Piotr BILSKI, Robert ŁUKASZEWSKI
KATEDRA INFORMATYKI, SZKOŁA GŁÓWNA GOSPODARSTWA WIEJSKIEGO
INSTYTUT RADIOELEKTRONIKI POLITECHNIKI WARSZAWSKIEJ

Czasowy model bloku przetwarzania danych w systemie pomiarowym

dr inż. Piotr BILSKI

W 2001 roku ukończył Wydział Elektroniki i Technik Informacyjnych Politechniki Warszawskiej, uzyskując tytuł mgr inż. Elektroniki. W 2006 roku uzyskał tytuł doktora nauk technicznych w dziedzinie elektroniki. Obecnie jest pracownikiem Katedry Informatyki na Wydziale Ekonomiczno-Rolniczym Szkoły Głównej Gospodarstwa Wiejskiego. Jego zainteresowania naukowe obejmują metody diagnostyczne, metody sztucznej inteligencji i uczenia się maszyn, a także projektowanie i analizę wirtualnych przyrządów pomiarowych.

e-mail: pbilski@mors.sggw.waw.pl

mgr inż. Robert ŁUKASZEWSKI

Od 1997 roku pracownik pracowni Komputerowej Techniki Pomiarowej, Instytutu Radioelektroniki Politechniki Warszawskiej. Prowadzone prace naukowo-badawcze oraz publikacje ściśle związane są z projektowaniem, modelowaniem i implementacją oprogramowania rozproszonych systemów pomiarowo-kontrolnych. Szczegółowy obszar zainteresowań obejmuje interfejsy pomiarowe, sieci komputerowe, projektowanie systemów, języki formalne.

e-mail: rlukaszewski@ire.pw.edu.pl

Streszczenie

Artykuł dotyczy modelowania komputerowych systemów pomiarowo-sterujących (KSPS) uwzględniającej czasowe aspekty przepływu informacji w KSPS. W artykule zamieszczono opis części większych prac dotyczących nowej metody opisu modelu KSPS z wykorzystaniem sieci Petri'ego i narzędzia CPN Tools. Szczegółowo przedstawiono model bloku przetwarzania KSPS. Opisano dokładnie funkcje opóźnień zaimplementowane w modelu oraz metodę ich wyznaczania. Dodatkowo zamieszczono badania opóźnień bloku przetwarzania rzeczywistego KSPS oraz symulacje weryfikujące zaprojektowany model.

Słowa kluczowe: systemy pomiarowe, sieci Petri'ego, model czasowy

Time Model of Data Processing Block in Measurement System

Abstract

The paper presents the method of the distributed measuring and control systems modelling, regarding the time aspects of the information flow inside the system. Results of the work including a new method of the measuring system model description using the Petri nets and CPN Tools are included. The data processing model inside the measuring system was presented in detail with delay functions implemented in the model. The paper is concluded with experimental results, comparing simulations with real measuring system behavior.

Keywords: modelling, Petri Nets, measurement systems

1. Wstęp

Zasady projektowania komputerowych systemów pomiarowo-sterujących (KSPS) obecnie mają charakter heurystyczny Sam proces projektowania KSPS odbywa się za pomocą prostych metod inżynierskich, w których nacisk kładzie się na dokładność pomiaru. Gdy projektant musi kontrolować czas trwania poszczególnych operacji pojawia się konieczność sprawdzenia możliwości synchronizacji wykonywanych zadań i zbadania możliwości utrzymania wymagań czasowych już na etapie projektu. Z tego powodu celowe jest tworzenie modelu projektowanego systemu, pozwalającego z dużą dokładnością określić cechy dynamiczne gotowego rozwiązania.

W [1] przedstawiono opracowaną metodykę projektowania KSPS korzystającą z modeli zbudowane z wykorzystaniem sieci Petri'ego. W toku badań opracowano czasowe modele: bloku akwizycji danych, interfejsu pomiarowego w standardzie IEC-625, protokołu TCP/IP, graficznego interfejsu użytkownika itp.

Niniejszy artykuł przedstawia wycinek prac dotyczący opracowanego ogólnego modelu bloku przetwarzania oraz jego weryfikacji.

1.1. Modelowanie KSPS

Dotychczas podjęto wiele prób modelowania systemów pomiarowych lub ich elementów składowych. W zastosowaniach inżynierskich modelowanie systemów pomiarowych często sprowadza się do wprowadzenia uproszczonych modeli przyrządów pomiarowych, złożonych z elementów skupionych. Nie uwzględnia się analizy czasowej, związanej z opóźnieniami wprowadzanymi przez poszczególne bloki. Istnieją jednak metody modelowania komunikacji systemów przemysłowych, gdzie można ściśle określić czasy przepływu komunikatów. Metody te opierają się na teorii szeregowania zadań [2], metodzie najgorszego przypadku czy też szeregowaniu GRMS [3]. W rozproszonych KSPS (RKSPS) metody te nie są wystarczające ze względu na niedeterministyczny charakter elementów RKSPS: systemu operacyjnego, czy sieci komputerowych.

Nie opracowano dotąd ogólnego sposobu opisu RKSPS. Tworzone są modele znanych elementów składowych, np.: czujników lub sieci przemysłowych. W systemach o niskim poziomie rozproszenia, m.in.: opartych na sieci dedykowanej, istnieją modele czasowe dla warstwy akwizycji i transportowej. W systemach o dużym stopniu rozproszenia, gdzie nieznane są dokładne opóźnienia wnoszone przez sieć, takiego modelu nie ma. Dlatego konieczne jest opracowanie modelu łączącego znane sposoby opisu systemów opartych na sieci przemysłowej z modelami sieci rozległych, analizowanymi w telekomunikacji.

1.2. Ograniczenia czasowe w KSPS

Źródłami opóźnień w KSPS mogą być wszystkie bloki toru przetwarzania danych pomiarowych oraz tor przesyłu sygnałów kontrolno-sterujących w KSPS (od bloku akwizycji danych do bloku prezentacji), co jest szczególnie uciążliwe w RKSPS. Przyczyny powstawania opóźnień w części sprzętowej związane są z bezwładnością elementów fizycznych oraz skończonym czasem wykonywania działań. Opóźnienia sprzętowe mogą być z reguły wyznaczone na podstawie danych katalogowych użytych elementów. Istotniejsze są opóźnienia o charakterze programowym i transmisyjnym.

Rozproszenie terytorialne KSPS powoduje wzrost czasu wymaganego na transmisję informacji. Zintegrowanie w jednym RSPS wielu zadań pomiarowych wiąże się z problemem

synchronizacji poszczególnych operacji. W systemach synchronicznych podstawowym problemem jest zapewnienie jednolitości czasu w różnych węzłach systemu w celu jednoczesnego wykonania pomiarów w wielu węzłach RKSPS.

W systemach asynchronicznych główną trudnością jest zapewnienie założonej kolejności wykonywanych zadań. Utrudnia to projektowanie w szczególności RKSPS wykorzystujących niedeterministyczne czasowo łącza komunikacyjne jak np. Internet. Czasy przesyłania wyników pomiarów oraz komunikatów kontrolnych ograniczają szybkość wykonywania pomiarów. Jest ona także uzależniona od przepustowości magistrali pomiarowej i przepustowości łącz telekomunikacyjnych. Istnienie sprzętowych, programowych oraz transmisyjnych opóźnień w systemie jest źródłem powstawania błędów pomiaru wielkości zmiennych w czasie [4].

1.3. Sieci Petri'ego

Sieci Petri'ego wykorzystuje się do specyfikowania i analizy systemów asynchronicznych i o równoległych obliczeniach. Opisują one przepływ sterowania i danych w systemach współbieżnych. Graficzną reprezentacją sieci jest dwudzielny graf złożony z dwu rodzajów węzłów zwanych miejscami (ang. places) i przejściami (ang. transitions). Węzły połączone są skierowanymi łukami. Miejsca (przedstawiane jako okręgi lub elipsy) służą do reprezentowania warunków, przejścia (prostokąty) reprezentują zdarzenia. Miejsca sieci zawierają znaczniki (ang. tokens), oznaczane kropkami.

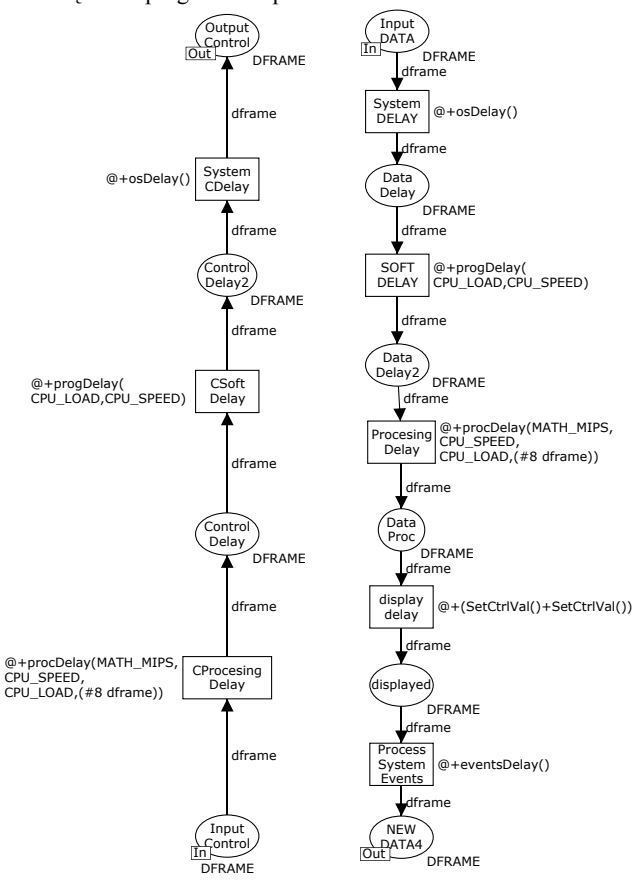
Dla modelowania SPS ważne są rozszerzenia sieci Petri'ego: sieci wysokiego poziomu (ang. High-Level Petri Nets), dzielące się na sieci otoczeniowe/relacyjne (ang. Environment Relationship Nets) [5] oraz kolorowane sieci Petri'ego (ang. Coloured Petri Nets - CPN) [6, 7]. Sieci wysokiego poziomu umożliwiają opisywanie systemów w sposób bardziej zwięzły oraz ułatwiają modelowanie i analizę systemów złożonych. W CPN każdy znacznik (kolor) wyposażony jest w skojarzoną z nim wartość (np. liczba całkowita). Do modelowania systemów złożonych, CPN pozwalają na rozdzielenie sieci na podsieci tworząc złożone hierarchiczne CPN. Czasowe rozszerzenia sieci Petri'ego zaproponowane zostały dla klasycznych sieci miejsc i przejść [8, 9] oraz dla sieci wysokiego poziomu [7, 10]. Czasowe CPN (TCPN) operują pojęciem globalnego zegara. Znaczniki przenoszą wartość czasową (pieczątkę czasową), która oznacza najwcześniejszą chwilę czasu, kiedy znacznik może być użyty. W opisywanym projekcie wykorzystano narzędzie programowe CPN Tools do modelowania i analizy KSPS.

2. Model bloku przetwarzania danych

Przetwarzanie danych w komputerowych systemach pomiarowych odbywa się po stronie serwera i klienta. W rozważanej klasie systemów przetwarzanie danych pomiarowych i sterujących odbywa się programowo – w środowisku uruchamianym na systemie operacyjnym powszechnego użytku. Zastosowanie tego typu systemów operacyjnych powoduje, iż opóźnienia wnoszone do systemu pomiarowego przez blok przetwarzania danych mają charakter niedeterministyczny i można je opisać jako rozkłady zmiennej losowej. Źródłem opóźnień jest działanie algorytmu obliczeniowego oraz systemu operacyjnego (procesów działających równolegle do procesu realizującego zadania przetwarzania danych).

2.1. Funkcjonalność modelu

Model procesu akwizycji danych (Rys. 1) skonstruowano w postaci jednorodnej czasowej kolorowanej sieci Petri'ego. Przepływ danych w modelu bloku przetwarzania odbywa się liniowo i dwukierunkowo. Żetony reprezentujące sygnały kontrolne wędrują od miejsca *Input Control* do miejsca *Output Control*. Przepływ danych pomiarowych odbywa się od miejsca *Input DATA* do miejsca *NEW DATA4*. Przemieszczanie żetonów obiema drogami odbywa się w sposób wzajemnie niezależny, co w rzeczywistym systemie oznacza niezależność przetwarzania danych sterujących i pomiarowych. Jest ona realizowana przez wielowątkowe programowe przetwarzanie zdarzeń.



Rys. 1. Model bloku przetwarzania danych.
Fig. 1. Structure of the data processing model.

2.2. Struktura danych modelu

Struktura danych sieci *MS CPU BLOCK* jest opisana kolorem *DFRAME* (Tab. 1) definiującym ramki danych pomiarowych i sterujących.

```
colset DFRAME  = product
        INT*INT*INT*INT*INT*ID*ID*INT*DATA timed;
colset DATA = string timed;
```

Tab. 1. Struktura koloru DFRAME
Tab. 1. Structure of the DFRAME color.

pole	1	2	3	4	5	6,7	8	9
typ	INT	INT	INT	INT	INT	ID	INT	DATA
Znaczenie	Identyfikator sesji	Identyfikator pomiaru	Ilość pomiarów	Okres wyzwalania	Pole rezerwowe	Pola rezerwowe	Długość pola danych	Dane

2.3. Mechanizmy czasowe modelu

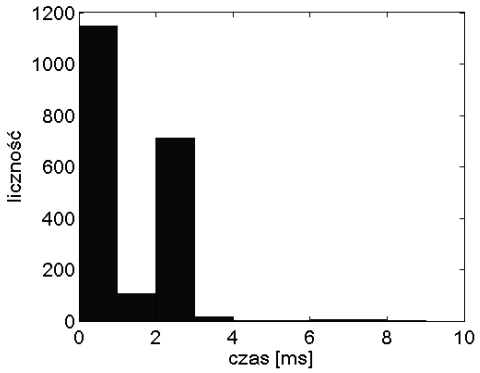
Zadaniem bloku przetwarzania danych w modelu jest symulacja opóźnień wprowadzanych przez zjawiska mające miejsce w tym bloku. Opóźnienia są realizowane poprzez wstrzymanie odpalenia przejść. Czas upływający od nadejścia

żetonu danych do odpalenia przejścia określają skojarzone z przejściami funkcje języka CPN ML. Czas określony przez te funkcje jest dodawany do czasu przenoszonego przez żeton danych. Po odpaleniu przejścia żetonowi danych przypisywana jest nowa wartość czasu.

Żeton danych pomiarowych w miejscu *Input DATA* modelu bloku przetwarzania danych jest opóźniany przez funkcję *osDelay* przypisaną do przejścia *System DELAY*. Symuluje ona opóźnienia wprowadzane przez system operacyjny. Wyznaczono ją na podstawie badań systemu operacyjnego Microsoft Windows XP. Wprowadza on losowe opóźnienie do wykonywanych algorytmów obliczeniowych spowodowane obsługą wewnętrznych zadań systemowych.

Po odpaleniu przejścia *System DELAY*, żeton danych przemieszczany jest do miejsca *Data Delay*, gdzie oczekuje na odpalenie przejścia *SOFT DELAY*. Opóźnienie odpalenia tego przejścia jest określone funkcją *progDelay*, modelującą opóźnienie wnoszone do bloku przetwarzania danych przez współbieżne procesy działające na serwerze pomiarowym. Opóźnienie to jest wprost proporcjonalne do obciążenia procesora w wyniku działania procesów współbieżnych oraz odwrotnie proporcjonalne do szybkości procesora realizującego zadania przetwarzania danych pomiarowych.

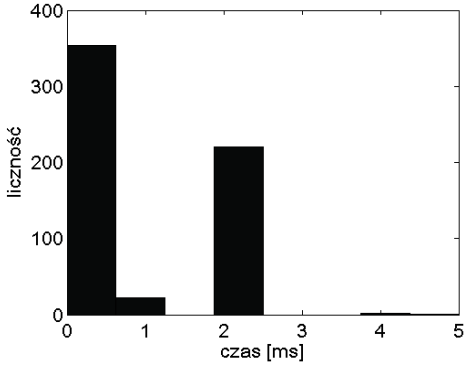
Po czasie określonym przez funkcję *progDelay* odpalane jest przejście *SOFT DELAY* i żeton danych przenoszony jest do miejsca *Data Delay2*, gdzie oczekuje na odpalenie przejścia *Processing Delay*. Wartość opóźnienia odpalenia tego przejścia jest określona przez funkcję *procDelay*, modelującą czas potrzebny na realizację algorytmu obliczeniowego. Wartość opóźnienia jest proporcjonalna do złożoności obliczeniowej algorytmu i aktualnego obciążenia procesora, a odwrotnie proporcjonalna do szybkości procesora i ilości danych do przetworzenia.



Rys. 2. Histogram rzeczywistych czasów wykonania funkcji *SetCtrlVal* (LabWindows/CVI, Celeron 1,2GHz@1,5GHz)
Fig. 2. Histogram of the *SetCtrlVal* function execution time (LabWindows/CVI, Celeron 1,2GHz@1,5GHz)

Odpalenie przejścia *Procesing Delay* powoduje przemieszczenie żetonu danych do miejsca *Data Proc*, gdzie oczekuje na odpalenie przejścia *display delay*. W opisywanym modelu serwera pomiarowego pominięto blok modelujący GUI serwera. Obsługa interfejsu użytkownika odbywa się w tym samym wątku programowym co operacje przetwarzania danych. Dlatego należy przenieść opóźnienia elementów GUI serwera do modelu bloku przetwarzania danych. Opóźnienie to jest modelowane przez funkcje skojarzone z przejściem *display delay*. W celu identyfikacji opóźnień wprowadzanych przez elementy graficzne przebadano działanie najczęściej stosowanych kontrolek. Stwierdzono, iż opóźnienia wnoszone przez ich modyfikacje nie są zależne od rodzaju kontrolki. Wielokrotne eksperymenty umożliwiły wykreślenie histogramów czasów wykonania funkcji *SetCtrlVal* dla różnych konfiguracji sprzętowych. Przykładowy histogram dla serwera pomiarowego z procesorem Celeron o nominalnej częstotliwości taktowania 1,2GHz po podwyższeniu jego częstotliwości do 1,5GHz zamieszczono na Rys. 2. Histogramy dla innych konfiguracji miały podobny charakter.

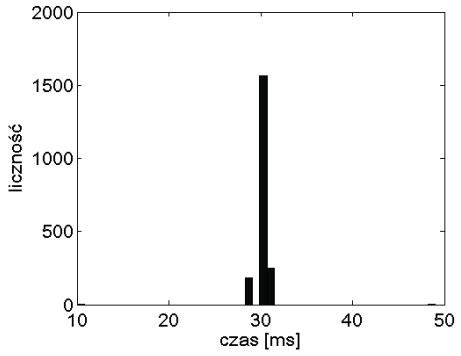
W celu weryfikacji modelu funkcji opóźnienia funkcji *SetCtrlVa*l przebadano otrzymany model symulacyjnie. Eksperymenty wykonano w środowisku CPN Tools [11]. W celu wyznaczenia modelowanego opóźnienia posłużono się mechanizmem monitorów dostępnym w tym środowisku. Pozwala on na zapisywanie czasów żetonów przybywających do określonych miejsc modelu. Opóźnienia związane z wybranym zjawiskiem można więc wyznaczyć na podstawie różnicy czasów zapisanych dla granicznych miejsc. Czasy związane z funkcjami graficznymi były zapisywane dla miejsc *data proc* i *displayed*. Różnice pomiędzy czasami przyjścia żetonów do tych miejsc symulują opóźnienie związane z zapisem danych do graficznych kontrolek w oprogramowaniu systemu. Otrzymane wyniki symulacji (Rys. 3) potwierdziły poprawność wyznaczonej funkcji modelującej czas opóźnienia modyfikacji wartości elementów graficznych. Wartość procentowa z prawej strony rysunku (wraz z krzywą w tle) oznacza prawdopodobieństwo, że wystąpi opóźnienie o czasie krótszym, niż wskazywany przez oś x.



Rys. 3. Przykładowy histogram opóźnień funkcji *SetCtrlVal* - CPU_SPEED = 3738 MIPS (Celeron 1,2GHz@1,5GHz).
Fig. 3. Histogram of the *SetCtrlVal* function execution time - CPU_SPEED = 3738 MIPS (Celeron 1,2GHz@1,5GHz).

Ostatnim opóźnieniem modelowanym w bloku przetwarzania danych serwera pomiarowego jest czas związany z obsługą zdarzeń systemowych. Przy sekwencyjnym wykonywaniu wielu pomiarów są one wykonywane w pętli, w której wyniki pomiarów są odbierane, przetwarzane, wyświetlane czy też wysyłane do innego węzła systemu pomiarowego. Czynności te wykonywane w pętli pomiarowej blokują obsługę zdarzeń systemowych, np. uniemożliwiają reakcje użytkownika. W celu umożliwienia obsługi tych zdarzeń w środowisku LabWindows/CVI stosuje się czasochłonną funkcję *ProcessSystemEvents*. W ramach badań wyznaczono histogramy opóźnień (Rys. 4) wprowadzanych przez tę funkcję dla różnych konfiguracji sprzętowych. Stosując identyczną metodę jak przy wyznaczaniu funkcji opóźnienia dla elementów graficznych wyznaczono funkcję *eventsDelay* modelującą czas wykonania funkcji *ProcessSystemEvents*.

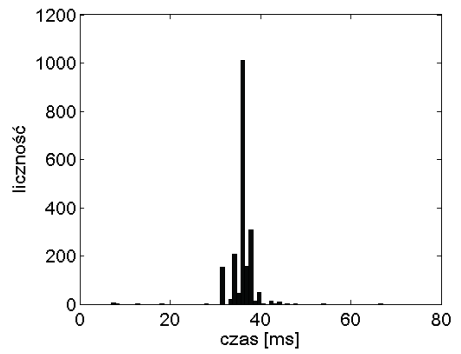
Przepływ żetonów reprezentujących sygnały sterujące w modelu bloku przetwarzania serwera pomiarowego odbywa się na podobnych zasadach jak dla żetonów danych pomiarowych. Różnice to nieuwzględnianie opóźnienia związanego z funkcjami *ProcessSystemEvents* oraz elementami graficznymi. W rzeczywistym systemie przy przetwarzaniu sygnałów sterujących nie ma potrzeby wywoływania funkcji obsługi zdarzeń systemowych, gdyż przetwarzanie to nie odbywa się w pętli programowej a jedynie w momencie zgłoszenia nadejścia danych. Przetwarzanie sygnałów sterujących zazwyczaj nie wymaga też obsługi elementów graficznych. Uwzględnienie tego opóźnienia w modelu wymaga jedynie utworzenia odpowiedniego przejścia i wykorzystania funkcji *SetCtrlVal* do określenia opóźnienia jego odpalenia na wzór przejścia *display delay*.



Rys. 4 Przykładowy histogram rzeczywistych czasów wykonania funkcji *ProcessSystemEvents* (LabWindows/CVI, Duron 1,2GHz)
Fig. 4 Histogram of the *ProcessSystemEvents* function execution time (LabWindows/CVI, Duron 1,2GHz)

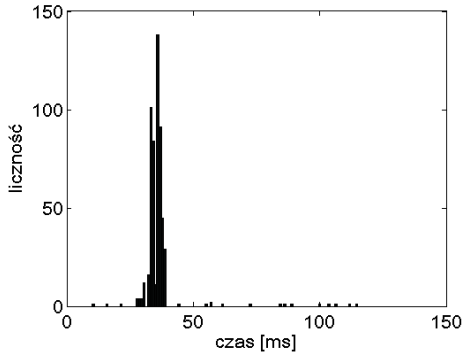
3. Weryfikacja modelu

W celu weryfikacji modelu zaprojektowano system pomiarowy w środowisku LabWindows/CVI. Opracowane oprogramowanie pozwala wyznaczać czasy wykonania algorytmów przetwarzania danych wraz z całą pętlą pomiarową. Wyznaczono histogramy czasów przetwarzania i wyświetlania danych. Badania przeprowadzono dla różnych platform sprzętowo-programowych w systemie nieobciążonym oraz w obecności innych procesów obciążających system. Otrzymane histogramy porównano z wynikami symulacji zaprojektowanego modelu. Symulacje wykonano w środowisku CPN Tools z wykorzystaniem wcześniej opisanego mechanizmu monitorowania czasów żetonów w określonych miejscach modelu.



Rys. 5. Histogram rzeczywistych czasów wykonania pętli programowej serwera pomiarowego (LabWindows/CVI)
Fig. 5. Histogram of the server's software loop execution (LabWindows/CVI)

Porównanie otrzymanych wyników wykazało podobne właściwości czasowe modelu i rzeczywistego systemu pomiarowego. Przykładowo, wyznaczono histogramy (Rys. 5) czasów wykonania pętli programowej zawierającej dwie funkcje zapisu do elementów graficznych oraz funkcję *ProcessSystemEvents* na serwerze pomiarowym z procesorem AMD Duron 1,2GHz i systemem operacyjnym Microsoft Windows XP Pro. Procesor obciążono w 20% procesami równoległymi. Przeprowadzono badania symulacyjne po sparametryzowaniu modelu. Parametrami modelu były: szybkość procesora wyrażona w MIPS oraz jego obciążenie wyrażone w procentach. Wyznaczono histogram opóźnień (Rys. 6) wnoszonych przez cały model na podstawie mechanizmu monitorów. Otrzymane wyniki symulacji wykazują, iż opóźnienia wnoszone przez badaną pętlę programową mają wartości w przedziale od 33ms do 43ms. Jako kryterium oceny poprawności modelu można przyjąć różnice wartości średnich uzyskanych na drodze symulacji i badań systemu rzeczywistego. Dla kilku różnych konfiguracji sprzętowych wartości te nie różniły się więcej niż 1%. Wyniki te potwierdzają prawidłowość modelu.



Rys. 6. Histogram modelowanych czasów wykonania pętli programowej serwera pomiarowego (CPN Tools)
Fig. 6. Histogram of the modelled server's software loop execution (CPN Tools)

4. Podsumowanie

Rozszerzenia sieci Petri'ego pozwalają na modelowanie złożonych systemów z czasem w sposób zwięzły, bez zatracania szczegółowości opisu. Dotychczasowe badania wykazują przydatność stosowania sieci Petri'ego do modelowania KSPS. Zamieszczone wyniki symulacji potwierdzają poprawność skonstruowanego modelu bloku przetwarzania danych. Pozwala on na wyznaczenie opóźnień wnoszonych przez poszczególne funkcje oraz cały blok przetwarzania danych. Wraz z modelami pozostałych bloków KSPS zaproponowany model umożliwia wyznaczenie ogólnych parametrów czasowych KSPS na etapie ich projektowania. Posługując się wynikami symulacji można np. wyznaczyć graniczne parametry czasowe akwizycji danych czy też stratność danych w systemie spowodowaną przekroczeniem wymagań czasowych związanych z transmisją informacji w systemie. Modele pozostałych bloków systemu pomiarowego będą przedmiotem kolejnych publikacji.

5. Literatura

[1] R. Łukaszewski, W. Winiecki, "Petri Nets in Measuring Systems Design", Proc. of the 23rd IEEE IMTC, Sorrento, Italy, 24-27 April, 2006, pp. 1564-1569.
[2] E. Michta, Modelowanie komunikacyjne sieciowego systemu pomiarowo – sterującego; WPZ, Zielona Góra, 2000
[3] A. Markowski, Wyznaczanie opóźnień transmisji danych w sieciowych systemach pomiarowo-sterujących, PAR 7-8/2004, s. 95 – 99
[4] J. Jakubiec: Błędy powodowane opóźnieniami w systemie pomiarowo-sterującym, PAR 7 8/2004, s. 71 - 74
[5] A. M. K. Cheng: Real-Time Systems, Scheduling, Analysis, and Verification, John Wiley & Sons 2002.
[6] T. Szmuc: Modele i metody inżynierii oprogramowania systemów czasu rzeczywistego, Uczelniane Wydawnictwa Naukowo Dydaktyczne AGH, Kraków, 2001.
[7] K. Jensen: Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use, Vol. 1-3, Springer, 1996.
[8] H. M. Hanisch: Analysis of Place/Transition Nets with Timed Arcs and its Application to Batch process Control, LNCS Vol. 691, Springer-Verlag, 1993, 282-299.
[9] E. Y. T. Juan, et. al.: Reduction Methods for Real-Time Systems Using Delay Time Petri Nets, IEEE Transactions on Software Engineering, Vol. 27, No. 5, May 2001, 422-448.
[10] K. Sacha: Projektowanie oprogramowania systemów wbudowanych, Politechnika Warszawska, Prace Naukowe Elektronika z. 115, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa, 1996.
[11] http://www.daimi.au.dk/CPNTools/, CPN May 2005.

Artykuł recenzowany