Przetwarzanie sygnałów w pakiecie PExSim: zasady, techniki, możliwości

Michał Syfert, Paweł Wnuk
Instytut Automatyki i Robotyki Politechniki Warszawskiej

W Instytucie Automatyki i Robotyki Politechniki Warszawskiej rozwijana jest platforma programowa systemu modelowania, zaawansowanego sterowania, monitorowania oraz diagnostyki procesów przemysłowych DiaSter. Jednym z podstawowych modułów systemu jest pakiet przetwarzania zmiennych systemowych PExSim. W pakiecie tym wykorzystano ciekawe rozwiązania z zakresu bieżącego przetwarzania sygnałów. W artykule przedstawiono zastosowane w tym pakiecie zasady przetwarzania sygnałów, techniki oraz możliwości, jakimi dysponuje użytkownik pakietu.

Słowa kluczowe: system wspomagania decyzji, przetwarzanie sygnałów, modelowanie

Na rynku dostępnych jest wiele środowisk umożliwiających budowę symulatorów. Są to często rozwiązania bardzo zaawansowane, dające bardzo duże możliwości, ale jednocześnie mało dostosowane do realizacji bieżących obliczeń związanych np. z monitorowaniem i diagnozowaniem procesu przemysłowego. Zwykle są to rozwiązania bardzo drogie.

W zakresie zaawansowanego przetwarzania zmiennych sytuacja jest jeszcze bardziej skomplikowana. W miarę swobodnie konfigurowalne przetwarzanie można zwykle realizować w systemach klasy SCADA czy DCS, zakładając, że mamy do takich systemów dostęp. Narzędzia dostępne w tego typu systemach mają spore ograniczenia, nie są wystarczająco elastyczne, konfigurowalne oraz łatwo rozszerzalne.

W Instytucie Automatyki i Robotyki Politechniki Warszawskiej od wielu lat są opracowywane i rozwijane systemy zaawansowanego modelowania, symulacji oraz monitorowania i diagnostyki. W ostatnich latach powstał nowatorski system DiaSter, którego jądrem jest specjalnie opracowana platforma programowa. Z kolei jednym z ważniejszych elementów platformy jest moduł przetwarzania zmiennych systemowych w postaci swobodnie konfigurowalnych torów przetwarzania nazwany PExSim.

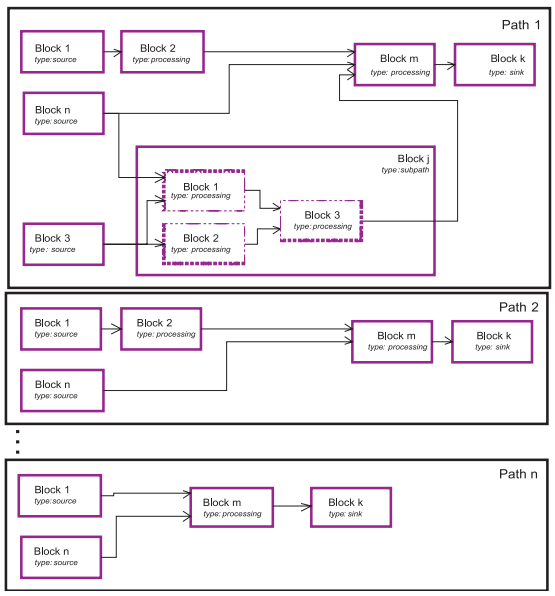
Moduł PExSim projektowany był głównie z myślą o wykonywaniu bieżących obliczeń związanych z realizacją zaawansowanych algorytmów modelowania, monitorowania oraz diagnostyki procesów przemysłowych. Moduł ten nie jest właściwie środowiskiem symulacji, takim jak np. MATLAB-Simulink, jakkolwiek z czasem dodane zostały do niego pewne elementy pozwalające na realizację nawet bardziej złożonych symulatorów.

Podczas projektowania modułu brano pod uwagę przede wszystkim możliwość jego łatwej rozbudowy przy zachowaniu zdolności do możliwie szybkiego wykonywania obliczeń, zadbano także o prostotę wprowadzania kolejnych rozszerzeń. Wymagania te zostały spełnione przez zastosowanie takich mechanizmów, jak: tworzenie bibliotek funkcyjnych w postaci wtyczek zrealizowanych jako dynamicznie ładowane biblioteki DLL, możliwości tworzenia własnych typów przetwarzanych sygnałów, elastyczność w zakresie sterowania wykonywanymi obliczeniami i wiele innych.

W artykule przedstawiono główne zasady działania oraz techniki zaimplementowane w module PExSim.

Struktury przetwarzania sygnałów

Algorytmy przetwarzania realizowane w pakiecie PExSim konstruowane są w postaci schematów blokowych, tzw. **ścieżek przetwarzania**. Konfiguracja modułu może zawierać wiele ścieżek przetwarzania. Ścieżki te mogą być niezależne lub zależne (gdy występuje między nimi wymiana danych). Ich liczba jest teoretycznie nieograniczona, ale należy liczyć się ze spadkiem wydajności obliczeniowej przy stosowaniu większej liczby ścieżek przetwarzania oraz złożonych algorytmów obliczeniowych. W takim przypadku system DiaSter umożliwia rozproszenie obliczeń przez konfigurację kilku współpracujących ze sobą, ale wykonywanych niezależnie, egzemplarzy modułu PExSim, które mogą być uruchomione na niezależnych komputerach. Wymiana danych między tak skonfigurowanymi modułami wykorzystuje ogólne mechanizmy komunikacji dostępne w systemie DiaSter. Tego typu współpraca nie będzie dokładniej omawiana w niniejszym artykule, gdyż jest on skoncentrowany na prezentacji wewnętrznych rozwiązań i mechanizmów modułu PExSim.

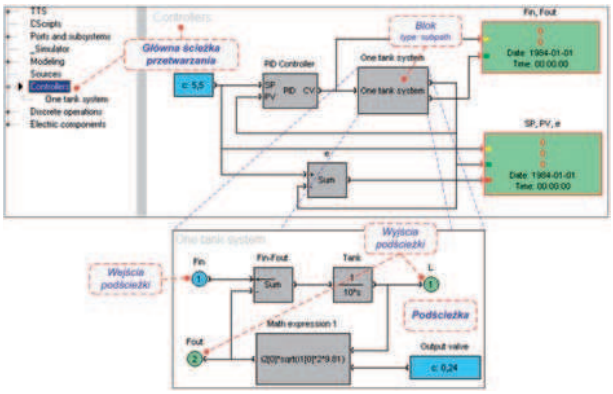


Rys. 1. Ogólna struktura algorytmów przetwarzania sygnałów w module PExSim

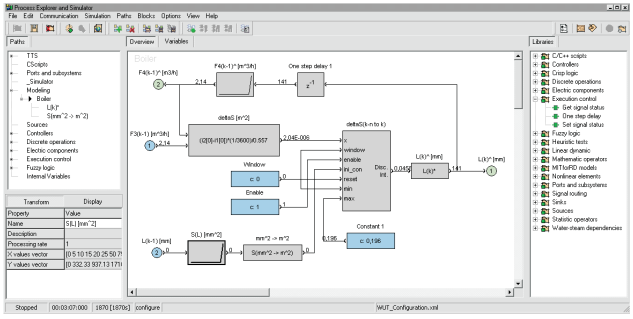
Fig. 1. General structure of the signal processing algorithms in the PExSim module

Każda ścieżka składa się z szeregu **bloków funkcyjnych** realizujących różnego rodzaju działania na przetwarzanych sygnałach. Przez **połączenie wejść** i **wyjść** bloków realizowany jest odpowiedni przepływ sygnałów. Przyjęty mechanizm tworzenia połączeń jest rozwiązaniem standardowym: każde wyjście może być podłączone do kilku wejść, natomiast do jednego wejścia może być podłączony tylko jeden sygnał wyjściowy, co związane jest z kierunkiem przepływu sygnałów. Ze względu na możliwość przetwarzania sygnałów różnego typu (w tym typów użytkownika) podczas tworzenia połączenia wyjście-wejście sprawdzana jest zgodność sygnałów. Dodatkowo bloki funkcyjne są tak projektowane, aby umożliwiać podłączenie różnych typów sygnałów wejściowych.

W ścieżce mogą być także osadzane **podsystemy** w postaci specjalnych bloków funkcyjnych, komunikujących się z innymi blokami za pomocą swoich wejść i wyjść. W ten sposób możliwa jest organizacja algorytmu przetwarzania w strukturze hierarchicznej. Z punktu widzenia logiki przetwarzania sygnałów podsystemy są elementami przeźroczystymi, pełnią wyłącznie funkcję porządkującą.



Rys. 2. Przykładowy widok ścieżki z osadzoną podścieżką (podsystemem)
Fig. 2. Exemplary view of the path with embedded sub-path (subsystem)



Rys. 3. Interfejs graficzny modułu przetwarzania PExSim
Fig. 3. Graphical user interface of the PExSim processing module

Dodatkową cechą związaną z podsystemami jest możliwość tworzenia **wzorcowych podsystemów użytkownika**. Każdy z podsystemów może zostać dodany do jednej z **podsystemowych bibliotek użytkowników** i wykorzystany w innej konfiguracji modułu PExSim.

Ogólny mechanizm przyjętej organizacji i projektowania algorytmów przetwarzania jest rozwiązaniem powszechnie stosowanym w systemach symulacyjnych. Różnica prezentowanego systemu w stosunku do innych rozwiązań jest efektem konkretnych szczegółów implementacji, które zostaną omówione w dalszej części.

Projektowanie oraz testowanie algorytmów przetwarzania realizowane jest przy wykorzystaniu **interfejsu graficznego**, w którym ścieżki wyświetlane są w postaci schematu blokowego. Jest to pierwszy tryb pracy modułu **PExSim,** dostępny dla systemu operacyjnego Windows. Podczas realizacji bieżących obliczeń moduł może być także uruchomiony w **trybie cichym**, bez interfejsu graficznego. W tym trybie moduł przygotowany jest do uruchomienia na różnych platformach systemowych.

Typy przetwarzanych sygnałów

Współczesne, komercyjne systemy automatyki stosują głównie proste typy danych – informacja niezbędna do bieżącego sterowania procesem według podstawowych algorytmów (np. PID, sterowanie logiczne) może zostać zakodowana tylko jako sygnały logiczne, całkowitoliczbowe czy też zmiennoprzecinkowe. Przez długi czas stosowanie tylko tych trzech typów danych pokrywało zapotrzebowanie implementowanych algorytmów. Niemniej jednak, wraz z wprowadzaniem nowoczesnych technik nadzorowania/sterowania oraz rozwojem informatyki, można stwierdzić, że takie ograniczenie w niektórych aplikacjach bywa uciążliwe. Na przykład – przesyłanie wartości pomiarowej wraz z niepewnością jej wyznaczenia wymaga utworzenia dwóch klasycznych sygnałów, mimo że jest to fizycznie jedna wartość. Co więcej, niektórych danych nie można przesłać w ten sposób. Przykładem mogą być wartości rozmyte. Z tych powodów w systemie DiaSter wprowadzono zmodyfikowane typy wartości. Ogólnie zakłada się, że każda wartość przetwarzana przez system charakteryzuje się kilkoma cechami:

- Zawsze z wartością powiązany jest status. Status jest nośnikiem informacji dodatkowej, powiązanej z wartością, taką jak błąd, który wystąpił przy jej wyznaczaniu, brak rzeczywistego pomiaru, przekroczenie wartości dopuszczalnych itp. Status jest nadawany danej wartości przez jej generator oraz może być modyfikowany przez każdy element przetwarzający tę wartość.
- Drugim elementem występującym zawsze z wartością jest stempel czasowy. W systemie zakłada się, że stempel czasowy jest nadawany przez generator wartości. Rozdzielczość stempla czasowego w systemie DiaSter wynosi 10E-6 s, co pozwala przetwarzać sygnały drganiowe o wysokiej częstotliwości.
- Wartości w systemie podlegają serializacji, co z punktu widzenia informatyki oznacza, że każdy typ wartości może być w dowolnym momencie zapisany jako ciąg bajtów, następnie przesłany przez sieć lub zapisany na nośniku nieulotnym, a potem odtworzony do dalszego wykorzystania.

Można zauważyć, że wśród cech podstawowych wartości w systemie nie ma definicji typu danych przesyłanych. Oznacza to, że system przetwarza wszystkie typy wartości w ten sam sposób, nie uwzględniając (na niskim poziomie) ich cech charakterystycznych. To, czy wartość może być przesłana między modułami systemu, przetwarzana wewnątrz modułu, archiwizowana czy też odtwarzana z archiwum, nie zależy od jej typu. W praktyce takie podejście pozwala na utworzenie elastycznego systemu, pozbawionego wad typowych systemów automatyki.

Techniczna realizacja przetwarzania dowolnych typów wartości przez system DiaSter jest oparta na koncepcji abstrakcyjnego interfejsu wartości nazwanego ICValue. Interfejs wraz z jego bazową implementacją zawiera mechanizmy zapewnia-

jące obsługę stempli czasowych, statusu oraz serializacji/deserializacji wartości.

Wszystkie wartości przetwarzane przez system są typami pochodnymi względem ICValue. Do typów dostarczanych wraz z platformą systemu zalicza się:

- wartości logiczne
- wartości całkowitoliczbowe
- wartości zmiennoprzecinkowe
- znaki
- napisy
- wektory składające się z elementów dowolnego typu.

Powyższa lista nie jest zamknięta – użytkownik może dodawać do systemu własne typy zmiennych przez wyprowadzenie typu pochodnego względem ICValue i zarejestrowanie go w centralnej bazie danych konfiguracyjnych. W ten sposób można implementować zarówno mało skomplikowane typy danych (jak np. liczby 12-bitowe odpowiadające standardowej rozdzielczości przetworników AC stosowanych w przemyśle), jak i te bardziej wyrafinowane.

Przykładem zaawansowanego typu danych mogą być opracowane w Instytucie Automatyki i Robotyki zmienne rozmyte (*fuzzy values*). Zmienne tego typu charakteryzują się ciekawymi właściwościami. Każda wartość rozmyta jest nierozerwalnie związana z odpowiadającą jej definicją sposobu rozmywania. Jako wartość rozmywaną przyjmuje się wartość funkcji przynależności do jednego lub więcej przedziałów rozmywania. W przypadku takiej definicji do pełnego odtworzenia i np. wizualizacji zmiennej potrzebny jest dostęp zarówno do wartości funkcji przynależności, jak też do definicji jej przedziałów. Przesyłanie definicji przedziałów wraz z każdą wartością jest wyjątkowo nieefektywne, tym bardziej że konfiguracja sposobu rozmywania zmienia się stosunkowo rzadko. W omawianym systemie problem efektywnego przesyłania zmiennych jest rozwiązany dzięki możliwości elastycznej definicji własnych typów. Wprowadzono dwa typy zmiennych rozmywanych – jeden zawierający same wartości funkcji przynależności, efektywny i oszczędzający pasmo transmisyjne. W przypadku zmiany konfiguracji zmiennej rozmywanej typ rozmyty jest automatycznie rozszerzany o nową konfigurację zmiennej i definicję sposobu rozmywania.

Elastyczne podejście opisane wyżej pozwala na uniknięcie większości problemów pojawiających się w systemach automatyki poprzedniej generacji przy implementacji przetwarzania nowych typów informacji.

Rodzaje i właściwości bloków funkcyjnych

Ze względu na stosowany mechanizm przetwarzania, bloki funkcyjne można podzielić na kilka podstawowych typów:

- **Źródła** – elementy pełniące rolę źródeł sygnałów odpowiedzialne za wyzwalanie obliczeń, zwykle bez wejść. Konkretne wartości mogą pochodzić z: generatorów przebiegów czasowych, wewnętrznych buforów danych, źródeł zewnętrznych (pliki, serwer komunikacyjny systemu DiaSter, bazy danych) itp. Źródłami mogą być także zwykłe bloki przetwarzania, które aktualny stan wyjść wyznaczają na podstawie przeszłych wartości sygnałów wejściowych. W takim przypadku blok jest wyzwalany, nawet jeśli nie ma ustalonych sygnałów wejściowych (mechanizm dokładniej omówiony w kolejnym punkcie).
- **Elementy przetwarzające** – podstawowe bloki funkcyjne przetwarzające zbiór sygnałów wejściowych na zbiór sygnałów wyjściowych według zadanego algorytmu.
- **Odbiorniki** – bloki funkcyjne bez sygnałów wyjściowych.

Odbiorniki można podzielić na dwie podstawowe grupy: elementy związane z wymianą danych (zapis do plików, wysyłanie poprzez serwer komunikacyjny systemu DiaSter, zapis do bazy danych itp.) oraz elementy wykorzystywane do analizy działania algorytmów przetwarzania (wyświetlacze numeryczne, wykresy czasowe i X-Y itp.). Druga grupa wykorzystywana jest jedynie w trybie pracy z interfejsem użytkownika. W trybie pracy cichej przetwarzanie w blokach drugiej grupy w ogólne nie jest realizowane.

Oddzielną grupę stanowią **bloki podsystemów** wykorzystywane do konstrukcji hierarchicznej struktury ścieżek przetwarzania.

Każdy z bloków funkcyjnych ma szereg parametrów konfiguracyjnych warunkujących sposób działania bloku. Parametry te podzielone są na dwie grupy:

- parametry związane z właściwościami algorytmu przetwarzania (Transform), np. liczba wejść i wyjść, współczynnik wzmocnienia, stała czasowa itp.
- parametry związane ze sposobem prezentacji bloku funkcyjnego w graficznym interfejsie użytkownika (Display), np. szerokość i wysokość symbolu bloku, kolor, położenie na ekranie itp. Parametry te nie mają żadnego wpływu na sposób przetwarzania sygnałów realizowanego przez dany blok.

Inny podział parametrów bloku wyróżnia właściwości **modyfikowalne** oraz **niemodyfikowalne** przez użytkownika.

Powiązane tematycznie bloki funkcyjne łączone są w **biblioteki funkcyjne**. Wszystkie dostępne biblioteki modułu przetwarzania PExSim zrealizowane są w postaci dynamicznie ładowanych bibliotek DLL obsługiwanych przez specjalnie przygotowany **mechanizm wtyczek**. Rozwiązanie takie pozwala na łatwe rozszerzanie możliwości modułu przez tworzenie bloków funkcyjnych użytkowników. Zastosowany mechanizm wtyczek praktycznie nie powoduje powstania żadnych narzutów obliczeniowych.

Razem z modułem PExSim dostarczany jest zestaw podstawowych bibliotek, zawierający m.in.: bloki wejściowe (Sources) i wyjściowe (Sinks), podstawowe operacje matematyczne (Mathematic operators), operatory statystyczne (Statistic operators), operatory liniowe (Linear dynamics), operatory dyskretne (Discrete Operations), sterownie przetwarzaniem (Execution control), logikę klasyczną (Crisp logic), elementy nieliniowe (Nonlinear elements) i inne.

Biblioteki funkcyjne w postaci wtyczek mogą być opracowywane zupełnie niezależnie od modułu PExSim. Są to **biblioteki użytkowników**. Aby bloki funkcyjne dostarczane przez bibliotekę mogły być wykorzystywane przez moduł przetwarzania, muszą implementować odpowiednie interfejsy związane z przetwarzaniem sygnałów (ICTransformObject) oraz wyświetlania bloków w interfejsie graficznym (ICDisplayObject). Aby ułatwić tworzenie bibliotek użytkowników, system udostępnia **klasy bazowe bloków przetwarzania**. W klasach tych zaimplementowana jest większość właściwości i zachowań charakterystycznych dla każdego z bloków (takich, które każdy blok musi realizować).

Z poziomu wymienionych interfejsów, każdy z bloków ma dodatkowo dostęp do:

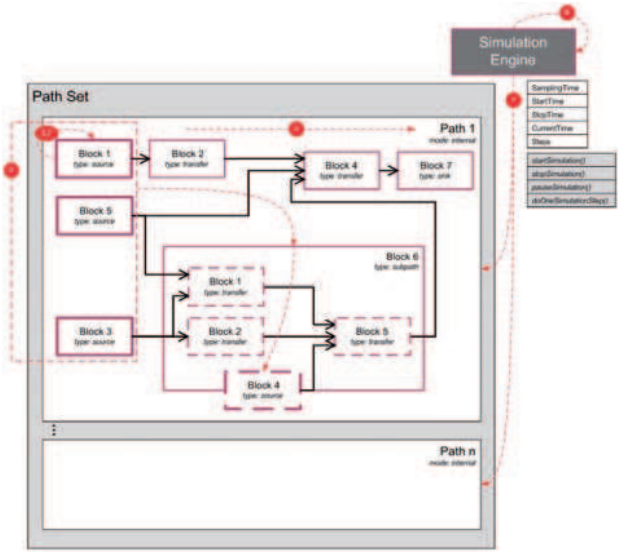
- interfejsów silnika odpowiedzialnego za symulację torów przetwarzania zmiennych, np. w celu uzyskania dostępu do parametrów symulacji
- archiwalnej bazy danych systemu DiaSter
- specjalistycznych baz danych tworzonych na życzenie użytkowników
- centralnej konfiguracji systemu DiaSter, np. w celu przeglądania zdefiniowanych zmiennych systemowych, ładowania modeli parametrycznych identyfikowanych za pomocą innych narzędzi systemu DiaSter itp.

Powyższy mechanizm pozwala na realizację bardzo ciekawych i zaawansowanych algorytmicznie bloków funkcyjnych. Przykładem może być realizacja bloków zajmujących się wnioskowaniem na podstawie przypadków (*Case Base Reasoning*). Bloki takie mogą łączyć się z bazą danych, w której zapisane są wzorcowe przypadki i na ich podstawie odpowiednio przetwarzać sygnały wejściowe.

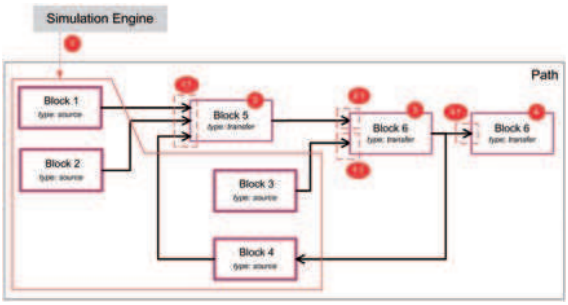
Zasady synchronizacji obliczeń pomiędzy blokami funkcyjnymi

W systemach umożliwiających elastyczną budowę ścieżek przetwarzania zazwyczaj pojawia się problem kolejności wywoływania obliczeń dla poszczególnych bloków. W przypadku modułu PExSim, ze względu na możliwość swobodnego łączenia wejść i wyjść poszczególnych bloków oraz tworzenia sprzężeń zwrotnych, problem urasta do rangi problemu pierwszoplanowego.

Rozwiązanie zaproponowane i wdrożone w module PExSim opiera się na zasadzie propagacji informacji i wyzwalania obliczeń w kolejności determinowanej przez przepływ sygnałów definiowany sposobem połączenia bloków (rys. 4). Głównym założeniem jest wydzielenie specjalnego typu bloków, nazwanych źródłami, które odpowiadają za generowanie sygnałów. Mechanizm symulacji (krok 1) dla każdej ścieżki poziomu głównego w pierwszym etapie wyzwala bloki będące źródłami, w kolejności definiowanej przez kolejność ich tworzenia (krok 2.1). Wyzwolony blok oblicza wartości swoich sygnałów wyjściowych i przekazuje wyniki obliczeń do następnych bloków (rys. 5). Pojawiające się sygnały na wejściach bloku standardowego (krok 1.1 na rys. 5) powodują jego wstępną aktywację, w trakcie której jest



Rys. 4. Schemat wyzwalania bloków – widok ogólny
Fig. 4. The scheme of blocks triggering – general view



Rys. 5. Schemat wyzwalania obliczeń – widok szczegółowy
Fig. 5. The scheme of calculations triggering - detailed view

sprawdzane, czy ustalone zostały wartości na wszystkich wejściach do danego bloku. Po pojawieniu się wszystkich sygnałów wejściowych blok jest wyzwalany i oblicza własne wartości wyjściowe (krok 2 na rys. 5), powodując wyzwolenie następnych bloków w strukturze przetwarzania.

W przypadku wystąpienia pętli zwrotnej system wymaga, by w jej skład wchodził również blok pełniący rolę źródła. Nie musi być to generator sygnału *sensu stricto* – musi istnieć jedynie możliwość obliczenia aktualnej wartości wyjścia bez znajomości aktualnej wartości wejść. Taki warunek spełniają np. blok opóźnienia czy też bloki dynamiki liniowej, w których dla każdego wejścia zdefiniowano opóźnienie na co najmniej jeden krok. W przypadku wystąpienia pętli algebraicznej nie ma możliwości jej rozwiązania, lecz nie wpłynie ona na działanie bloków niezwiązanych z tą pętlą. Omawiany mechanizm po prostu pominie pętle algebraiczne, a dodatkowy test, wykonywany po każdym kroku symulacji, podświetli te elementy struktury na czerwono.

Podścieżki wbudowane w ścieżki główne są obsługiwane według tego samego algorytmu, przy założeniu wyzwalania wejść podścieżki przez połączone wyjścia bloków ścieżki nadrzędnej (rys. 4).

Normalny przebieg informacji jest zgodny z kierunkiem połączeń pomiędzy blokami i przebiega zawsze w kierunku od wejścia bloku do jego wyjścia. Jednakże w uzasadnionych przypadkach każdy blok może wywołać przetwarzanie wsteczne, czyli odpytać bloki przyłączone do jego wejść o dodatkowe dane. Mechanizm ten nie jest wykorzystywany przez standardowe bloki, dostarczane wraz z platformą, jednakże umożliwia tworzenie przykładowo zaawansowanych bibliotek logiki rozmytej, gdzie każdy blok przetwarzający sygnał rozmyty może odpytać wstecz o jego definicję, która normalnie nie jest przesyłana.

Dodatkową funkcją jest możliwość zdefiniowania dla każdego z bloków tzw. krotności przetwarzania, to znaczy współczynnika ograniczającego wykonywanie danego bloku do jednego kroku na *n* kolejnych. Domyślnie wartość tego współczynnika wynosi 1, co oznacza że blok jest przetwarzany w każdym kroku symulacji. W przypadku jego zmiany na wartość *n* większą niż 1, blok jest przetwarzany standardowo tylko co *n* kroków symulacji, natomiast w pozostałych krokach wartości wejść są ignorowane i po aktywacji danego bloku na wyjściu jest powtarzany sygnał z poprzedniego kroku obliczeń.

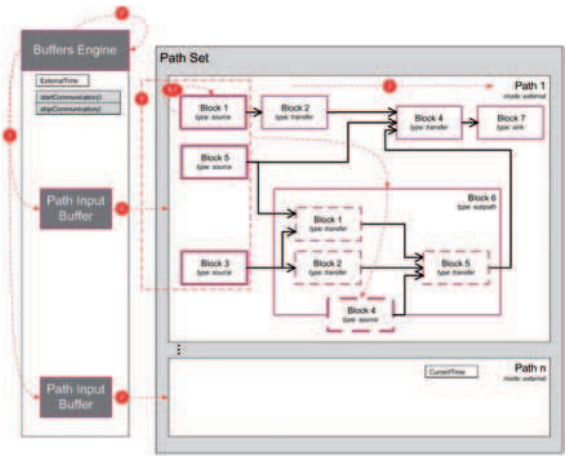
Tryby przetwarzania ścieżek

Wyzwalaniem obliczeń w ścieżkach przetwarzania steruje **jądro symulacji**. Jego zadanie polega na odmierzaniu aktualnego czasu symulacji (zgodnie z zadanym okresem przetwarzania) oraz wyzwalaniu obliczeń dla kolejnych ścieżek przetwarzania.

Ścieżki wyzwalane są w kolejności zgodnej z tą, w jakiej umieszczone są na liście. Dodatkowo dla każdej ścieżki przetwarzania można określić krotność przetwarzania. Jest to liczba całkowita określająca, co ile kroków symulacji ma być wykonywana dana ścieżka przetwarzania.

W ramach danej ścieżki obliczenia wyzwalane są zgodnie z opisanymi zasadami – jądro symulacji wyzwala przetwarzanie dla bloków źródeł, a następnie kolejne bloki wyzwalane są przez odpowiednią propagację sygnałów. Jest to tzw. **tryb wewnętrzny** wyzwalania obliczeń.

Moduł PExSim może pracować także jako jeden z modułów wielomodułowego systemu rozproszonego. W tym przypadku pojawia się często konieczność wyzwalania obliczeń przez przychodzące nowe paczki danych z innych modułów systemu DiaSter. Jest to tzw. **tryb zewnętrzny** wyzwalania obliczeń. Proces odbierania danych zewnętrznych, ich synchronizacji (w tym np. określanie stempli czasowych dla nowego kroku symulacji) oraz ostatecznie wyzwalania obliczeń w ścieżkach sterowany jest przez specjalne **bufory wejściowe**.



Rys. 6. Schemat wyzwalania obliczeń – tryb zewnętrzny
Fig. 6. The scheme of calculations triggering – external mode

Za pomocą trybu wewnętrznego mogą być także wyzwalane ścieżki przetwarzania, które pobierają dane z innych ścieżek za pomocą specjalnych **buforów wewnętrznych**. Jest to jedna z metod przesyłania danych pomiędzy ścieżkami. Drugą metodą przesyłania są zmienne wewnętrzne – proste bufory przechowujące wartości wyjściowe z określonych ścieżek i umożliwiające odczyt przez specjalne bloki źródeł typu „Zmienna Wewnętrzna". Ten drugi mechanizm synchronizacji nie zapewnia właściwej kolejności wykonania obliczeń (zapełnienie bufora – odczyt), jeśli w buforze nie jest ustawione jakiekolwiek opóźnienie. W tym przypadku odpowiedzialność za prawidłowe wykonanie obliczeń spoczywa na projektancie algorytmu przetwarzania.

Trzeci tryb symulacji jest **trybem mieszanym** łączącym wybrane cechy dwóch trybów podstawowych. W tym trybie obliczenia wyzwalane są tak jak w przypadku trybu wewnętrznego, z tym że uwzględniane są dodatkowe kroki obliczeniowe związane z przychodzącymi danymi zewnętrznymi.

Wszystkie trzy tryby symulacji mogą być określane dla każdej ze ścieżek przetwarzania niezależnie. Zaimplementowane jądro symulacji ma także zabezpieczenie przed zbyt długim wykonywaniem obliczeń w poszczególnych ścieżkach (lub wręcz wstrzymaniem obliczeń). Podczas konfiguracji można określić, czy dla danej ścieżki ma być kontrolowany czas jej wykonania. W przypadku włączenia tej opcji przetwarzanie ścieżki zostanie przerwane, jeśli czas ten zostanie przekroczony. Aby mechanizm ten mógł być wykorzystany, symulacja ścieżek podzielona jest na oddzielne wątki. Ścieżki z ograniczonym czasem symulacji realizowane są w jednym wątku, natomiast ścieżki z kontrolą czasu wykonywane są w drugim wątku.

Wymiana danych ze światem zewnętrznym

Moduły przetwarzania pakietu PExSim mogą pracować niezależnie albo stanowić element większego systemu przetwarzania i analizy informacji. W drugim przypadku konieczna jest oczywiście wymiana danych ze światem zewnętrznym.

Podstawowy mechanizm wymiany danych obsługuje różnego rodzaju źródła danych (pliki, bazy danych, bieżące wartości zmiennych z serwera komunikacji systemu DiaSter itd.) zrealizowane w postaci odpowiednich bloków funkcyjnych typu **Źródło**. Powiązanie tych elementów z różnym trybem wyzwalania obliczeń w poszczególnych ścieżkach umożliwia realizację bardzo złożonych struktur przetwarzania danych.

Dodatkowy kanał informacji wykorzystuje mechanizm **zdalnego wywoływania procedur** (*Remote Procedure Call*). Z poziomu zewnętrznego oprogramowania, przez mechanizm komunikacyjny systemu DiaSter, można wywoływać zdalne procedury poszczególnych bloków funkcyjnych. Odpowiednie komunikaty (zgodne ze standardem RPC zaimplementowanym w systemie DiaSter) odbierane są przez moduł PExSim, a następnie przekazywane do określonego bloku funkcyjnego. Odpowiedni blok wyszukiwany jest na podstawie unikalnego identyfikatora. W tym celu blok funkcyjny musi mieć zaimplementowaną obsługę określonej procedury RPC. Przykładem wykorzystania tego mechanizmu może być realizacja bloku regulatora, którego nastawy mogą być zmieniane zdalnie lub może być wywoływana procedura samostrojenia.

Najbardziej elastycznym sposobem komunikacji w systemie jest zdalny dostęp do obiektów reprezentujących poszczególne ścieżki i bloki za pomocą mechanizmu **CORBA** (*Common Object Request Architecture*). Zdalny dostęp w tym przypadku oznacza dostęp do poszczególnych modułów, ścieżek i bloków zarówno z poziomu innych programów pracujących na tym samym komputerze, jak i programów pracujących na innych jednostkach połączonych siecią. Komunikacja realizowana w ten sposób jest asynchroniczna i niepowiązana z taktem symulacji realizowanym przez silnik obliczeniowy. Mechanizm CORBA umożliwia wywoływanie metod każdego z obiektów, a tym samym odpytywanie poszczególnych bloków o specjalistyczne parametry. Ta funkcjonalność jest wykorzystywana głównie przez zaawansowane wyświetlacze umieszczone w module wizualizacji, które mogą zarówno pobrać wartości parametrów poszczególnych bloków funkcyjnych, jak i ustawić ich nowe wartości.

Podsumowanie

Opisane techniki zapisu, przesyłania i przetwarzania cyklicznej informacji, zaimplementowane w systemie DiaSter, wykraczają poza możliwości typowych systemów automatyki dostępnych na rynku. Ich pełne wykorzystanie pozwala na łatwą realizację

wyrafinowanych technik przetwarzania sygnałów, takich jak logika rozmyta czy przetwarzanie informacji tekstowej. Omawiane cechy pozwalają z jednej strony wykorzystać moduł PExSim jako pełnoprawne środowisko symulacji w czasie rzeczywistym nawet złożonych instalacji przemysłowych, z drugiej – jako system bieżącego przetwarzania sygnałów niekoniecznie związany z zadaniami modelowania.

Możliwość implementacji własnych typów danych, w połączeniu z szerokimi możliwościami komunikacyjnymi i zaawansowanym silnikiem obliczeniowym o wielu opcjach konfiguracyjnych, pozwalają mieć nadzieję, że system jeszcze przez długi czas nie będzie ograniczał inwencji autorów w implementacji nowych, zaawansowanych algorytmów przetwarzania danych procesowych w trybie on-line, co pozwala wykorzystywać DiaSter jako wygodną platformę do prototypowych wdrożeń nowych rozwiązań.

Bibliografia

1. Blanke M., Kinnaert M., Lunze J.: *Diagnosis and Fault-Tolerant Control*. Springer 2003.
2. Kościelny J.M., Korbicz J.: *Inteligentny system sterowania i diagnostyki procesów przemysłowych DiaSter*. WNT 2009.
3. Kościelny J.M., Syfert M., Wnuk P.: *System zaawansowanego monitorowania i diagnostyki procesów przemysłowych – AMandD.* Control Engineering Nr 2 (45), marzec 2008, s. 40–44.
4. Janiszowski K., Wnuk P.: *PExSim – a novel approach to problem of investigation of complex dynamic systems in industrial environment.* Problemy Eksploatacji, 2007.

Signal processing in PExSim package: principles, techniques, capabilities

DiaSter is a software platform dedicated to advanced control, monitoring and diagnosis of industrial processes developed in Institute of Automatic Control and Robotics, Warsaw University of Technology. One of the main modules of the system is process variables processing package called PExSim. In this package interesting solutions for on-line signal processing are implemented. This article describes rules, techniques and possibilities for the user in area of signal processing in DiaSter system.

Keywords: decision support systems, data processing, modeling

dr inż. Michał Syfert

Jest zatrudniony w Instytucie Automatyki i Robotyki Politechniki Warszawskiej. Zajmuje się badaniami w dziedzinie diagnostyki procesów przemysłowych oraz zastosowań logiki rozmytej. Jest jednym ze współautorów systemów zaawansowanego monitorowania oraz diagnostycznego AMandD oraz DiaSter.

e-mail: m.syfert@mchtr.pw.edu.pl

dr inż. Paweł Wnuk

Jest zatrudniony w Instytucie Automatyki i Robotyki Politechniki Warszawskiej. Zajmuje się badaniami w dziedzinie nowych metod modelowania dynamiki systemów, ze szczególnym uwzględnieniem logiki rozmytej. Jest jednym ze współautorów systemów zaawansowanego monitorowania oraz diagnostycznego AMandD oraz DiaSter.

e-mail: p.wnuk@mchtr.pw.edu.pl