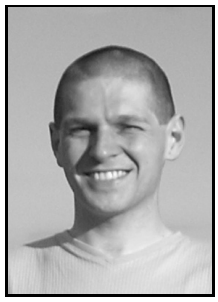
Michał GROBELNY, Iwona GROBELNA
UNIWERSYTET ZIELONOGÓRSKI,
ul. Podgórna 50, 65-246 Zielona Góra

Problem hierarchii w transformacji diagramów aktywności UML 2.x do sieci Petriego sterowania

Mgr inż. Michał GROBELNY



W roku 2007 ukończył studia na Wydziale Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego. Absolwent Zintegrowanych Studiów Zagranicznych Uniwersytetu Zielonogórskiego i Fachhochschule Giessen-Friedberg (Niemcy). Obecnie uczestnik studiów doktoranckich. Zainteresowania naukowe obejmują metody specyfikacji osadzonych systemów sterowania. Członek Polskiego Towarzystwa Informatycznego.

e-mail: M.Grobelny@weit.uz.zgora.pl

Mgr inż. Iwona GROBELNA

Absolwentka Wydziału Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego oraz Fachhochschule Giessen-Friedberg (Niemcy). Od marca 2008 zatrudniona na stanowisku asystenta w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Studentka studiów doktoranckich. Zainteresowania naukowe obejmują metody weryfikacji specyfikacji systemów osadzonych. Członek Polskiego Towarzystwa Informatycznego.

e-mail: I.Grobelna@iie.uz.zgora.pl

Streszczenie

Proces specyfikacji zachowania odgrywa istotną rolę z powodu określania na tym etapie cech i sposobu funkcjonowania sterownika logicznego. W artykule omówione zostały dwie metody graficznego specyfikowania zachowania sterowników logicznych – diagramy aktywności języka UML 2.x oraz sieci Petriego sterowania. Zaproponowana została metoda transformacji hierarchicznych diagramów aktywności do hierarchicznych sieci Petriego. Dzięki przedstawionej metodzie możliwe jest połączenie zalet obu typów graficznego opisu specyfikacji urządzeń. Dodatkowo, zaproponowana została metoda formalnej weryfikacji hierarchicznej formy specyfikacji umożliwiająca wykrycie potencjalnych błędów na tym wczesnym etapie projektu.

Słowa kluczowe: diagramy aktywności UML, sieci Petriego sterowania, modelowanie behawioralne, metody formalnej weryfikacji.

Hierarchy problem in transformation of UML 2.x Activity Diagrams into Control Interpreted Petri Nets

Abstract

Behavior specification is one of the most important steps in embedded systems design. It plays a significant role because system properties and functionality are specified in this phase. There exist some techniques which can be helpful for a designer. In the paper two methods for graphical specification of logic controller behavioral specification [1, 2, 6], namely UML 2.x activity diagrams [10] and control Petri nets [11], are considered. A novel transformation method for transformation of hierarchical activity diagrams into hierarchical Petri nets is proposed. The presented method allows combining the advantages of both types of graphical system specification. Additionally, a formal verification method for hierarchical specification form is proposed. It enables detecting potential errors at early stage of system development. Hierarchical form of specification is commonly used in design process. Activity diagrams can include complex actions (Fig. 1) which describe some subprocesses. Petri nets also support hierarchy, but it can be realized in two different forms [9] – as macroplaces or macrotransitions (Fig. 2). According to the transformation method from [3, 8], actions in activity diagrams are interpreted as transitions in Petri net. In hierarchical processes by means of Petri nets two elements, macrotransitions (Fig. 3) and macroplaces (Fig. 4), have to be considered. The macroplace (Fig. 5a) can be surrounded by two transitions (Fig. 5b), decomposed (Fig. 5c), and finally compressed to macrotransition (Fig. 5d), then transformed into complex activity in an UML 2.x activity diagram (Fig. 5e). Verification of both considered specification forms [3] allows comparing two versions of the same behavior description. The model checking technique [4] can be used to verify the whole system or a part of it. Partial verification can be used for hierarchical specifications, as the verification process can be performed step by step (Fig. 6).

Keywords: UML activity diagrams, control Petri nets, behavioral modelling, formal verification methods.

1. Wprowadzenie

Specyfikacja sterownika logicznego [1, 2, 3] jest jednym z pierwszych i kluczowych elementów procesu projektowania sterownika logicznego. Na tym etapie szczególnie ważne jest, aby specyfikacja spełniała wymagania i założenia projektowe określone przez projektanta, jak również klienta. Specyfikacja może zostać sformalizowana na kilka sposobów, m.in. za pomocą sieci Petriego [4] czy diagramów aktywności (często też nazywanych *diagramami czynności*) języka UML [5]. Diagramy języka UML nie są dotychczas w pełni wpierane przez narzędzia formalnej weryfikacji specyfikacji. Problem ten może zostać rozwiązany przez zastosowanie transformacji do sieci Petriego, która może zostać formalnie zweryfikowana pod kątem spójności pomiędzy opisem modelu a stawianymi założeniami odnośnie funkcjonowania sterownika logicznego.

Zastosowanie metod weryfikacji modelowej (ang. *model checking*) [6] pozwala na wczesne wykrycie niezauważalnych rozbieżności wynikających z niepełnego zrozumienia specyfikacji.

1.1. Diagramy aktywności UML

Notacja UML (ang. *Unified Modelling Language*) w wersji 2.x upraszcza przepływ informacji pomiędzy członkami zespołu projektowego i umożliwia łatwe zrozumienie zachowania systemu nawet osobom bez doświadczenia. Początkowo UML został zaprezentowany jako narzędzie do specyfikacji, wizualizacji i dokumentacji oprogramowania, jednakże bardzo szybko zyskał sobie zwolenników w innych dziedzinach. Systemy osadzone mogą być również z wyśmienitym skutkiem opisywane przy użyciu takich diagramów jak diagramy aktywności, maszyny stanów czy sekwencji.

Obecnie diagramy aktywności są szeroko wykorzystywane w biznesie, modelowaniu przepływu informacji, do opisu właściwości behawioralnych oprogramowania czy systemów osadzonych oraz w współprojektowaniu oprogramowania i sprzętu (ang. *software/hardware co-design*). Często, jak w [5], zamiast nazwy *diagramy aktywności* używana jest nazwa *diagramy czynności*. Obie nazwy są w pełni zamienne.

1.2. Sieci Petriego sterowania

Sieci Petriego zostały zaprezentowane w 1962 przez matematyka Carla Adama Petriego jako model matematyczny ogólnego zastosowania do opisu relacji pomiędzy warunkami i zdarzeniami. Sieci Petriego są obecnie powszechnie używane w wielu dziedzinach przemysłu do planowania oraz kontroli produkcji, projektowania i programowania urządzeń mikroprocesorowych, sterowników logicznych i wielu innych. Dostępne są narzędzia wytwarza-

jące automatycznie kod programu na podstawie specyfikacji stworzonej w sieci Petriego.

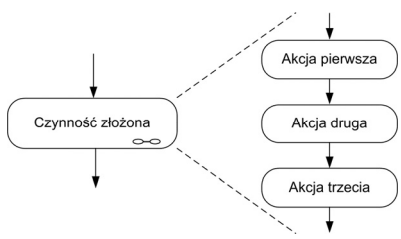
Graficzna reprezentacja sieci Petriego przedstawia modelowany proces w sposób zwarty i jednoznaczny. Przy wykorzystaniu tej techniki możliwe jest zaprojektowanie procesów zawierających równoległość i współbieżność, wybór, synchronizację, zapis, odczyt danych czy współdzielenie dostępnych zasobów.

Interpretowane Sieci Petriego Sterowania (ang. *Control Interpreted Petri Nets*) opisują zachowanie współbieżnych sterowników logicznych [2, 7] biorąc pod uwagę właściwości kontrolowanych obiektów. Stany lokalne mogą się zmieniać po uruchomieniu (odpaleniu) tranzycji, jeżeli wystąpią oczekiwane zdarzenia. Warunki tranzycji są skojarzone z sygnałami wejściowymi sterownika, a miejsca są powiązane z jego sygnałami wyjściowymi. Stan globalny sterownika logicznego zbudowany jest na podstawie jego wszystkich stanów lokalnych.

2. Specyfikacja hierarchiczna

Specyfikacja zachowania urządzenia jest niezwykle istotnym elementem całego projektu. Ważne jest, aby owa specyfikacja była zrozumiała i bardzo czytelna zarówno dla samego projektanta, jak i osób realizujących oprogramowanie i sprzęt zgodnie ze specyfikacją zleceniodawcy. Specyfikacja musi przedstawiać wszystkie elementy behawioralne projektowanego systemu. Nawet najprostsze urządzenia w obecnych czasach realizują dziesiątki operacji. Każda taka operacja niejednokrotnie składa się z kilku akcji. Szczególnie istotną rolę odgrywa możliwość dekompozycji oraz podzielenia specyfikacji. Z pomocą idą tutaj techniki hierarchicznej specyfikacji. Metody takie obecne są w obu omawianych technikach specyfikacji graficznej. Istnieją jednakże pewne rozbieżności pomiędzy technikami dekompozycji dostępnymi w diagramach aktywności języka UML 2.x i sieciach Petriego. Część z nich znacząco komplikuje transformację, co zostało opisane w dalszych akapitach.

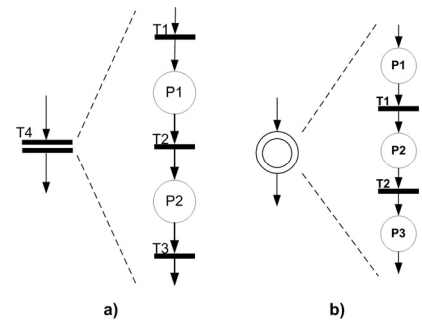
W diagramach aktywności języka UML 2.x akcja przedstawia elementarną jednostkę specyfikacji zachowania, jaka może być realizowana przez projektowany system. Z założenia trwa ona nieskończenie krótko. Specyfikacja systemu niejednokrotnie przedstawia skomplikowane czynności reprezentujące złożone procesy. Mogą one zostać zobrazowane w postaci jednej czynności [5], na którą składa się wiele akcji. Przykład takiej czynności złożonej oraz diagramu przedstawiającego zdekomponowane akcje pokazany został na rysunku 1.



Rys. 1. Czynność złożona
Fig. 1. Complex activity

Sieci Petriego także posiadają odpowiednie mechanizmy do opisu złożonych procesów w czytelny sposób z wykorzystaniem struktur hierarchicznych. W literaturze opisywane są dwie możliwości modelowania hierarchicznych sieci Petriego – poprzez makromiejsca i makrotranzycje [7]. Przykładowa makrotranzycja oraz jej postać zdekomponowana została przedstawiona na rysunku 2a, natomiast makromiejsce przedstawia rysunek 2b. Oba sposoby różnią się nieznacznie pomiędzy sobą. Elementy diagramu odpowiadające makrotranzycji rozpoczynają się i kończą zawsze od tranzycji, natomiast w przypadku makromiejsca odpowiadający mu fragment sieci zaczyna się od miejsca i miejscem kończy.

Z powodu dwóch sposobów obrazowania hierarchicznych sieci Petriego transformacja z diagramów aktywności do sieci Petriego komplikuje się.

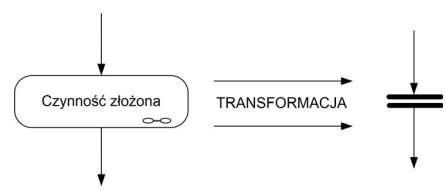


Rys. 2. a) Makrotranzycja; b) makromiejsce
Fig. 2. a) Macrotransition; b) macroplace

3. Transformacja

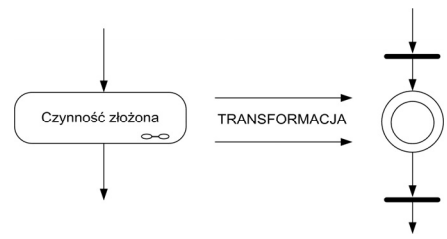
Zgodnie z przyjętą metodologią transformacji diagramów aktywności UML 2.x do sieci Petriego sterowania akcjom diagramu aktywności przypisywane są tranzycje sieci Petriego [8, 9, 10]. Jest to odmienne podejście do stosowanego do tej pory, gdzie akcje diagramu aktywności transformowane są do miejsc sieci Petriego [11]. Przyjmując dwie możliwości modelowania hierarchii w sieciach Petriego rozważane są dwa sposoby odwzorowania hierarchii w transformacji.

Pierwszym przyjmowanym typem tłumaczenia diagramów aktywności UML 2.x do sieci Petriego jest odwzorowanie czynności złożonej do makrotranzycji (rys. 3). Można przyjąć, że wynika to w sposób naturalny z transformacji akcji na tranzycje. Tłumaczenie w omawiany sposób naturalnie prowadzi też do odwzorowania diagramów zdekomponowanych.



Rys. 3. Transformacja czynności złożonej na makrotranzycję
Fig. 3. Transformation of complex activity into macrotransition

Drugim z możliwych sposobów modelowania procesów zachodzących w systemie w sposób hierarchiczny jest przedstawienie złożonych czynności w postaci makromiejsc. Przyjąć można także w transformacji, że czynność złożona diagramu aktywności UML 2.x zostanie odwzorowana przez makromiejsce otoczone dwoma tranzycjami (rys. 4).

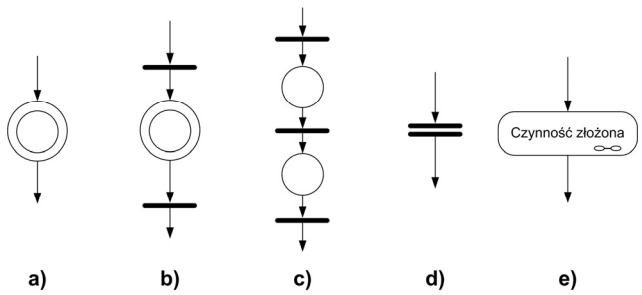


Rys. 4. Alternatywny sposób transformacji czynności złożonej na makromiejsce z tranzycjami
Fig. 4. Transformation of complex activity into macroplace with transitions

Sposób takiego odwzorowania wymuszony jest przez fakt, iż odpowiednik miejsca oraz makromiejsca nie jest elementem jawnie występującym w diagramach aktywności. Zabieg dodania dwóch otaczających tranzycji powoduje, że niejako pośrednio pojawia się makrotranzycja zbudowana z dwóch tranzycji i ma-

kromiejsca. Omawiany sposób odwzorowania czynności złożonej zaproponowany jest ze względu na kompatybilność z sieciami Petriego, w których występują makromiejsca. Nasuwa się jednak pewne spostrzeżenie, które komplikuje transformacje w obie strony. Problem ten wynika właśnie z braku elementu odpowiadającego miejscu i makromiejscu w diagramach aktywności.

Transformacja w omawiany przez autorów sposób z założenia jest transformacją obustronną, z diagramów aktywności do sieci Petriego i z sieci Petriego do diagramów aktywności. Zakładając taki schemat i zakładając pełną kompatybilność z sieciami Petriego należy rozważyć następujący przypadek.



Rys. 5. Kolejne kroki transformacji: a) makromiejsce; b) makromiejsce z dodatkowymi tranzycjami; c) sieć po dekompozycji; d) makrotranzycja; e) czynność złożona
Fig. 5. Steps of transformation: a) macroplace; b) macroplace with additional transitions; c) net after decomposition; d) macrotransition; e) complex activity

Sieć Petriego, która w omawianym przypadku jest elementem źródłowym, zbudowana jest z co najmniej jednego procesu złożonego zamodelowanego przy wykorzystaniu makromiejsca (rys. 5a). Otaczając to makromiejsc dwoma dodatkowymi tranzycjami (rys. 5b) niewpływającymi na sam modelowany proces i dekomponując makromiejsce (rys. 5c) otrzymana zostaje makrotranzycja (rys. 5d). Proces dekompozycji realizowany jest w celu zapewnienia poprawnej transformacji pamiętając, że makromiejsce nie posiada swojego odpowiednika w diagramach aktywności. Otrzymana makrotranzycja może zostać przetłumaczona zgodnie z wcześniejszymi założeniami na czynność złożoną (rys. 5e).

Zgodnie z wcześniej przyjętymi założeniami sieć Petriego zawierająca makromiejsce była siecią źródłową. Zakładając, że w docelowym diagramie aktywności zostały naniesione poprawki i zgodnie z przyjmowaną zasadą, diagram zostaje ponownie przetłumaczony na sieć Petriego w celu dalszej pracy nad projektem. Pojawia się tutaj przytaczany wcześniej problem z istnieniem w sieciach Petriego dwóch rodzajów modelowania hierarchicznego. Przy tłumaczeniu w celu powrócenia do źródłowej sieci Petriego nie wiadomo, na którym etapie należy zakończyć transformację. Czy transformacja powinna się zakończyć otrzymaniem makrotranzycji (rys. 5d), czy makromiejsca bez dodatkowych zbędnych w tym przypadku tranzycji (rys. 5a). Z założenia owa różnica nie zmienia zachowania modelowanego procesu, ale może wprowadzić niepotrzebne komplikacje, przy dalszej pracy z siecią Petriego. Komplikacje polegają na konieczności wprowadzenia zmian niezbędnych do uzyskania pożądanego efektu.

Wnioskując z omawianego przykładu należy dodać dodatkowe elementy lub znaczniki, które będą w jednoznaczny sposób określały jaki typ hierarchii w danym momencie jest transformowany, aby przy transformacji wstecznej otrzymać prawidłową sieć Petriego. Należy także uwzględnić fakt ewentualnych rozbieżności przy interpretacji omawianych sposobów hierarchicznej prezentacji czynności i miejsc złożonych. Należałoby rozważyć fakt, czy makromiejsce źródłowej sieci Petriego może zostać zawsze odwzorowane w postaci czynności złożonej z wcześniej dodanymi neutralnymi tranzycjami. Ważnym aspektem jest także wątpliwość, czy taki sposób odwzorowania nie zmieni sposobu odczytania modelowanego procesu przez odbiorców docelowych. Może to doprowadzić do błędnego zrozumienia procesu, wykonania zbędnych poprawek, które docelowo mogą wprowadzić błędy.

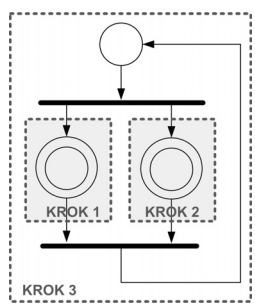
Omawiany aspekt niejednoznaczności hierarchii sieci Petriego oraz wynikający z tego problem przy transformacji będzie elementem dalszych badań autorów.

4. Weryfikacja

Weryfikacja wyników transformacji [8] pozwala na zachowanie spójności pomiędzy diagramami obu omawianych technik specyfikacji zachowania sterownika logicznego. Dodatkowo, pozwala ona na wykrycie błędów, które mogły powstać w trakcie tworzenia diagramu, jeszcze przed jego transformacją, a związane są z niewłaściwym zrozumieniem działania projektowanego sterownika logicznego bądź też nieuwzględnieniem wpływu środowiska.

Technika weryfikacji modelowej [6] może być wykorzystana do weryfikacji całego systemu, bądź też jego części. Proste systemy można projektować, modelować, weryfikować i implementować od razu w całości. Jednakże, w przypadku systemów złożonych, gdzie sam proces projektowania jest zadaniem czasochłonnym, warto wykorzystać możliwość częściowej weryfikacji (ang. *partial verification*). Proces projektowania może być wtedy podzielony na kilka etapów, z których każdy będzie niezależnie weryfikowany.

Hierarchiczną interpretowaną sieć Petriego można podzielić na kilka części, wydzielając z niej odpowiednie makromiejsca. Każde makromiejsce zawierające przynajmniej kilka miejsc i tranzycji opisujące proces współbieżny, może być następnie oddzielnie weryfikowane. Sporządzony na podstawie makromiejsca model podsystemu będzie opisywał proste operacje w nim zachodzące (krok 1 i krok 2 na rys. 6). Weryfikacja modelowa takiego makromiejsca potwierdzi, czy spełnia ono stawiane mu wymagania. Makromiejsca, które zawierają niewiele miejsc i tranzycji i nie wymagają osobnej weryfikacji mogą zostać rozwinięte do pełnych rozmiarów. Po pomyślnym zweryfikowaniu wybranych makromiejsc możliwe jest przeprowadzenie weryfikacji na całej sieci Petriego (krok 3 na rys. 6). Zweryfikowane wcześniej makromiejsca (krok 1 i krok 2 na rys. 6), w miarę możliwości, mogą być traktowane jako stany złożone bez wnikania w ich wewnętrzną strukturę. Możliwe jest wtedy sprawdzenie, czy cały zaprojektowany system spełnia stawiane mu wymagania behawioralne.



Rys. 6. Weryfikacja hierarchicznej sieci Petriego sterowania
Fig. 6. Verification of hierarchical control Petri net

Występują jednak takie sytuacje, gdy weryfikacja makromiejsc nie wnosi nam nic nowego do projektu (proste makromiejsca), a sieć Petriego z makromiejscami nie daje możliwości sprawdzenia wszystkich właściwości. Wtedy jedynym wyjściem wydaje się rozwinięcie wszystkich makromiejsc i weryfikacja kompletnej sieci Petriego.

5. Podsumowanie

Hierarchiczna specyfikacja zachowania sterownika logicznego w znaczny sposób usprawnia proces projektowania. Język UML, a wraz z nim diagramy aktywności, stają się coraz bardziej popularnym narzędziem projektantów. Sieci Petriego, posiadając duży zasób technik i rozwiązań dedykowanych wspomagających proces projektowania, cieszą się niemniej dużą grupą zwolenników. Połączenie tych obu wymienionych metod daje jeszcze większe

możliwości osobom pracującym przy projektowaniu sterowników logicznych. Zastosowanie metod weryfikacji modelowej przyczynia się zaś do zwiększenia jakości projektowanych urządzeń i dzięki temu umocnienia pozycji przedsiębiorstwa na rynku. Metody transformacji i weryfikacji hierarchicznych struktur mają na celu usprawnienie przepływu informacji w zespole oraz zwiększenie niezawodności projektowanych urządzeń. Jednakże przedstawiony w artykule problem niejednoznaczności hierarchii w sieciach Petriego znacząco komplikuje transformację.

Iwona Grobelna jest stypendystą w ramach Poddziałania 8.2.2 „Regionalne Strategie Innowacji", Działania 8.2 „Transfer wiedzy", Priorytetu VIII „Regionalne Kadry Gospodarki" Programu Operacyjnego Kapitał Ludzki współfinansowanego ze środków Europejskiego Funduszu Społecznego Unii Europejskiej i z budżetu państwa.

6. Literatura

[1] Adamski M., Chodań M.: Modelowanie układów sterowania dyskretnego z wykorzystaniem sieci SFC, Zielona Góra: Wydawnictwo Politechniki Zielonogórskiej, 2000.
[2] Adamski M., Karatkevich A., Węgrzyn M. (ed.): Design of embedded control systems, Springer (USA), 2005.
[3] Gomes L., Barros J.P., Costa A.: Modeling formalisms for embedded system design, Embedded Systems Handbook, Taylor & Francis Group, LLC, 2006.
[4] David R., Alla H.: Petri Nets & Grafcet. Tools for modeling discrete event systems, Prentice Hall, 1992.
[5] Wrycza S., Marcinkowski B., Wyrzykowski K.: Język UML 2.0 w modelowaniu systemów informatycznych, Gliwice: Helion, 2005.
[6] Emerson E.A.: The beginning of model checking: a personal perspective, Lecture Notes in Computer Science, 25 Years of Model Checking: History, Achievements, Perspectives, 2008, ss. 27-45.
[7] Węgrzyn A.: Symboliczna analiza układów sterowania binarnego z wykorzystaniem wybranych metod analizy sieci Petriego, Zielona Góra: Oficyna wydawnicza UZ, 2003.
[8] Grobelna I., Grobelny M., Adamski M.: Petri Nets and activity diagrams in logic controller specification – transformation and verification, Mixed Design of Integrated Circuits and Systems – MIXDES 2010, Wrocław, Polska, 2010, ss. 607-612.
[9] Grobelny M., Grobelna I.: Diagramy aktywności języka UML i sieci Petriego w systemach sterowania binarnego – od transformacji do weryfikacji, Pomiary Automatyka Kontrola, nr 10, ss. 1154-1158, 2010.
[10] Staines T.S.: Intuitive Mapping of UML 2 Activity Diagrams into Fundamental Modeling Concept Petri Net Diagrams and Colored Petri Nets, 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, 2008, ss. 11 200.
[11] Basile F., Chiachio P., Del Grosso D.: Modelling automation systems by UML and Petri Nets, Proceedings of the 9th International Workshop on Disscreet Event Systems, Goteborg, Sweden, 2008, ss. 308-313.

otrzymano / received: 15.04.2011
przyjęto do druku / accepted: 06.06.2011 *artykuł recenzowany*

INFORMACJE

Informacje dla Autorów

Redakcja przyjmuje do publikacji tylko prace oryginalne, nie publikowane wcześniej w innych czasopismach. Redakcja nie zwraca materiałów nie zamówionych oraz zastrzega sobie prawo redagowania i skracania tekstów oraz streszczeń.

Artykuły naukowe publikowane w czasopiśmie PAK są formatowane jednolicie zgodnie z ustaloną formatką zamieszczoną na stronie redakcyjnej www.pak.info.pl. Dlatego artykuły przekazywane redakcji należy przygotowywać w edytorze Microsoft Word 2003 (w formacie DOC) z zachowaniem:

- wielkości czcionek,
- odstępów między wierszami tekstu,
- odstępów przed i po rysunkach, wzorach i tabelach,
- oznaczeń we wzorach, tabelach i na rysunkach zgodnych z oznaczeniami w tekście,
- układu poszczególnych elementów na stronie.

Osobno należy przygotować w pliku w formacie DOC notki biograficzne autorów o objętości nie przekraczającej 450 znaków, zawierające podstawowe dane charakteryzujące działalność naukową, tytuły naukowe i zawodowe, miejsce pracy i zajmowane stanowiska, informacje o uprawianej dziedzinie, adres e-mail oraz aktualne zdjęcie autora o rozmiarze 3,8 x 2,7 cm zapisane w skali odcieni szarości lub dołączone w osobnym pliku (w formacie TIF).

Wszystkie materiały:

- artykuł (w formacie DOC),
- notki biograficzne autorów (w formacie DOC),
- zdjęcia i rysunki (w formacie TIF lub CDR),

prosimy przesyłać w formie plików oraz dodatkowo jako wydruki na białym papierze (lub w formacie PDF) na adres e-mail: wydawnictwo@pak.info.pl lub pocztą zwykłą, na adres: Redakcja Czasopisma Pomiary Automatyka Kontrola, Asystent Redaktora Naczelnego mgr Agnieszka Skórkowska, ul. Akademicka 10, p.21A, 44-100 Gliwice.

Wszystkie artykuły naukowe są dopuszczane do publikacji w czasopiśmie PAK po otrzymaniu pozytywnej recenzji. Autorzy materiałów nadesłanych do publikacji są odpowiedzialni za przestrzeganie prawa autorskiego. Zarówno treść pracy, jak i wykorzystane w niej ilustracje oraz tabele powinny stanowić dorobek własny Autora lub muszą być opisane zgodnie z zasadami cytowania, z powołaniem się na źródło cytatu.

Przedrukowywanie materiałów lub ich fragmentów wymaga pisemnej zgody redakcji. Redakcja ma prawo do korzystania z utworu, rozporządzania nim i udostępniania dowolną techniką, w tym też elektroniczną oraz ma prawo do rozpowszechniania go dowolnymi kanałami dystrybucyjnymi.