dr inż. Stanisław Bocian
Instytut Pojazdów Szynowych „TABOR'

Przyszłościowe współbieżne mikroprocesorowe inteligentne systemy mechatroniczne w sterowaniu i diagnostyce pojazdów szynowych (narzędzia sprzętowe i programistyczne)

W artykule przedstawiono przyszłościowe rozproszone współbieżne mikroprocesorowe inteligentne systemy mechatroniczne w sterowaniu i diagnostyce pojazdów szynowych. Systemy te będą podstawą do tworzenia różnych modeli i układów informatycznych i informacyjnych dla pojazdów szynowych. Artykuł stanowi trzecią część powyższej publikacji. Zawiera opis stosowanych narzędzi sprzętowych i programistycznych w rozwiązywaniu podjętego problemu badawczego.

Artykuł powstał w wyniku realizacji projektu badawczego KBN 4T 12C 04929 pt." Rozproszone współbieżne mikroprocesorowe inteligentne podsystemy mechatroniczne w sterowaniu i diagnostyce pojazdów szynowych".

8. Zastosowane narzędzia sprzętowe i programistyczne w rozwiązywaniu omawianego problemu badawczego

W realizowanym projekcie badawczym wykorzystano następujące narzędzia sprzętowe i programistyczne:

8.1. Mikrosystemy cyfrowe [19]

Mikrosystemem cyfrowym nazywa się układ scalony, który w swej strukturze integruje rdzeń mikroprocesorowy oraz programowalny blok sprzętowy.

W procesie projektowania cyfrowych układów sterowania oprócz stosowania modeli specyfikacji formalnej bardzo ważne jest sprecyzowanie docelowej platformy realizacyjnej:
sprzętowej, programowej i sprzętowo – programowej. Realizacja sprzętowa to struktury układowe małej i średniej skali integracji oraz nowoczesne matryce reprogramowalne. Realizacja programowa to połączenie pewnego zestawu instrukcji (program) oraz odpowiednich struktur sprzętowych (np. mikroprocesor, pamięć) zdolnych do wykonywania określonych działań zapisanych w kodzie tego programu. Zalety i wady obu rozwiązań, tzn. realizacji sprzętowej oraz programowej, rozpatruje się pod względem dwóch podstawowych kryteriów:

- czas reakcji – szybsze rozwiązanie sprzętowe
- koszty realizacji – przyjmuje się niższe dla rozwiązań programowych, ze względu na niższą cenę zarówno samych układów, jak i narzędzi wspomagających proces projektowania.

Własności te wydają się wzajemnie sprzeczne, dlatego należałoby zastosować jednocześnie obie metody stosując kompromis, łączący zalety szybkości działania z niskimi kosztami produkcji. Funkcje takie spełniają zintegrowane w jednej strukturze scalonej **mikrosystemy cyfrowe**. Przykładem są systemy cyfrowe PSoC (Programmable System on Chip) firmy CYPRESS [15]. Układ ten różni się od typowych mikrokontrolerów tym, że posiada programowalne peryferia nie tylko cyfrowe, ale także analogowe. Wybrano tę rodzinę układów ze względu na bogate możliwości programowego tworzenia układów analogowych z rekonfigurowanych bloków. Ponadto dostępne są układy cyfrowe: liczniki, timery, generatory pseudolosowe itp. Komunikację zapewniają moduły RS232, I^2C, USB.

8.2. Skończony automat cyfrowy [16]

Automat skończony A definiuje się za pomocą wzoru 1:

$$A = (X, S, Y, \delta, \lambda, s_o) \quad (1)$$

gdzie:

$X=\{x_1,...,x_i\}$ – skończony niepusty zbiór wejść (alfabet wejściowy)

$S=\{s_1,...,s_j\}$ – skończony niepusty zbiór stanów

$Y=\{y_1,...,y_k\}$ – skończony niepusty zbiór wyjść (alfabet wyjściowy)

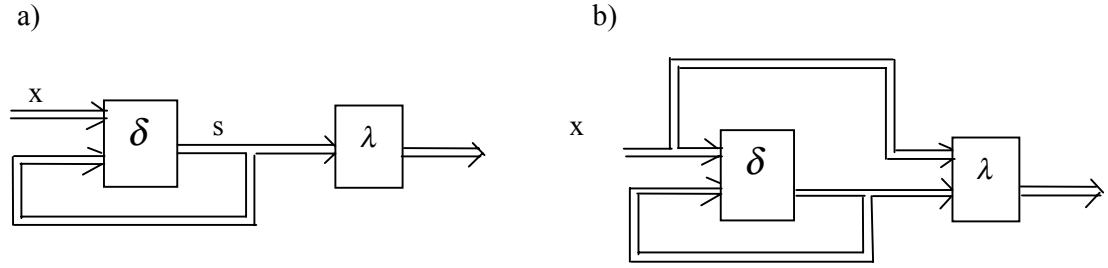
$\delta : S\times X \rightarrow S$ – funkcja przejść (funkcja stanu następnego)

$\lambda : S\times X \rightarrow Y$ – funkcja wyjścia automatu Mealy'go

$\lambda : S \rightarrow Y$ – funkcja wyjścia automatu Moore'a

$s_o \in S$ –wyróżniony stan początkowy, od którego automat zaczyna działanie.

Automat przekształca doprowadzoną informację zgodnie z założonym algorytmem działania (funkcja przejścia) tak, aby na jego wyjściu otrzymać w wyniku informację przekształconą. Automat skończony określany jest na zbiorach skończonych wielkości dyskretnych oraz działa w dyskretnych odcinkach czasu. Rozpatrując automat skończony z punktu widzenia informacyjnego można stwierdzić, że jest to pewnego rodzaju przetwornik informacji.



Rys.5 Graficzna struktura automatu skończonego: a) Mealy'ego, b) Moore'a

Tworzenie oprogramowania dla mikrokontrolerów PSoC firmy CYPRESS jest możliwe poprzez tworzenie programu w oparciu o sporządzony wcześniej graf automatu.

8.3. Sieć Petriego [17 18 i 19]

Sieć Petriego PN definiuje się za pomocą wzoru 2:

$$PN=(P, T, F_o, \kappa, \omega, m_o) \qquad (2)$$

gdzie:

P – jest niepustym, skończonym zbiorem miejsc $P=\{p_1, p_2,...,p_n\}$

T – jest niepustym, skończonym zbiorem tranzycji $T=\{ t_1, t_2,...,t_n\}$

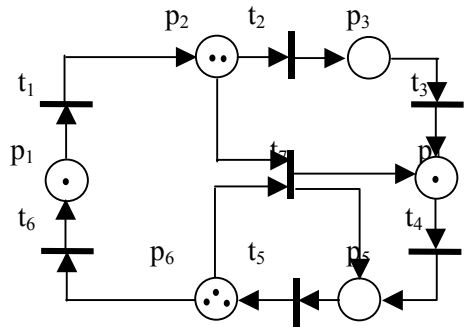
$P \cap T=\phi$ jest zbiorem pustym

F_o – jest niepustym, skończonym zbiorem łuków skierowanych (strzałek), takich że: $F_o \subset (P\times T)\cup(T\times P)$

κ – jest funkcją pojemności miejsc κ: $P \rightarrow N \cup \{?\}$ (N jest zbiorem liczb naturalnych)

ω – jest funkcją wagi łuków $\omega: F_o \rightarrow N$,

m_o – jest funkcją znakowania początkowego m_o: $P \rightarrow N \cup \{0\}$.



Rys. 6 Przykład klasycznej sieci Petriego

Sieć Petriego w ujęciu klasycznym jest dwudzielnym skierowanym grafem, posiadającym dwa rodzaje węzłów: miejsca reprezentowane przez okręgi oraz tranzycje – reprezentowane przez prostokąty lub pogrubione linie. Łuki (strzałki) skierowane łączą miejsca z tranzycjami. Stan sieci określany jest przez znakowanie, charakteryzowane rozmieszczeniem znaczników – punktów wewnątrz miejsc (znaczniki, żetony, markery). Z miejscami związana jest funkcja pojemności miejsca, określająca maksymalną liczbę znaczników jakie może dane miejsce pomieścić, a łukami – funkcja wagi łuków, określająca liczbę znaczników, które jednocześnie mogą się po danym łuku przemieścić.

Dla łatwej reprezentacji współbieżności oraz ze względu na dobrze zdefiniowane pojęcia, sieci Petriego najlepiej nadają się do modelowania układów sterowania dyskretnego.

Układy współbieżne stanowią ważną grupę układów cyfrowych. Współbieżność zdarzeń odzwierciedla się wzajemną niezależnością tych zdarzeń. Opisane za ich pomocą działania układu sterującego sekwencją czynności, wykonywanych współbieżnie jest znacznie prostsze, niż opis funkcjonowania tego samego układu za pomocą innych metod. Ponadto, układy opisane metodami sieci Petriego, dzięki rozbudowanemu aparatowi matematycznemu i dużemu zestawowi metod analitycznych, mogą być weryfikowane w sposób formalny. Dla modelowania układów sterowania dyskretnego ważną rolę odgrywa specyfikacja funkcjonalna układu, która powstaje na podstawie analizy wymagań użytkownika. Zadaniem specyfikacji jest precyzyjne wyrażenie zewnętrznych skutków działania układów cyfrowych. Wykorzystując formalną specyfikację można już we wczesnym studium projektowania wykryć i usunąć błędy. Specyfikacja formalna jednoznacznie określa postulowane działania układu cyfrowego. Stosowanie sieci Petriego jako pośredniej formy specyfikacji pomiędzy opisem w języku naturalnym i specyfikacją logiczna ma wiele zalet. Sieć Petriego jest matematyczną reprezentacją dyskretnych systemów rozproszonych. Przez swoją zdolność do wyrażenia współbieżnych zdarzeń uogólniają one teorię automatów. Najpopularniejszymi modelami przepływu informacji sterowania są automaty skończone i sieci działań. Semantyka tych modeli obejmuje jednak tylko systemy sekwencyjne. Semantyka sieci Petriego wychodzi poza systemy sekwencyjne i umożliwia modelowanie synchronizacji systemów równoległych. Zastosowania sieci Petriego w informatyce obejmują:

- jednoznaczny opis semantyki (np. specyfikacji programu lub protokołu komunikacyjnego)
- makietowanie (symulacja wykonania sieci)

- analizę właściwości, weryfikację (dowodzenie) poprawności programów.

W informatyce sieci Petriego wykorzystuje się do modelowania synchronizacji i komunikacji procesów współbieżnych. W takim zastosowaniu pozycja sieci (stany) interpretuje się zazwyczaj jako warunki programu, a tranzycje jako akcje (instrukcje lub funkcje). Struktura sieci odwzorowuje wtedy strukturę programów, a ruch znaczników w sieci modeluje postępujące wykonanie procesów.

Układy współbieżne w sposób intuicyjny można zapisywać w postaci **interpretowanej sieci Petriego**, która pozwala na interpretacje miejsc analogicznie do stanów w automacie skończonym, a tranzycje do akcji związanych ze zmianą stanu.
Interpretowana sieć Petriego (IPN) definiuje się za pomocą wzoru 3 [19]:

$$IPN=\{P, T, F_o, X, Y, m_o, \delta, \lambda\} \quad (3)$$

gdzie:

P, T, F_o, m_o – definiowane są jak we wzorze 2

X i Y – są alfabetami wejściowymi i wyjściowymi

δ – jest funkcją przyporządkowującą każdej tranzycji pewien podzbiór z przestrzeni wejścia $\delta : T \rightarrow X$

λ – jest funkcją przyporządkowującą każdemu miejscu pewien podzbiór z przestrzeni wyjścia $\lambda : P \rightarrow Y$

W modelu pominięto funkcje pojemności miejsc oraz wagi łuków, gdyż dla sieci interpretowanych stosuje się ich ograniczenia do $\kappa(p)=1$ oraz $\omega(f)=1$.

Interpretowana sieć Petriego jest obrazową formą przedstawienia automatu współbieżnego. W odróżnieniu od klasycznego automatu sekwencyjnego, automat współbieżny znajduje się równocześnie w jednym lub kilku stanach wewnętrznych. Maksymalne zbiory równocześnie występujących stanów lokalnych definiują stany globalne automatu. Dowolny podzbiór równocześnie występujących stanów lokalnych nazywany jest stanem częściowym. W automatach współbieżnych reprezentuje się zamiast globalnych funkcji przejść lokalne relacje, wiążące ze sobą wewnętrzne stany częściowe, aktualne i następne, oraz odpowiednie stany wejść i wyjść automatu.

Układ cyfrowy może być rozpatrywany jako system składający się z części operacyjnej i części sterującej (czyli sterownika logicznego). Przetwarzaniem danych w części operacyjnej steruje sterownik, który rozpoznaje stan systemu i wytwarza odpowiednie sygnały sterujące, synchronizując pracę części operacyjnej. Często pewne dane są przetwarzane w niezależnych procesach, które są wykonywane przez niezależne pracujące części systemu. Dlatego sterowniki muszą być przystosowane do sterowania procesami równoległymi, co powoduje, że opis sterownika z wykorzystaniem sieci Petriego jest bardzo efektywny.
Podstawowym i uniwersalnym narzędziem stosowanym w projekcie badawczym są programowalne sterowniki logiczne PLC (ang. Programmable Logic Controller) firmy PEP, SELECTRON i WAGO. W sterownikach tych stosuje się metody analityczne (programowe) takie jak SFC (ang. Sequential Function Chart – Sekwencyjna tablica funkcji), których podstawy teoretyczne stanowi sieć Petriego.

8.4. Sieci SFC [18]

Sieć SFC jest językiem graficznym, objętym normą IEC [22]. Wykorzystując sieci SFC można przedstawić operacje sekwencyjne w sposób współbieżny. Proces binarny jest reprezentowany za pomocą poprawnie zdefiniowanych kolejnych kroków połączonych z tranzycjami. Elementy sieci SFC są przydatnym narzędziem graficznym opisu sterowników logicznych.
Sieć SFC definiuje się za pomocą wzoru 4:

$$SFC=(S, T, L, I) \quad (4)$$

gdzie:

S – niepusty, skończony zbiór kroków (etapów)

T – niepusty, skończony zbiór tranzycji

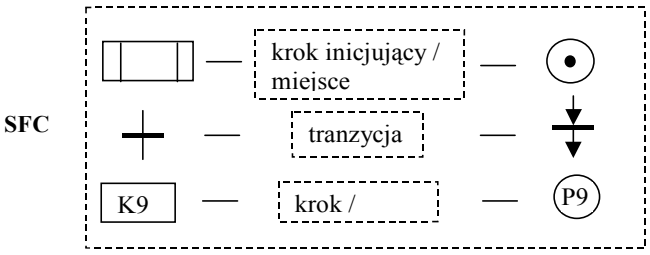
L – niepusty, skończony zbiór połączeń pomiędzy krokiem a tranzycją lub tranzycją a krokiem

$I \subset S$ – zbiór kroków inicjujących

Elementy S i T są reprezentowane jako węzły grafu.

W normie IEC 1131-3 przedstawiono sposób tworzenia struktury wewnętrznej progra- mu w postaci grafu sekwencji SFC, który pozwala na opisywanie zadań sterowania sekwencyj- nego za pomocą grafów zawierającego etapy (kroki) i warunki przejścia (tranzycji) pomiędzy tymi etapami. Z każdym etapem jest skojarzony zbiór odpowiednich działań, a każdemu przejściu między etapami towarzyszy warunek przejścia. Grafy SFC mogą być wykorzysty -wane przy programowaniu sterownika w jednym ze zdefiniowanych w normie IEC:1131-3 języków programowania sterowników PLC. Określono w niej dwie grupy języków programo- wania : języki tekstowe (język listy rozkazów IL i język tekstu strukturalnego ST) i graficzne (język schematu drabinkowego LD i język funkcjonalny schematów blokowych FBD), w celu otrzymania odpowiedniej struktury programu użytkownika. SFC umożliwia modelowanie algorytmów procesu sterowania i stanowi on modyfikację sieci Petriego.

Sposób przedstawienia procesu sterowania za pomocą sieci SFC jest podobny do sposobu reprezentacji tego samego procesu za pomocą interpretowanych sieci Petriego.
Podobne są również reguły modelowania, natomiast inna jest reprezentacja graficzna podstawowych elementów obu sieci: kroków, kroku początkowego, realizacji procesów współbieżnych.



Rys. 7 Porównanie elementów graficznych sieci SFC i sieci Petriego

Zgodnie z normą IEC 1131, zbiór kroków inicjujących został ograniczony do jednego kroku. Krok określający stan procesu, oraz tranzycja, określająca zmianę stanu wraz z warunkiem tej zmiany, są dwoma podstawowymi i niezbędnymi elementami sieci. Wykorzystanie tylko tych dwóch komponentów daje realny obraz działania sterownika. Dodatkowy składnik sieci, taki jak blok działania, ukazuje bardzo szczegółowy, techniczny opis układu. Pozwala określić rodzaj akcji (kwalifikator), symbol zwrotnej zmiennej logicznej lub opisać zachowanie układu sterującego z wykorzystaniem języka ST, IL, a także reprezentację akcji boolowskiej.

8.5. Logika rozmyta

Pojęcie pozbioru rozmytego wprowadził po raz pierwszy L.A.Zadeh [20 i 21] jako uogólnienie pojęcia zbioru zwykłego lub nierozmytego. Podzbiór rozmyty można uważać za predykat (funkcja zdaniowa – stosowana w matematyce i logice matematycznej), którego wartość logiczna należy do przedziału jednostkowego I=[0,1], a nie do zbioru {0,1} jak w przypadku zbioru zwykłego. Podzbiór rozmyty umożliwia opisanie pojęcia, dla którego granica między posiadaniem pewnej własności i jej brakiem jest rozmyta. Logika rozmyta jest uogólnieniem koncepcji logik wielowartościowych i zakłada ocenianie wartości logicznych formuł poprzez ograniczenia narzucone na zbiory ich wartości logicznych. Ograniczenia odpowiadać mają znaczeniom takich pojęć jak „prawdziwe” , „fałszywe” , „raczej prawdziwe”, „raczej fałszywe” itd. [20 i 21]. Podstawowe elementy (człony) sterowania oraz cały system sterowania (np. lokomotywą) można przedstawić jako nieliniową funkcję transmitancji (wejście - wyjście). Funkcje takie dobrze są realizowane w konwencjonalnych układach automatyki z wykorzystaniem regulatora PID (proporcjonalno - całkująco - różniczkującym). W przypadku znacznych zakłóceń zewnętrznych, które powodują nagłe zaburzenie działania układu zaprojektowany klasyczny regulator PID działa albo szybko ze znacznym przesterowaniem albo łagodnie, ale z powolnymi reakcjami. W takim przypadku idealnie nadają się do takiego sterowania inteligentne podsystemy mechatroniczne. Stosując układy sterowania z wykorzystaniem logiki rozmytej (sterowniki rozmyte) stwarza się możliwości prostych, ale odpornych rozwiązań, które obejmują szeroki zakres parametrów systemu i które mogą się uporać z dużymi zakłóceniami. Sterowanie rozmyte oferuje lepszy interfejs użytkownika, które umożliwia „wniknięcie” do nieliniowości urządzenia sterującego za pomocą systemu translacji procesu. Jest to cecha użyteczna i wartościowa, gdyż dotychczas nieliniowości poprawiające działanie były zbyt mało wykorzystywanymi właściwościami regulatorów.

W praktyce podejście rozmyte sprawdza się w odniesieniu do:

- systemów o znacznym stopniu złożoności, utrudniających opracowanie zadowalającego modelu fizycznego lub matematycznego
- przypadków występowania w obiekcie nieliniowości charakterystyk roboczych
- systemów obarczanych niejednoznacznościami bądź na wejściach, bądź w samym opisie.

Sterowanie rozmyte traktuje się jako rozszerzenie istniejącej techniki oraz poszukiwanie rozwiązań hybrydowych, wzmacniając działanie klasycznego sterowania automatycznego. Można połączyć sterowanie rozmyte i sterowanie PID, stosując system PID w pobliżu wartości zadanej w trakcie pracy oraz delinearyzacji systemu w innych obszarach, przez opis żądanego zachowania lub strategii sterowania za pomocą reguł rozmytych. Sterowanie rozmyte dotychczas stosowane do rozwiązywania elementarnych problemów sterowania znajduje obecnie zastosowanie na wyższych poziomach w hierarchii systemów sterowania, monitorowania, diagnozowania i rozwiązywania problemów logistycznych.

W projekcie badawczym wykorzystywane zostaną:

- sterownik rozmyty firmy OMRON
- program MATLAB- SIMULINK z biblioteką Toolbox (Fuzzy Logic).

8.6. Systemy agentowe [14]

Prowadzone w pracy rozważania odnoszą się do klasy inteligentnych systemów zdecentralizowanych, spełniających paradygmaty agentowności. Mogą być to zarówno systemy projektowane jako agentowe (działające w świecie wirtualnym np. systemy usług informatycznych), jak również systemy działające w świecie rzeczywistym (usługi transportowe, złożone systemy przemysłowe).

Agent Λ definiuje się za pomocą wzoru (5):

$$\Lambda = \{A, S, F \subset S \times A \times S\} \quad (5)$$

gdzie:

A – skończony zbiór akcji (działań elementarnych) agenta

S – skończony zbiór stanów wewnętrznych agenta

F –trójczłonowa relacja określająca możliwe następstwa stanów i akcji agenta,

interpretowana w następujący sposób: w określonym stanie agent może wykonywać akcję (drugi człon relacji), która odpowiednio zaprowadzi go do nowego stanu (trzeci człon relacji).

Zachowanie związków przyczynowo – skutkowych wymaga, aby F była taka, że zachodzi

$(s, a, s_1) \in F \wedge (s, a, s_2) \in F \Rightarrow s_1 = s_2$.

Relacja F, wiążąca akcje (niepodzielne) ze stanami, wskazuje nie tylko, do jakiego stanu agenta prowadzi jej wykonanie, ale również określa, jaki podzbiór akcji jest w danym stanie wykonalny (dopuszczalny), czyli możliwości wyboru, przed jakim staje agent. Spoiwem systemu mogą być tylko ich akcje.

Systemem agentowym Ω opisany jest wzorem 6

$$\Omega = \{\{\Lambda^i\}_{i=1,...,N}, I\} \quad (6)$$

gdzie:

$\{\Lambda^i\}_{i=1,...,N}$ jest skończony zbiorem agentów

I jest relacją zwana relacją współdziałania o postaci

$$I = \bigcup_{\substack{i,j = 1,..., N \\ i \neq j}} I^{ij}$$

przy czym: $A^i \times A^j \supset I^{i,j} \ni (a^i, a^j)$: $A^i \in \Lambda^i$; $A^j \in \Lambda^j$; $\Lambda^i, \Lambda^j \in \Omega$.

Relacja I jest symetryczna : $(a^i, a^j) \in I \Rightarrow (a^j, a^i) \in I$.

Relacja współdziałania określa potencjalne możliwości współdziałania agentów w obrębie systemu. Faktyczne współdziałanie następuje, gdy znajdujące się w relacji I akcje są wykonywane przez agentów, przy czym żadna z akcji składowych nie może być wykonywana niezależnie. W konsekwencji, działanie systemu nie może być zdefiniowane jako złożenie działań agentów wchodzących w jego skład.

Agent jest fizyczną lub wirtualną jednostką obliczeniową. Wyposażony jest w sensory – postrzega przez nie swoje otoczenie i efektory – umożliwiające wymianę informacji z otaczającym go środowiskiem i komunikację z innymi agentami. Agent jest jednostką autonomiczną, może działać bez interwencji użytkownika systemu lub innych agentów. Działalność agenta najczęściej jest zdominowana przez dążenie do określonego celu lub wykonanie określonego zadania. System wieloagentowy to system, który zawiera: zbiór agentów $A=\{a_1, a_2, ..., a_n\}$, środowisko, w którym osadzone zostały jednostki agentów, zbiór reguł, które określają interakcje pomiędzy agentami i środowiskiem. Cechy charakterystyczne systemu wieloagentowego:

- żaden z agentów nie posiada pełnych informacji o całym systemie
- kontrola nad systemem jest rozproszona
- dane w systemie są zdecentralizowane
- działanie agentów są asynchroniczne.

Potrzeba budowy rozległych systemów informatycznych i informacyjnych rozproszonych w czasie i przestrzeni, powoduje konieczność tworzenia skutecznych i efektywnych narzędzi umożliwiających zarówno modelowania zachowań takiego systemu jak i późniejszą weryfikacją jego działania. Funkcję taką spełnia język AUML (Agent UML) będący rozszerzeniem już istniejącego standardu modelowania obiektowego, UML (ang.Unified Modeling language – zunifikowany język modelowania) na potrzeby systemów agentowych. Jakkolwiek język AUML wydaje się być bardzo wygodnym sposobem modelowania komunikacji pomiędzy dwoma lub więcej agentami, jednak brak ścisłych rygorów formalnych w połączeniu z wysokim stopniem złożoności diagramów sprawia, że łatwo jest popełnić błąd w trakcie projektowania systemu. Sposobem na wykrywanie błędów w diagramach interakcji AUML może być translacja takich diagramów do sieci Petriego w celu dalszej analizy. Umożliwia to symulację i analizę zachowania komunikujących się agentów programowych oraz wprowadzenie weryfikacji już na poziomie modelu systemu, a nie dopiero na poziomie jego implementacji.

Literatura

[14] Dobrowolski G., Nawarecki E.: Sytuacje kryzysowe w systemach agentowych. Automatyka 2005, T 9, Z 1-2. Katedra Informatyki, Akademia Górniczo-Hutnicza w Krakowie.

[15] Kania D.: Wielokontekstowy sterownik programowalny przyszłości wykorzystujący układy programowalne pSoC. Pomiary Automatyka Robotyka, nr 1/2006.

[16] Kołopieńczyk M.: Zastosowanie konwertera adresów do zmniejszenia rozmiaru pamięci mikroprogramowanego układu sterującego ze współdziałaniem kodów. Praca doktorska, Uniwersytet Zielonogórski, Instytut Informatyki i Elektroniki, czerwiec 2007.

[17] Starke P.H.: Sieci Petri. PWN, Warszawa 1987.

[18] Węgrzyn A.: Symboliczna analiza układów sterowania binarnego z wykorzystaniem wybranych metod analizy sieci Petriego. Praca doktorska, Politechnika Warszawska, Wydział Elektroniki i Technik Informatycznych, 2003.

[19] Wolański P.: Modelowanie układów cyfrowych na poziomie RTL z wykorzystaniem sieci Petriego i podzbioru języka VHDL. Praca doktorska, Politechnika Warszawska, Wydział Elektroniki i Technik Informatycznych, 1999.

[20] Yager R.R., D.P. Filev.: Podstawy modelowania i sterowania rozmytego. WNT Warszawa 1995.

[21] Zadeh L.A.: Fuzzy Sets. Information and Control 8, 1965.

[22] Norma IEC 1131 " Programmable Controllers" wydana w 1993r przez Międzynarodową Komisję Elektroniki (International Electronical Commission IEC).