Weryfikowanie specyfikacji wymagań sterownika logicznego za pomocą diagramów aktywności UML, logiki temporalnej LTL i środowiska NuSMV

Iwona Grobelna*, Michał Grobelny**

*Instytut Informatyki i Elektroniki, Uniwersytet Zielonogórski

**Katedra Mediów i Technologii Informacyjnych, Uniwersytet Zielonogórski

Streszczenie: W artykule przedstawiono ideę zastosowania diagramów aktywności UML do specyfikacji wymagań dotyczących zachowania sterownika logicznego. Lista wymagań podlegających weryfikacji zwykle definiowana jest bezpośrednio za pomocą formuł logiki temporalnej. Użycie przyjaznych dla użytkownika, powszechnie znanych i wykorzystywanych diagramów pozwala na prostsze i bardziej intuicyjne zapisanie wymagań. Diagramy są następnie formalnie przekształcane do formuł liniowej logiki temporalnej (LTL).

Słowa kluczowe: diagramy aktywności UML, specyfikacja, model logiczny, weryfikacja modelowa, logika temporalna

Język UML (ang. *Unifed Modelling Language*) [1–5] jest technologią do specyfikowania, wizualizowania i dokumentowania systemów informatycznych z uwzględnieniem wsparcia skalowalności, bezpieczeństwa i zapewnienia wysokiej jakości produktów finalnych. Behawioralny projekt systemu wbudowanego można sporządzić stosując wybrane diagramy UML, takie jak diagramy aktywności, maszyny stanów czy też diagramy sekwencji, jak również za pomocą interpretowanych sieci Petriego sterowania. Następnie specyfikacja wymagań może zostać poddana analizie, animacji, bądź formalnej weryfikacji. W artykule rozpatrywana jest weryfikacja modelowa specyfikacji [6, 7]. Sieć Petriego [8] zapisywana jest w postaci abstrakcyjnego regułowego modelu logicznego, dogodnego zarówno do formalnej weryfikacji modelowej, jak również do syntezy logicznej w strukturach FPGA [9, 10]. Istnieją także metody umożliwiające przekształcanie diagramów aktywności języka UML do sieci Petriego opisujących procesy sterowania [11]. Lista wymagań podlegających sprawdzeniu definiowana jest za pomocą formuł logiki temporalnej [12].

1. Wprowadzenie

W artykule zaproponowano zastosowanie diagramów aktywności języka UML do specyfikacji wymagań behawioralnych dotyczących działania projektowanego sterownika logicznego. Użycie przyjaznych dla użytkownika, powszechnie znanych i wykorzystywanych diagramów pozwala na prostsze i bardziej intuicyjne zapisanie wymagań. Notacja ta jest formalnie przekształcana do formuł liniowej logiki temporalnej.

W punkcie 2 przedstawiono zagadnienia związane ze specyfikacją sterownika logicznego oraz jej formalną weryfikacją. Omówiono też diagramy aktywności języka UML jako behawioralny opis działania sterownika logicznego, a także formalną weryfikację specyfikacji pod kątem stawianych jej wymagań z wykorzystaniem techniki weryfikacji modelowej. W punkcie 3 zaproponowano wykorzystanie diagramów aktywności do definicji globalnych wymagań, wraz z ich interpretacją z wykorzystaniem logiki temporalnej. Artykuł zakończono podsumowaniem oraz wnioskami.

2. Specyfikacja sterownika logicznego i jej weryfikacja

Etap specyfikacji urządzeń czy procesów w zastosowaniach przemysłowych jest bardzo ważnym, a nawet krytycznym elementem całego cyklu projektowania. Jest on jednym z pierwszych i kluczowych elementów procesu projektowego. Na tym etapie szczególnie istotne jest, aby specyfikacja spełniała wymagania i założenia określone przez projektanta, jak również klienta. Specyfikacja może zostać sformalizowana na kilka sposobów [13]. Przykładowymi technologiami specyfikacji behawioralnej sterowników logicznych są sieci Petriego, algorytmiczne maszyny stanów ASM, maszyny stanów czy diagramy aktywności języka UML.

2.1. Diagramy aktywności języka UML

Diagramy aktywności graficznie przedstawiają sekwencyjne i (lub) współbieżne przepływy sterowania oraz danych pomiędzy uporządkowanymi ciągami aktywności, akcji i obiektów [5]. Diagramy aktywności [1–5] charakteryzują się rozbudowaną składnią i funkcjonalnością umożliwiającą szerokie spektrum zastosowań. Stosując tę

technikę specyfikacji można formalizować behawioralne aspekty złożonych systemów informatycznych, procesów biznesowych czy precyzyjnie definiować algorytmy przetwarzania.

Za pomocą diagramów aktywności można sporządzić specyfikację projektowanego sterownika logicznego [11]. Możliwa jest także ich późniejsza transformacja do interpretowanych sieci Petriego sterowania [8], które z kolei mogą zostać poddane animacji, analizie czy weryfikacji.

W dalszej części artykułu zaproponowano wykorzystanie diagramów aktywności do definiowania wymagań, które będą następnie weryfikowane. Podstawowymi elementami diagramów, które znajdują tu zastosowanie są: węzeł początkowy i końcowy, akcja oraz przepływ. Właściwości wyspecyfikowane za pomocą diagramów aktywności są następnie zapisywane w postaci formuł liniowej logiki temporalnej.

2.2. Weryfikacja specyfikacji pod kątem stawianych jej wymagań

Technika weryfikacji modelowej (ang. *model checking*) [6, 7] umożliwia wykrywanie błędów w specyfikacji behawioralnej procesu sterowania. Właściwości systemu definiowane są użyciu postaci formuł logiki temporalnej i jej podstawowych operatorów. Zastosowanie metod weryfikacji modelowej pozwala na wczesne wykrycie niezauważalnych rozbieżności wynikających z niepełnego zrozumienia specyfikacji. Jest to jedna z możliwych metod formalnej weryfikacji specyfikacji sterownika logicznego pod kątem behawioralnym. Innym przykładem formalnej weryfikacji może być automatyczne dowodzenie twierdzeń (ang. *theorem proving*). Weryfikacja formalna jest alternatywą dla interpreterów czy maszyn wirtualnych, służących także do sprawdzania poprawności modeli behawioralnych.

Specyfikacja sterownika logicznego może zostać poddana weryfikacji modelowej, przykładowo z wykorzystaniem autorskiego regułowego modelu logicznego [9]. Regułowy model logiczny jest formatem pośrednim między pierwotną specyfikacją, modelem podlegającym formalnej weryfikacji oraz modelem poddawanym syntezie logicznej. Dzięki temu otrzymuje się syntezowalny model całkowicie spójny z modelem zweryfikowanym. Do przeprowadzenia procesu weryfikacji modelowej, oprócz opisu modelu, potrzebna jest także lista stawianych mu wymagań. Zwykle definiowane są one z użyciem logiki temporalnej – liniowej (LTL) lub rozgałęzionej (CTL).

W artykule zaproponowano zapis wybranych diagramów aktywności języka UML w postaci formuł liniowej logiki temporalnej (ang. *Linear Time Logic*). Logika ta opisuje zależności w systemie przedstawiając sekwencję stanów. Jej podstawowymi operatorami są: G (zawsze), F (czasami) i X (następnie). Logika LTL porównywana jest do testowania *black-box* [14], gdzie użytkownik otrzymuje „zamknięty w pudełku" system i może jedynie sprawdzić wartości sygnałów wejściowych oraz odpowiednie reakcje na nie sygnałów wyjściowych, bez znajomości procesów wewnętrznych.

Trafny dobór właściwości podlegających weryfikacji jest zadaniem trudnym i wymaga dużej uwagi ze strony projektanta lub inżyniera. Nie jest możliwe sprawdzenie wszystkich potencjalnie możliwych właściwości systemu, sprawdzane są tylko te cechy, które zostaną zdefiniowane. Osoba weryfikująca powinna więc dobrać takie właściwości, które są istotne z punktu widzenia działania systemu [15]. Jednak w przypadku złożonych systemów nigdy nie ma pewności, czy wyspecyfikowano wszystkie ważne wymagania, czy też pewne z nich pominięto. Minimalny zbiór podlegający weryfikacji opisuje właściwości krytyczne decydujące o zdrowiu czy nawet życiu osób, których los zależy od zaprojektowanego systemu (przemysł lotniczy, samochodowy, energetyczny, medyczny itp.).

Weryfikacja modelowa przeprowadzana jest z użyciem odpowiednich narzędzi komputerowych, przykładowo NuSMV [16].

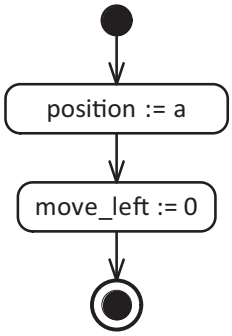
3. Definicja globalnych wymagań za pomocą diagramów aktywności

Właściwości podlegające weryfikacji [14] określają zarówno wymagania dotyczące bezpieczeństwa (sytuacje, które nie mogą mieć miejsca; ang. *something bad will never happen*), jak i wymagania dotyczące żywotności (sytuacje, które muszą się zdarzyć; ang. *something good will eventually happen*). Wymagania dotyczące bezpieczeństwa i żywotności są najczęściej specyfikowanymi wymaganiami podlegającymi weryfikacji.

3.1. Formuły z operatorem *X*

Stosując liniową logikę temporalną (LTL) można określić sekwencję sytuacji, które mają zawsze miejsce, na przykład następstwa stanów systemu. Diagramy aktywności języka UML w opisywanym przypadku mają częściowo charakter stanowy i w ten sposób nawiązują do maszyn stanów UML. Dzięki temu zdefiniowane wymagania bardziej odpowiadają specyfikacji behawioralnej systemu wysokiego poziomu. Wówczas możliwe jest zastosowanie diagramów aktywności zarówno do specyfikacji zachowania projektowanego sterownika logicznego [11], jak również do definicji wymagań. Weryfikacja modelowa może być przeprowadzona narzędzia środowisku NuSMV [16].

W artykule zaproponowano autorski schemat tworzenia globalnych diagramów aktywności, a więc diagramów, które definiują właściwości zawsze spełnione w projektowanym systemie. Dla operatora logiki temporalnej X (następny stan) diagram taki zawiera następujące po sobie akcje zawierające przypisania wartości do zmiennych. Należy tutaj podkreślić, że nazwy zmiennych odpowiadają tym zdefiniowanym wcześniej w behawioralnej specyfikacji. W szczególności diagramy określające wymagania mogą być wycinkiem diagramów specyfikacyjnych, większy sens mają jednak te, które przedstawiają pewien uproszczony wycinek systemu.



Rys. 1. Globalny diagram aktywności dla operatora (*X*)

Fig. 1. A global activity diagram for the operator (*X*)

Przykładowy globalny diagram aktywności dla operatora *X* przedstawiono na rys. 1. Określa on jedną właściwość systemu (jeden główny przebieg, jeden węzeł początkowy oraz końcowy) – zawsze, gdy zmienna *position* przyjmie wartość *a*, to następnie zmienna *move_left* przyjmie wartość 0. Zmienna *position* jest tutaj sygnałem wejściowym sterownika logicznego, a w rzeczywistym systemie jej wartość generowana jest przez zewnętrzny czujnik. Zmienna *move_left* jest sygnałem wyjściowym sterownika logicznego i decyduje o ruchu w lewą stronę.

Wymaganie zdefiniowane przy użyciu diagramu aktywności (rys. 1) można zapisać w liniowej logice temporalnej jak pokazano na Lst. 1 (format wymagań narzędzia weryfikacji modelowej NuSMV [16]). Znaki przypisania z rys. 1 zamieniane są tutaj na znaki równości. Wymaganie to może zostać zinterpretowane w rzeczywistym systemie tak, że zawsze gdy pojazd osiągnie pewien punkt (w tym przypadku punkt *a*), to następnie sygnał wyjściowy sterujący ruchem w lewą stronę przyjmie wartość *0*, a więc pojazd przerwie swoją jazdę i zatrzyma się.

```
LTLSPEC G (position = a -> X (move_left = 0))
```

Lst. 1. Wymaganie określone za pomocą logiki LTL dla operatora (*X*)

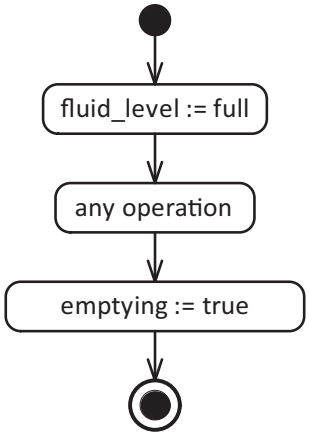
Lst. 1. Property expressed with LTL logic for the operator (*X*)

Prosta interpretacja takich diagramów umożliwia automatyczne zapisywanie z wykorzystaniem formuł logiki LTL.

3.2. Formuły z operatorem *F*

Analogicznie można przedstawić na diagramie aktywności formułę z operatorem *F*. Przykładowy globalny diagram aktywności dla operatora *F* przedstawiono na rys. 2.

Również określa on jedną właściwość systemu (jeden główny przebieg, jeden węzeł początkowy oraz końcowy) – zawsze, gdy zmienna *fluid_level* przyjmie wartość *full*, to w końcu zmienna *emptying* przyjmie wartość *true* (w międzyczasie mogą wystąpić dowolne operacje). Zmienna *fluid_level* jest tutaj sygnałem wejściowym sterownika logicznego, a w rzeczywistym systemie jej wartość generowana jest przez zewnętrzny czujnik określający poziom cieczy w zbiorniku. Zmienna *emptying* jest sygnałem wyjściowym sterownika logicznego i steruje opróżnianiem zbiornika (np. otwierając odpowiednie zawory). Występująca pomiędzy nimi akcja *any operation* oznacza dowolną lub dowolne (wiele) operacje. Jest to analogia do występującej w specyfikowaniu i implementacji systemów informatycznych notacji *nop* (ang. *no operation*) oznaczającej stan bezczynności. Akcja *any operation* może być akcją prostą lub złożoną i określa jedynie, że między dwiema konkretnie podanymi akcjami może wystąpić wiele innych.



Rys. 2. Globalny diagram aktywności dla operatora (*F*)

Fig. 2. A global activity diagram for the operator (*F*)

Wymaganie zdefiniowane przy użyciu diagramu aktywności na rys. 2 można zapisać w logice LTL jak pokazano na Lst. 2 (format wymagań narzędzia weryfikacji modelowej NuSMV [16]). Znaki przypisania z rys. 2 zamieniane są na znaki równości. Wymaganie to może zostać zinterpretowane w rzeczywistym systemie tak, że zawsze gdy zbiornik będzie pełny, to w końcu zostanie on opróżniony.

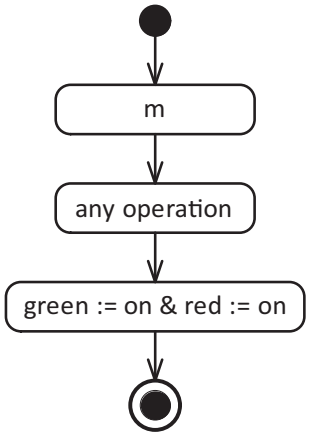
```
LTLSPEC G (fluid_level = full ->
                        F (emptying = true))
```

Lst. 2. Globalny diagram aktywności dla operatora (*F*)

Lst. 2. A global activity diagram for the operator (*F*)

3.3. Kontrprzykłady

Mając dany opis modelu (dostarczony początkowo chociażby w postaci diagramu aktywności języka UML [11]) oraz listę stawianych mu wymagań (opisanych za pomocą proponowanych globalnych diagramów aktywności) można przeprowadzić proces weryfikacji modelowej. Jeżeli któreś z wymagań nie może być spełnione, narzędzie weryfikujące (w tym przypadku NuSMV) wygeneruje odpowiednie kontrprzykłady, które pozwolą zlokalizować miejsce błędu. Przy weryfikacji modelowej wykorzystywane są mechanizmy zaproponowane w [9]. Opis modelu bazuje tutaj na interpretowanej sieci Petriego sterowania (utworzonej np. na podstawie diagramu aktywności UML), stąd też kontrprzykład operuje na dostępnej przestrzeni stanów globalnych sieci.



Rys. 3. Globalny diagram aktywności dla kontrprzykładu

Fig. 3. A global activity diagram for the counterexample

Kontrprzykład dla niespełnionego globalnego wymagania (przedstawionego na diagramie aktywności na rys. 3 i zapisanego w logice temporalnej jak na Lst. 3) zamieszczono na Lst. 4. Sygnał wejściowy do sterownika logicznego m decyduje o zmianie świateł – po jego naciśnięciu zapala się zielone światło, które następnie gaśnie i zapala się czerwone światło.

Sprawdzana jest właściwość, czy kiedykolwiek po naciśnięciu przycisku zapalone będzie zarówno światło zielone, jak i czerwone (odpowiedni diagram aktywności na rys. 3). Wymaganie to formalnie zapisano z wykorzystaniem logiki LTL (Lst. 3).

```
LTLSPEC G (m -> F (green = on & red = on));
```

Lst. 3. Wymaganie określone za pomocą logiki LTL dla kontrprzykładu

Lst. 3. Property expressed with LTL logic for the counterexample

```
-- specification  G (m ->  F (green = on &
red = on))  is false
-- as demonstrated by the following execu-
tion sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
-- Loop starts here
-> State: 1.1 <-
  green = off
  red = off
  m = FALSE
-> State: 1.2 <-
  m = TRUE
-> State: 1.3 <-
  green = on
  m = FALSE
-> State: 1.4 <-
  green = off
  red = on
-> State: 1.5 <-
  red = off
```

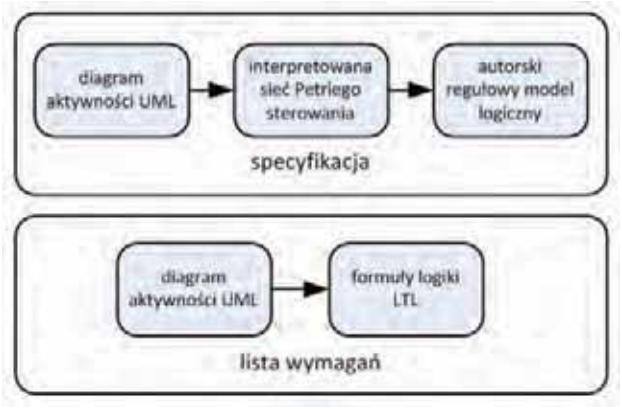
Lst. 4. Kontrprzykład

Lst. 4. The counterexample

Po przeprowadzonej weryfikacji modelowej użytkownik otrzymuje wygenerowany kontrprzykład (Lst. 4). Pokazuje on, że sytuacja, w której jednocześnie zapalone jest światło zielone i czerwone nie jest możliwa do osiągnięcia. W tym przypadku kontrprzykład potwierdza poprawne zaprojektowanie sterownika logicznego, zwykle jednak wymagania definiowane są w taki sposób, żeby były domyślnie spełnione – kontrprzykłady jawnie wskazują wtedy na potencjalne błędy w specyfikacji systemu bądź też w liście stawianych mu wymagań.

4. Podsumowanie i wnioski

W artykule zaproponowano wykorzystanie diagramów aktywności języka UML do specyfikacji wymagań projektowanego sterownika logicznego. Idea ta doskonale wpisuje się w schemat formalnej weryfikacji specyfikacji systemu wbudowanego (rys. 4) w oparciu o technikę weryfikacji modelowej [9, 10]. Sprawdzane jest wtedy, czy zdefiniowane wymagania są spełnione w opisie modelu, a weryfikacja przeprowadzana jest z użyciem narzędzia NuSMV. Diagramy aktywności wykorzystywane są tutaj zarówno do specyfikacji zachowania sterownika logicznego, jak również do definiowania wymagań. Specyfikacja zapisywana jest ostatecznie w postaci autorskiego regułowego



Rys. 4. Schemat proponowanej idei

Fig. 4. Schema of the proposed idea

modelu logicznego, który z jednej strony podlega weryfikacji, a z drugiej – syntezie logicznej. Wymagania zapisywane są formalnie za pomocą formuł logiki temporalnej (konkretnie logiki LTL) bazujących na poszczególnych diagramach aktywności.

Diagramy aktywności definiujące globalne wymagania stawiane projektowanemu sterownikowi logicznemu mogą być w szczególności pewnym fragmentem źródłowej specyfikacji. Mogą one jednak określać inne pożądane zachowania systemu. Należy bowiem mieć na uwadze fakt, że tylko wyspecyfikowane właściwości zostaną sprawdzone.

Użycie diagramów aktywności języka UML zarówno do specyfikacji sterownika logicznego, jak i do definicji wymagań ułatwia proces projektowania i weryfikacji sterownika logicznego. Inne formaty, dogodne do formalnej weryfikacji modelowej lub syntezy logicznej, otrzymywane są na podstawie danych diagramów według ściśle określonych reguł. Zespół analityków i projektantów może zaś pracować z przystępną, znaną i przejrzystą techniką modelowania, pośrednio wykorzystując zalety także innych formalnych technik.

Bibliografia

1. OMG Unified Modeling Language™ (OMG UML) Superstructure Version 2.4.1 – [www.omg.org/spec/UML/2.4.1/Superstructure/PDF]
2. Graessle P., Baumann H., Baumann P., *UML 2.x w akcji. Przewodnik oparty na projektach*, Wydawnictwo Helion, 2006.
3. Miles R., Hamilton K., *UML 2.0. Wprowadzenie*, Wydawnictwo Helion, 2007.
4. Pender T., *UML Bible*, Wiley Publishing, Inc., 2003.
5. Wrycza S., Marcinkowski B., Maślankowski J., *UML 2.0 w modelowaniu systemów informatycznych*, Wydawnictwo Helion, 2005.
6. Clarke E.M., Grumberg O., Peled D.A., *Model Checking*, The MIT Press, 1999.
7. Emerson E.A., *The Beginning of Model Checking: A Personal Perspective* [w:] Grumberg O., Veith H. (ed.), *25 Years of Model Checking*, vol. 5000 of Lecture Notes in Computer Science, Berlin, Heidelberg: Springer-Verlag, 2008, 27–45.
8. David R., Alla H., *Discrete, Continuous, and Hybrid Petri Nets*, 2nd ed., Springer Publishing Company, Incorporated, 2010.
9. Grobelna I., *Weryfikacja modelowa specyfikacji sterowników logicznych*, Oficyna Wydawnicza Uniwersytetu Zielonogórskiego, Zielona Góra 2012.
10. Grobelna I., *Formal verification of embedded logic controller specification with computer deduction in temporal logic*, „Przegląd Elektrotechniczny", nr 12a, 2011, 40–43.
11. Grobelny M., Grobelna I., *Diagramy aktywności UML w projektowaniu rekonfigurowalnych sterowników logicznych*, „Pomiary Automatyka Kontrola", vol. 58, nr 7, 2012, 596–598.
12. Ben-Ari M., *Logika matematyczna w informatyce. Klasyka informatyki*, Wydawnictwa Naukowo-Techniczne, 2005.
13. Gomes L., Barros J., Costa A., *Modeling Formalisms for Embedded System Design, Embedded Systems Handbook*, Taylor and Francis Group, LLC, 2006.
14. Kropf T., *Introduction to Formal Hardware Verification: Methods and Tools for Designing Correct Circuits and Systems*, 1st ed., Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1999.
15. Huth M., Ryan M., *Logic in Computer Science*: *Modelling and Reasoning about Systems*, New York, USA: Cambridge University Press, 2004.
16. *NuSMV: a new symbolic model checker*, NuSMV home page, [http://nusmv.fbk.eu/].

Verification of logic controller requirements specification by means of UML activity diagrams, LTL temporal logic and NuSMV tool

Abstract: The article introduces an idea to use UML activity diagrams [1–5] for specification of requirements regarding logic controller behavior. Requirements list to be verified [14] (using model checking technique [6, 7]) is usually directly defined using temporal logic formulas [12, 15]. Using user-friendly, commonly known and practiced diagrams allows to easier and more intuitively write down the requirements easier and more intuitively. Activity diagrams are then formally transformed into linear temporal logic (LTL) formulas. In this paper some sample UML activity diagrams which specify global properties are presented, together with their interpretation using LTL logic. To perform model checking process, model description (based i.e. on a control interpreted Petri net [8] or indirectly on an UML activity diagram [11]), and requirements list are needed. Afterwards it is checked, whether defined properties are satisfied in specified model description. If a requirement cannot be fulfilled, appropriate counterexample is generated allowing to localize error source. The article is structured as follows. Section 1 is an introduction. Background of a logic controller specification and its verification is presented in section 2. A novel approach to logic controller requirements definition using activity diagrams is shown in section 3. The paper ends with a short summary.

Keywords: UML activity diagrams, specification, logical model, model checking, temporal logic

Artykuł recenzowany, nadesłany 15.06.2013, przyjęty do druku 26.08.2013.

dr inż. Iwona Grobelna

Absolwentka Wydziału Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego oraz Fachhochschule Giessen-Friedberg (Niemcy). Obecnie zatrudniona na stanowisku adiunkta w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego.

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

mgr inż. Michał Grobelny*

Absolwent Wydziału Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego oraz Fachhochschule Giessen-Friedberg (Niemcy). Obecnie zatrudniony na stanowisku asystenta w Katedrze Mediów i Technologii Informacyjnych Uniwersytetu Zielonogórskiego.

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

*Autor 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.