- Hierarchiczna budowa modeli.
- Łatwa weryfikacja działań uczestników gry przez sprawdzenie, w których miejscach znajdowały się znaczniki.
- Konieczność opracowania dodatkowej struktury danych zawierających opis do miejsc i transycji.

4. Technika tworzenia modeli elementów środowiska

Poniżej opisano przyjęte zasady tworzenia modelu środowiska za pomocą kolorowanych sieci Petriego.

Elementami modelu są obiekty odpowiadające rolom jakie pełnią uczestnicy gry w procedurach oraz obiekty środowiska symulacji. Poniżej przedstawiono kroki, jakie należy wykonać w celu utworzenia obiektów odpowiadających symulowanym elementom i uczestnikom gry w procedurach:

- analiza treści procedur pod kątem uczestników procedur i obiektów jakie w nich biorą udział.
- Określenie graczy (ich ról) i obiektów w grze.
- Identyfikacja wszystkich czynności (działań) jakie mogą wykonać uczestnicy gry i stanów w jakich może się znajdować symulowane środowisko.
- Stworzenie sieci Petriego dla poszczególnych ról – model przedstawia wszystkie możliwe zachowania roli w każdej możliwej sytuacji.
- Zamodelowanie komunikacji pomiędzy obiektami.
- Parametryzacja elementów modeli odpowiedzialnych za reprezentację czynników zewnętrznych (np. elementów środowiska) niezależnych od uczestników scenariusza.

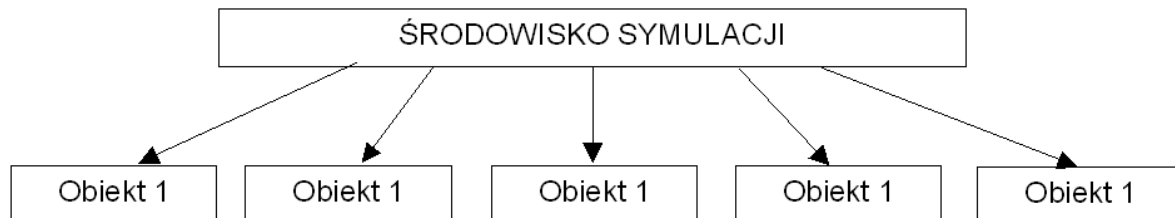
Podczas modelowania należy zwrócić szczególną uwagę na zachowanie zasad wynikających z zastosowania kolorowanych sieci Petriego. Ponieważ modelowane są obiekty reprezentujące poszczególnych uczestników wraz z wszystkimi możliwymi do podjęcia przez nich decyzjami, ich struktura jest bardzo skomplikowana i trudna do weryfikacji. Utrudnieniem może być także liczba znaczników znajdujących się w danej chwili w modelu. Dlatego należy dołożyć starań, aby budowany model był od początku możliwie poprawny.

Produktami, jakie otrzymujemy po wykonaniu powyższych kroków są:

- Zamodelowane obiekty poszczególnych ról, które mogą brać udział w działaniach opisywanych przez procedury.
- Zamodelowane obiekty reprezentujące elementy środowiska uczestniczące w grze.
- Ogólny model sytuacji przedstawiający stan gry i aktywność poszczególnych jej uczestników.

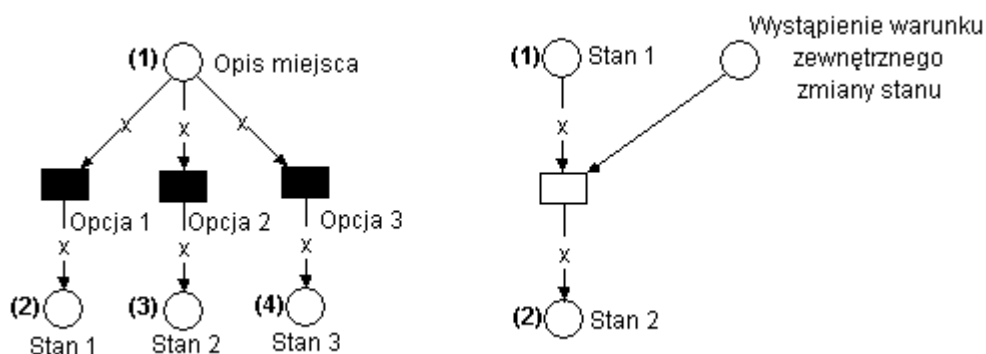
Podczas opracowywania modelu tworzy się strukturę hierarchiczną zależności pomiędzy poszczególnymi obiektami. Struktura zależności pomiędzy obiektami została przedstawiona na Rys. 5, na którym widać, że obiekt na wyższym poziomie hierarchii „powołuje do życia” obiekt na niższym poziomie. Przedstawiony obiekt „środowisko symulacji” jest obiektem nadrzędnym dla obiektów odpowiadających rolom i elementom środowiska w procedurach i jest niezbędny dla ich prawidłowego funkcjonowania. Scala on działanie poszczególnych obiektów przez zamodelowanie przesyłania komunikatów pomiędzy nimi i ogólnych stanów całego środowiska rzeczywistego w tym także i gry. Obiekty Obiekt 1 i Obiekt 2 mogą symulować elementy środowiska rzeczywi-

stego np. magazyn, budynek, firmę, huragan, itp. Zawierają one wszystkie możliwe stany jakie zakłada się, że mogą wystąpić dla poszczególnych elementów, warunki ich wystąpienia, dane wyjściowe jaki mogą generować, itp. Natomiast Obiekt 3, Obiekt 4 i Obiekt 5 mogą reprezentować graczy wykonujących procedurę.



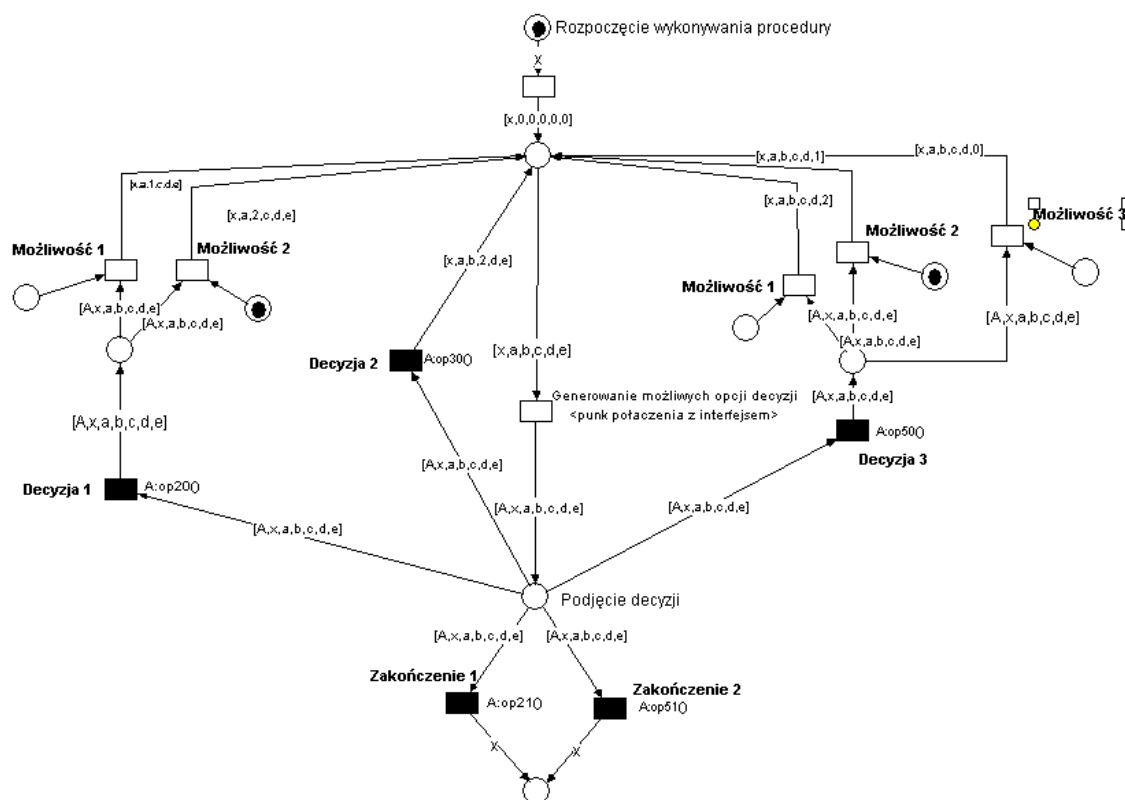
Rys. 5. Hierarchia komunikacji pomiędzy obiektami środowiska symulacji

Decyzje podejmowane przez gracza (różne ścieżki postępowania) realizowane są przez zamodelowanie przejścia z jednego miejsca do wielu. Przykład decyzji został przedstawiony na rys. 6. W sytuacji gdy znacznik znajduje się w miejscu oznaczonym symbolem (1) i spełnione są wszystkie warunki do odpalenia dowolnej z tranzycji związanej z kolorowaną siecią Petriego (tzn. znacznik jest tego samego koloru co tranzycje, nie ma warunków określonych przy tranzycjach uniemożliwiających ich odpalenie przez ten znacznik oraz łuki umożliwiają przeniesienie znacznika danego koloru), gracz ma możliwość wyboru, którą tranzycję odpalić przez podjęcie decyzji. Uruchomienie tranzycji spowoduje przesunięcie znacznika na inne miejsce i nowy stan gry (Stan 1, Stan 2 lub Stan 3). Stan gry może ulec zmianie także po zaistnieniu warunków, które nie zależą od decyzji gracza np. zaistnienie warunku powodującego katastrofę.



Rys. 6. Fragment modelu przedstawiający podejmowanie decyzji i zaistnienie warunku zewnętrznego zmieniającego stan obiektu

Dla kompletnego zamodelowania środowiska rzeczywistego w sposób umożliwiający wykorzystanie modelu przez programistę wymagane jest dodatkowo opracowanie struktury danych zawierającej opisy miejsc i tranzycji, z których korzystać będzie symulator. Przykład takiej struktury danych przedstawiono w Tabeli 1.



Rys. 8. Przykładowy model z decyzjami

5. Podsumowanie

Do tworzenia modeli środowiska rzeczywistego można wykorzystać wiele różnych technik modelowania przedstawionych w niniejszym artykule. Najlepszym podejściem wydaje się zastosowanie techniki modelowania, która będzie spełniała następujące cechy:

- Umożliwi podejście modułowe „obiektywne” do problemu (podzielenie świata na obiekty i relacje pomiędzy nimi).
- Umożliwia tworzenie hierarchii obiektów.
- Da możliwość przedstawienia stanów obiektu, jego parametrów oraz logiki bez potrzeby określania jej w osobnych strukturach danych.
- Zrozumiały i zwarty graficzny język notacji.
- Umożliwia modelowanie działań współbieżnych.
- Integracja z kodem symulatora za pomocą interfejsu (zmiany w modelu nie wymagają zmian w kodzie).

Spośród opisanych w artykule technik modelowania kolorowane sieci Petriego spełniają w/w założenia. Dodatkowym atutem przemawiającym na korzyść kolorowanych sieci Petriego jest duża liczba dostępnych na rynku edytorów i symulatorów tych sieci (np. Renev2, CPN Tool, PN-Sim, JPetriNet, Petri Net Kernel), które w łatwy sposób można zintegrować z tworzonymi aplikacjami. Weryfikacja takiego modelu jest uproszczona przez zastosowanie tychże narzędzi edytorskich. Modelowanie obiektów za pomocą sieci Petriego umożliwia utworzenie pełnego modelu zawierającego obiekty, ich parametry i logikę działania. Edycja poszczególnych elementów

w modelu wymaga jedynie pracy analityka bez potrzeby ingerowania programisty odpowiedzialnego za kod symulatora. Samo modelowanie po opanowaniu podstaw dotyczących tworzenia kolorowanych sieci Pateriego staje się intuicyjne i proste a powstałe diagramy pomimo złożoności reprezentowanych obiektów nie są bardzo skomplikowane przez zastosowanie hierarchii.

Bibliografia

- [1] <http://wikipedia.org> Wolna encyklopedia Wikipedia
- [2] giaur, Tomasz Rostański, Marcinek Drozd: „Teoria gier”
- [3] http://www.weknowhow.pl/205_350.htm
- [4] <http://www.ctpartners.pl/index.php?pageid=98>
- [5] <http://www.door.com.pl>
- [6] <http://tziaja.strony.wi.ps.pl/>
- [7] <http://info.wsisiz.edu.pl/~konopack/bd/bdpojpodst.htm>
- [8] <http://www.omg.org>
- [9] Tadao Murata, IEEE: „Petri Nets: Properties, Analysis and Applications”
- [10] „Petri Nets 2000” 21st International Conference on Application and Theory of Petri Nets
- [11] Michael Weber, Ekkart Kindler: „The Petri Net Markup Language”