Michał GROBELNY
UNIWERSYTET ZIELONOGÓRSKI, WYDZIAŁ ELEKTROTECHNIKI, INFORMATYKI I TELEKOMUNIKACJI

Transformacja diagramów aktywności UML 2.0 do sieci Petriego w systemach sterowania binarnego

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). Uczestnik studiów doktoranckich. Zainteresowania naukowe obejmują metody specyfikacji osadzonych systemów sterowania.

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

Streszczenie

Artykuł zawiera omówienie oraz porównanie dwóch formalnych metod specyfikacji behawioralnej systemów osadzonych - diagramów aktywności języka UML 2.0 oraz sieci Petriego. Przedstawione są podobieństwa oraz różnice pomiędzy nimi. Poruszone jest zagadnienie transformacji pomiędzy dwoma wymienionymi technikami specyfikacji wraz z obrazującym ten proces przykładem. Uwzględniono także problem nadmiarowości sieci Petriego po bezpośredniej transformacji i wynikającą z tego konieczność redukcji miejsc i tranzycji przy zachowaniu jednoznaczności obu diagramów. Artykuł podzielony jest następująco. Rozdział 2 zawiera wprowadzenie do diagramów aktywności w UML 2.0. Rozdział 3 przedstawia sieci Petriego. Rozdział 4 porównuje obie metody specyfikacji - diagramy aktywności oraz sieci Petriego. Rozdział 5 porusza zagadnienie transformacji pomiędzy dwoma omawianymi technikami. Rozdział 6 zawiera podsumowanie wcześniejszych rozdziałów oraz wnioski.

Słowa kluczowe: diagramy aktywności UML 2.0, sieci Petriego.

Transformation of UML 2.0 activity diagrams into Petri nets in binary control systems

Abstract

The paper presents and compares two formal behavioural specification methods of embedded systems [1] – activity diagrams of UML specification language [2, 3, 4, 13, 14] and Petri nets [9, 15]. Similarities and differences as well as the aspect of transformation between both specification techniques are concerned. The transformation is explained on the example of a sample control process shown in Fig. 1. Fig. 2 presents the specification using UML activity diagram. The problem of redundant places and transitions after direct transformation from the activity diagram into the Petri net is dealt with. The Petri net after transformation is shown in Fig. 3 (a), while the reduced diagram - in Fig. 3 (b). The paper is divided into sections. Section 1 contains introduction to the topic of embedded system specification techniques. Section 2 presents UML 2.0 activity diagrams with their basic elements. Petri nets and their syntactic are described in Section 3. Section 4 focuses on comparison of both techniques by means of embedded control systems (see [12] for more details). The transformation problem is considered in Section 5. Section 6 summarises and concludes the paper.

Keywords: UML 2.0 activity diagrams, Petri nets.

1. Wstęp

Etap specyfikacji jest niezmiernie istotny w procesie powstawania systemów osadzonych, szczególnie, jeżeli brane są pod uwagę bezpieczne systemy wbudowane [1]. Poprawny projekt nowego urządzenia jest pierwszym krokiem do jego realizacji zakończonej sukcesem. Etap ten może zostać zrealizowany przy wykorzystaniu różnych metod specyfikacji i dokumentacji.

Notacja UML (ang. *Unified Modelling Language*) [2, 3 ,4, 13, 14] usprawnia przepływ informacji pomiędzy członkami zespołu i umożliwia lepsze zrozumienie zachowania systemu. Notacja ma przejrzystą formę i jest zrozumiała nawet dla osób nie zajmujących się na co dzień informatyką. Behawioralny projekt systemu osadzonego [1] można sporządzić przy wykorzystaniu wybranych typów diagramów UML, takich jak diagramy aktywności, maszyny stanów czy też diagramów sekwencji.

Sieci Petriego są inną metodą specyfikacji behawioralnej [9, 15]. Przedstawiają one w zwięzły sposób relacje zachodzące w procesach sterowania.

2. Diagramy aktywności UML

Notacja UML została wprowadzona przez konsorcjum OMG (*Object Management Group*) jako język służący do specyfikacji, wizualizacji i dokumentacji oprogramowania [3].

Notacja UML w wersji 2.0 wprowadziła nowe typy diagramów, a także wiele usprawnień do już istniejących, m.in. do diagramów aktywności, które stały się semantycznie podobne są do sieci Petriego.

Diagramy aktywności są powszechnie wykorzystywane w dziedzinie biznesu czy modelowaniu przepływu informacji [5, 6], a także w behawioralnym projektowaniu oprogramowania oraz systemów osadzonych we współbieżnym projektowaniu sprzętu i oprogramowania (software/hardware co-design) [7]. Systemy osadzone są zwykle definiowane jako zbiór aktywności wykonywanych przez zewnętrznych aktorów lub stany wewnętrzne. W odróżnieniu do innych technik modelowania, diagramy aktywności nie koncentrują się na jednym obiekcie lub obszarze, lecz na ogólnym przepływie akcji [8].

Podstawowe elementy diagramów aktywności obejmują akcje (graficznie przedstawione, jako zaokrąglone prostokąty z nazwą akcji) oraz czynności, które prezentowane są graficznie w identyczny sposób. Różnicą pomiędzy nimi jest fakt, iż czynności są bardziej złożone od akcji i można je zagnieżdżać (hierarchia). Kolejnymi elementami są przepływy pomiędzy akcjami i czynnościami przedstawione przy pomocy strzałki, węzeł początkowy (wypełniona kropka) oraz węzeł końcowy (wypełniona kropka z otoczką). Dodatkowo, zwłaszcza przy systemach osadzonych, może pojawić się procesy współbieżne. Opisywane są one przy wykorzystaniu tranzycji *fork* (początek współbieżnych procesów) oraz *join* (koniec współbieżnych procesów, poziome lub pionowe kreski). Należy tutaj zauważyć, iż tranzycja *join* pełni dodatkowo rolę synchronizacyjną – wykonywana jest w momencie osiągnięcia jej przez wszystkie procesy. Współbieżne przypływy danych mogą również zostać zredukowane do pojedynczego przepływu przy wykorzystaniu węzła łączącego *merge* (romb), co stanie się za każdym razem, gdy którykolwiek z procesów osiągnie dany punkt. Decyzje (także romb) posiadają jedno wejście, ale dowolną liczbę wyjść, zaś ich warunki określone są w kwadratowych nawiasach. Wymienione w tym rozdziale elementy diagramów aktywności UML 2.0 zostały uznane za niezbędne, ale jednocześnie wystarczające do opisu procesów sterownia i te elementy będą wykorzystywane w dalszej części pracy.

3. Sieci Petriego

Sieci Petriego są techniką modelowania wprowadzoną w latach 60-tych ubiegłego stulecia. Opisują one relacje pomiędzy warunkami i zdarzeniami w procesach sterowania [9] wykorzystując graficzną notację.

Składnia sieci Petriego obejmuje graficzne elementy przedstawiające miejsca (okrąg), tranzycje (kreski), przepływy (strzałki) i tokeny (kropki). Sieć Petriego posiada pewną skończoną liczbę miejsc i tranzycji. Miejsca reprezentują stany systemu, tranzycje akcje lub procesy pomiędzy dwoma stanami, zaś tokeny zaznacza-

ją aktualne stany systemu. Przy inicjalizacji systemu token znajduje się w miejscu oznaczonym jako miejsce początkowe.

Współbieżność przedstawiana jest przy wykorzystaniu tranzycji. Początek współbieżnych procesów zaznaczany jest przez tranzycję z jednym wejściem a wieloma wyjściami, zaś ich koniec – przez tranzycję z wieloma wejściami a jednym wyjściem.

4. Porównanie diagramów aktywności i sieci Petriego

Diagramy behawioralne odgrywają ważną rolę, ponieważ opisują dynamikę systemów [10]. Zarówno diagramy aktywności UML 2.0, jak i sieci Petriego, nadają się do behawioralnego modelowania systemów dyskretnych. Posiadają one mechanizmy służące do opisu współbieżności, synchronizacji czy też sekwencyjności zadań. Sieci Petriego wydają się być lepsze do opisu nisko-poziomowego. Są one jednak bardziej skomplikowane i mogą zawierać więcej węzłów i krawędzi [11].

Diagramy aktywności są bardziej przyjazne dla użytkowników, co nabiera szczególnego znaczenia w projektach wymagających konsultacji z klientem nieposiadającym wiedzy technicznej.

Sieci Petriego są używane do opisywania i wizualizacji zachowania, współbieżności oraz synchronizacji skomplikowanych procesów sterowania. Opis jest często nisko-poziomowym modelem opartym na sygnałach i stanach systemu. Bardziej formalna składnia oferuje lepszy i bardziej dopasowany opis akcji realizowanych przez system. Czasem sieć może przyjąć skomplikowaną postać, lecz ważne jest, że uwzględnia wszystkie sygnały i stany. Ponadto, ubogość składni sieci Petriego sprawia, że stają się one mniej podatne na błędy wynikające z interpretacji zachowania danego elementu. Miejscem potencjalnego błędu w diagramie aktywności mogą być węzły *join* i *merge*. Oba węzły ogrywają z pozoru taką samą rolę, lecz ich znaczenie jest różnie, jeżeli rozpatruje się synchronizację procesów.

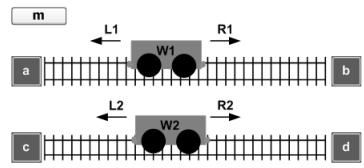
Inną ważną różnicą pomiędzy sieciami Petriego a diagramami aktywności UML są miejsca synchronizacji w sieciach Petriego. Mogą się one wydawać nadmiarowe, jednakże jasno wskazują stan systemu, kiedy jeden ze współbieżnych procesów musi oczekiwać na pozostałe. Diagramy aktywności nie posiadają takich akcji, a stany oczekiwania są domniemane przy węzłach *join*, gdzie wszystkie współbieżne procesy muszą oczekiwać na siebie wzajemnie. Różnica ta może również powodować problemy przy transformacji pomiędzy omawianymi technikami. Bezpośrednie wierne odwzorowanie systemu pomijające owe miejsca oczekiwania nie jest zatem możliwe.

Podsumowując, obie omawiane techniki nadają się do modelowania zachowania systemów dyskretnych. Mają wiele wspólnych cech, co można wytłumaczyć faktem, że diagramy aktywności w UML 2.0 były wzorowane na sieciach Petriego. Każda z technik posiada swoje zalety i wady, a wybór pomiędzy nimi wydaje się być zależnym tylko od preferencji projektanta systemu. Głębsza analiza podobieństw i różnic pomiędzy wymienionymi technikami przedstawiona została w [12].

5. Transformacja

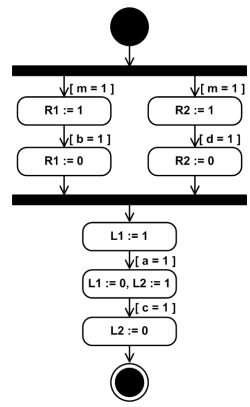
Język UML jest bardzo rozpowszechnioną technologią w wielu dziedzinach, jednakże jest to nadal młoda technika posiadająca wiele ograniczeń. W porównaniu do sieci Petriego obecnych od ponad 40 lat w sferze projektowania systemów sterownia język UML nie może pochwalić się wieloma mechanizmami weryfikacji formalnej. Stąd istnieje potrzeba transformacji specyfikacji opisanej przy pomocy diagramów aktywności języka UML do sieci Petriego. Jej celem jest możliwość zweryfikowania zgodności modelu i jego zachowania z założeniami poczynionymi w początkowych fazach projektu.

Przykład zaczerpnięty został z [18] i zawiera opis procesu sterowania ruchem dwóch wózków. W chwili początkowej oba wózki znajdują się w punktach *a* i *c*. Po naciśnięciu przycisku *m* rozpoczyna się ich współbieżny przejazd w prawą stronę. Po osiągnięciu punktów *b* i *d*, rozpoczyna się najpierw powrót pierwszego wózka, a następnie wózka drugiego. Graficzny model opisywanego procesu przedstawiony jest na rys. 1.



Rys. 1. Graficzny model analizowanego procesu
Fig. 1. Graphical model of the analysed process

Specyfikacja omawianego procesu sterowania zapisana została przy wykorzystaniu diagramu aktywności UML 2.0 (rys. 2). Ruch wózków w prawą stronę odbywa się współbieżnie, natomiast ich powrót odbywa się sekwencyjnie. Akcje może zostać uruchomiana wtedy, gdy spełniony będzie warunek opisany w nawiasie kwadratowym. Przykładowo, ruch wózków w prawą stronę rozpocznie się po naciśnięciu przycisku *m* przez użytkownika systemu. Wewnątrz akcji określone są wartości sygnałów wyjściowych służących do sterowania ruchem dwóch wózków (tab. 1).



Rys. 2. Diagram aktywności języka UML prezentujący omawiany proces
Fig. 2. UML activity diagram presenting the described process

Tab. 1. Znaczenie sygnałów wyjściowych
Tab. 1. Meaning of the output signals

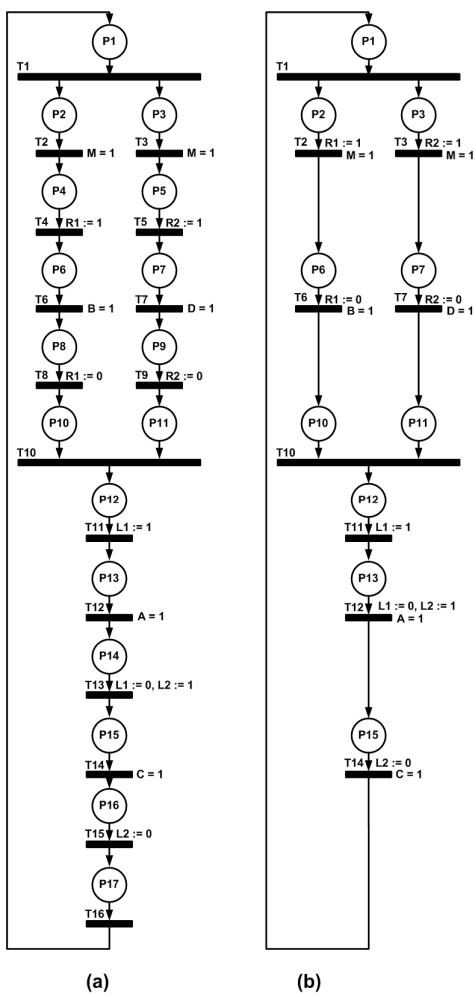
Sygnał	Znaczenie
R1	Ruch pierwszego wózka w prawą stronę.
L1	Ruch pierwszego wózka w lewą stronę.
R2	Ruch drugiego wózka w prawą stronę.
L2	Ruch drugiego wózka w lewą stronę.

Akcje diagramów aktywności w trakcie transformacji do sieci Petriego traktowane są jako tranzycje [11, 16]. Sieć Petriego po bezpośredniej transformacji przedstawiona jest na rys. 3 (a). Sieć zawiera 17 miejsc i 16 tranzycji. Sieć przedstawia krok po kroku zachowanie procesu sterowania bazując na jego interpretacji w postaci diagramu aktywności. Punkt początkowy i końcowy diagramu aktywności posiadają odpowiadające im miejsca *P1* i *P17* w sieci Petriego. Dodatkowo warunki wejściowe do akcji potraktowane zostały tak, jakby były zapisane jako bloki decyzyjne przed akcją. Stąd każdemu warunkowi wejściowemu z diagramu aktywności w sieci Petriego przypisana została tranzycja z odpowiednim warunkiem odpalenia. Zauważyć także należy dodatkowe miejsca synchronizacji (*P10, P11*) w wynikowej sieci Petriego. Są to elementy jawnie nieobecne w diagramie aktywności, jednakże składnia UML wymusza synchronizację w węźle *join*, stąd konieczność uwzględnienia tych elementów.

Analizując otrzymaną sieci Petriego można zauważyć, że część miejsc i tranzycji jest nadmiarowych. Proponowana metoda redukcji sieci Petriego polega na zastąpieniu tranzycji komplementarnych jedną tranzycją. Procedura ta powiązana jest z usuwaniem

zbytecznych miejsc sieci Petriego. Przykładowo, miejsce *P4* i następująca po nim tranzycja *T4* mogą zostać usunięte, a ich etykiety przeniesione do tranzycji *T2*. Tranzycja *T4* odzwierciedla wykonanie akcji przypisania sygnałowi *R1* wartości *1*. Zgodnie z diagramem aktywności (rys. 2), rozpatrywana akcja może zostać wykonana dopiero wtedy, gdy aktywny będzie sygnał *m*. Biorąc pod uwagę składnię sieci Petriego, akcja ta może być przypisana do tranzycji sprawdzającej stan sygnału *m*. Po redukcji nadmiarowych miejsc oraz tranzycji otrzymujemy sieć Petriego przedstawioną na rys. 3 (b) zawierającą 10 miejsc i 9 tranzycji.

Sieć Petriego po bezpośredniej transformacji opisuje zachowanie procesu sterowania krok po kroku. W rzeczywistości jednak, wiele operacji może być wykonywane jednocześnie, np. sprawdzanie warunku odpalenia tranzycji i wykonanie odpowiedni akcji w przypadku spełnienia badanego warunku. Taki sposób transformacji w jasny i przejrzysty sposób przedstawia sekwencję sygnałów wejściowych oraz wyjściowych. Jednakże, liczba miejsc i tranzycji staje się przez ten fakt dość duża, co może utrudniać analizę zachowania bardziej złożonych procesów sterowania. Konieczna staje się wtedy redukcja nadmiarowych miejsc i tranzycji wraz z przeniesieniem przypisanych do nich operacji.



Rys. 3. Sieć Petriego po transformacji z diagramu (a) oraz po redukcji nadmiarowych miejsc (b)
Fig. 3. Petri net diagram after transformation from activity diagram (a) and after reduction of redundant places (b)

6. Podsumowanie i wnioski

W artykule zarysowano sposób opisu zachowania procesów sterownia przy pomocy dwóch technik – diagramów aktywności języka UML oraz sieci Petriego. Obie techniki nadają się zarówno do opisu procesów sekwencyjnych jak także współbieżnych. Porównano obie techniki projektowania pod kątem wykorzystania ich w projektowaniu osadzonych systemów sterownia.

Przedstawiono również możliwość transformacji pomiędzy obiema technikami bazując na przykładzie sterowania dwoma wózkami. Po transformacji można zaobserwować, że bezpośrednie przekształcenie diagramów prowadzi do znacznej nadmiarowości w wynikowej sieci Petriego. Jednoznacznie dowodzi to, iż kolejnym niezbędnym krokiem w procesie transformacji jest eliminacja nadmiarowych miejsc i tranzycji w wynikowej sieci. Na podstawie omawianego przykładu można zaobserwować, że redukcja prostej sieci prowadzi do znacznego zmniejszenia ilości miejsc i tranzycji. Część elementów wprowadzonych podczas transformacji została użyta ze względu na poprawności procesu, jednakże w niektórych procesach te dodatkowe miejsca nie mają wpływu na opis zachowania i można je zredukować. Należy jednak zawsze pamiętać, żeby redukcja nie doprowadziła do zmiany zachowania procesu.

Celem zastosowania omawianej transformacji pomiędzy obiema technikami może być konieczność późniejszej weryfikacji formalnej specyfikacji [17] projektu pod kątem stawianych mu wymagań. Czynność ta motywowana jest bogatszą ofertą narzędzi weryfikujących dedykowanych dla sieci Petriego niż ma to miejsce dla diagramów aktywności języka UML.

7. Literatura

[1] M.A. Adamski, A. Karatkevich, M. Wegrzyn (ed.): Design of em-bedded control systems, Springer Science+Business Media, Inc., 2005.
[2] R. Miles, K. Hamilton: UML 2.0. Wprowadzenie, Helion 2007.
[3] http://www.omg.org
[4] P. Graessle, H. Baumann, P. Baumann: UML 2.0 w akcji. Przewodnik oparty na projektach, Helion 2006, s. 17 – 38.
[5] R. Eshuis, R. Wieringa: A Comparison of Petri Net and Activity Diagram Variants, Proc. of 2nd Int. Coll. on Petri Net Technologies for Modelling Communication Based Systems 2001, pp. 93 – 104.
[6] C. Yen-Liang, C. Sammy, C. Chyun-Chyi, C. Irene: Workflow Process Definition and Their Applications in e-Commerce, IEEE 2000, pp. 193 – 200.
[7] T. Schattkowsky: UML 2.0 – Overview and Perspectives in SoC Design, Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05).
[8] G. Łabiak, M. Adamski: Zastosowanie języka UML w modelowaniu sterownika dyskretnego, KNWS'06, PAK 6bis/2006, s. 50 – 52.
[9] R. David, H. Alla: Petri Nets and Grafcet, Prentice Hall, 1992.
[10] J.P. López-Grao, J. Merseguer, J. Campos: From UML Activity Diagrams To Stochastic Petri Nets: Application To Software Per-formance Engineering, WOSP 04 January, pp. 14-16.
[11] T.S. Staines: Intuitive Mapping of UML 2 Activity Diagrams into Fundamental Modeling Concept Petri Net Diagrams and Colored Petri Nets, 15th Annual IEEE International Conference and Work-shop on the Engineering of Computer Based Systems, 2008, pp. 191 200.
[12] M. Grobelny: A short comparison between UML Activity Diagrams and Petri Nets in hardware behavioural modelling, X Międzynarodowe Warsztaty Doktoranckie OWD'2008, Archiwum konferencji PTETIS, Vol. 25, październik 2008, pp. 433 436.
[13] S. Wrycza, B. Marcinkowski, K. Wyrzykowski: UML 2.0 w modelowaniu systemów informatycznych, Helion 2005.
[14] M. Fowler: UML w kropelce, Oficyna Wydawnicza LTP Sp. z o.o., Warszawa 2004.
[15] L. Gomes, J.P. Barros, A. Costa: Modeling Formalisms for Embedded System Design. Embedded Systems Handbook, Taylor & Francis Group, LLC, 2006.
[16] I. Tričković: Formalizing activity diagram of UML by Petri nets, Novi Sad J. Math, Vol. 30, No. 3, 2000, pp. 161 171.
[17] I. Grobelna: Formal verification of logic controller specification using NuSMV model checker, X Międzynarodowe Warsztaty Doktoranckie OWD'2008, Archiwum konferencji PTETIS, Vol. 25, październik 2008, pp. 459 464.
[18] M. Adamski, M. Chodań: Modelowanie układów sterowania dyskretnego z wykorzystaniem sieci SFC, Wydawnictwo Politechniki Zielonogórskiej, Zielona Góra 2000.

Artykuł recenzowany