Technologia OPC, OPC UA, Technologia Zdalnego Wykonania Skryptów, OLE, COM/DCOM, SOAP, XML, DCS, Systemy automatyki przemysłowej, Akwizycja danych procesowych, Sterowanie procesem technologicznym, Sterowniki PLC, Urządzenia automatyki

KWIECIEŃ Roman[1]
SZYCHTA Leszek[2]
FIGURA Radosław[3]

WSPÓŁPRACA ZADAŃ W SERWERZE INFORMATYCZNEGO SYSTEMU SKRYPTOWEGO

Informatyczne serwery systemów automatyki przemysłowej są elementem Rozproszonego Systemu Sterowania DCS [1]. Pełnią one istotną rolę w procesie przepływu informacji z urządzeń automatyki do stacji inżynierskich, operatorskich, diagnostycznych oraz innych systemów komputerowych należących do wyższej warstwy nadzorowania przedsiębiorstwem. W związku z tym autorzy pragną przedstawić sposób współpracy zadań uruchomionych w serwerze Informatycznego Systemu Skryptowego ISS, za pomocą którego można realizować złożony proces sterowania obiektem technologicznym.

COOPERATION TASKS IN SERVER OF INFORMATION SCRIPTING SYSTEM

Computer servers of industrial automation systems are part of the Distributed Control System DCS [1]. They perform an important role in the flow of information from automation devices to the engineering station, operator, diagnostic, and other computer systems belonging to the higher layer controlling company. Therefore, the authors would like to present how the cooperation tasks running in the server of the Information Scripting System ISS, which can be used to implement a complex process of technological object control.

1. WSTĘP

Obecnie obowiązującym standardem komunikacyjnym używanym w systemach automatyki przemysłowej jest technologia OPC (ang. *OLE for Process Control*) [6-9]. Powstała ona w 1996r. i bazowała na najbardziej rozwiniętej w owym czasie technologii informatycznej COM/DCOM (ang. *Component Object Model* / *Distributed Component Object Model*), dawniej nazywaną jako OLE (ang. *Object Linking and Embanding*). Wady technologii OPC, wynikające ze stosowania technologii DCOM, spowodowały powstanie

[1] *Politechnika Radomska, Wydział Transportu i Elektrotechniki; 26-600 Radom; ul. Malczewskiego 29, r.kwiecien@pr.radom.pl*
[2] Politechnika Radomska, Wydział Transportu i Elektrotechniki; 26-600 Radom; ul. Malczewskiego 29, l.szychta@pr.radom.pl
[3] Politechnika Radomska, Wydział Transportu i Elektrotechniki; 26-600 Radom; ul. Malczewskiego 29, r.figura@pr.radom.pl

od 2007/2009r. technologii OPC UA (OPC Unified Architecture). Specyfikacja tej technologii bazuje na usługach sieciowych (ang. *Web Services*), w których treść informacji przesyłana jest w formie języka znaczników XML (ang. *eXtensible Markup Language*) za pomocą protokołu SOAP (ang. *Simple Object Access Protocol*). Wprowadzenie specyfikacji OPC Unified Architecture pozwoliło na przesyłanie informacji pomiędzy aplikacjami komputerowymi pracującymi w różnych systemach operacyjnych, przy jednoczesnym uniezależnieniu się od technologii informatycznych stosowanych wyłącznie w systemie operacyjnym Microsoft Windows.

Producenci urządzeń tworzą oprogramowanie nadzorujące pracę swych produktów w standardzie komunikacyjnym OPC. Dysponują tylko wytycznymi, do projektowania serwerów, które muszą wykonać od podstaw organizując cały proces właściwego przepływu informacji z urządzeń automatyki do klientów technologii OPC.

Alternatywą dla technologii OPC jest technologia Zdalnego Wykonania Skryptu RSE (ang. *Remote Script Execution*), na podstawie której został utworzony Informatyczny System Skryptowy ISS [2-5, 10, 11]. W skład tego systemu wchodzi serwer RSE zarządzający dowolnym urządzeniem automatyki, dzięki czemu producenci urządzeń mogą wykorzystać oprogramowanie systemu ISS do zaprojektowania tylko i wyłącznie sterownika informatycznego (ang. *Driver*), gdyż transfer informacji zapewniony jest przez warstwę komunikacyjną systemu ISS.

Serwer RSE jest aplikacją komputerową przeznaczoną do pracy wielozadaniowej, w której niezależne zadania (procesy) są wykonywane przez osobne wirtualne procesory. W zależności od przeznaczenia serwera, określone procesy mogą być zaprojektowane do wzajemnego współdziałania ze sobą. Z tego względu artykuł ten pokazuje sposób wpływania na przebieg zadania wykonywanego na serwerze poprzez inne zadanie należącego do tego samego środowiska uruchomieniowego.

2. TECHNOLOGIE KOMUNIKACYJNE

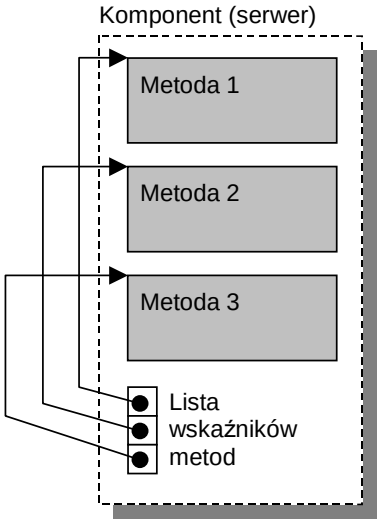
2.1 Technologia COM/DCOM

Początkowa idea technologii OPC opierała się na technologii łączenia i osadzania obiektów OLE (ang. *Object Linking and Embedding*), która w wersji 2.0 powstała w 1992r. Technologia OLE była rozszerzeniem protokołu dynamicznej wymiany danych DDE (ang. *Dynamic Data Exchange*), wprowadzonego na przełomie lat 80 i 90-ych w Microsoft Windows 3.x (dostępny też w OS/2 i Mac OS). Zrodziła się ona z potrzeby wymiany informacji pomiędzy obiektami, np. aplikacjami. Technologia OLE pozwala na kopiowanie danych z aplikacji serwera do aplikacji klienta, wraz z informacjami dotyczącymi serwera lub odwołaniem do pewnych informacji przechowywanych w rejestrze systemu operacyjnym Windows. Firma Microsoft rozbudowała OLE do OLE2 i rozpoczęła dodawanie nowych możliwości, takich jak automatyzacja OLE i kontrolki OLE. Kolejnym krokiem było zbudowanie shella Windows 95 z wykorzystaniem technologii OLE i interfejsów. Następnie dokonano zmiany nazwy kontrolek OLE, znanych jako OCX, na kontrolki ActiveX wraz ze zmianą specyfikacji, w celu umożliwienia dystrybucji prostych kontrolek poprzez Internet.

Ze względu na stale rosnące znaczenie technologii OLE dla platformy Windows, Microsoft zmienił tę nazwę na COM, a później COM+ dla Windows 2000. Zmiany te

w nazewnictwie są tylko częściowo związane ze zmianami technologicznymi, a w dużej mierze powodowane są działaniami marketingowymi.

Każda aplikacja, utworzona w technologii COM, posiada wykaz wskaźników przechowujących adresy pamięci operacyjnej do swych udostępnianych metod (funkcji i procedór). Znajomość adresów tych metod pozwala je wykonywać poprzez inne aplikacje komputerowe (rys. 1). Jest to bardzo waże stwierdzenie, gdyż aplikacja udostepniająca swoje zasoby (dalej nazywana komponentem lub serwerem) nie obciąża procesor jesdnoski komputerowej, tylko aplikacja wywołująca określone metody (dalej nazywana klientem).



Rys.1. Struktura organizowania obiektów budowanych w technologii COM.

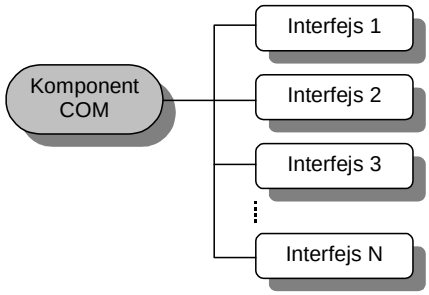
Klienci technologii COM, aby wywołać określoną metodę komponentu, muszą go odpowiednio identyfikować. Identyfikacje komponentów w postaci unikatowej 16-znakowej nazwy zapisane są w systemie operacyjnym, któch konfiguracja jest dostępna w panelu sterowania dla usług składowych. W rzeczywistości rejestracja komponentu pozwala na jego uruchomiuniu w systemie operacyjnym (zaalokowaniu do pamięci operacyjnej) i zapisamiu jego adresu w systemie. Za pomocą metod obsługujących kompunikaty (ang. *Message*) dostepnych w bibliotekach API jądra systemu operacyjnego możliwe jest uzyskanie adresu do serwera utworzonego w technologii COM, a tym samym do pożądanej jego metody.

W obiektach informatycznych definiuje się tzw. interfejsy do metod komponentu, czyli określa się typ metody (funkcję lub procedurę) oraz rozpoznawalne przez system operacyjny typy danych parametrów wywołania i rezultatu. Określenie interfejsu jest słuszne, gdyż klient technologii COM nie posiada implementacji metod, posiada tylko adres do niej, dla której musi załadować odpowiednią liczbę parametrów na stos procesora.

Obecnie technologia COM umożliwia efektywną komunikację między obiektami informatycznymi (rys. 2). Definiuje komponenty programowe niezależne od języka programowania, co umożliwia włączanie do tworzonych aplikacji elementów należących

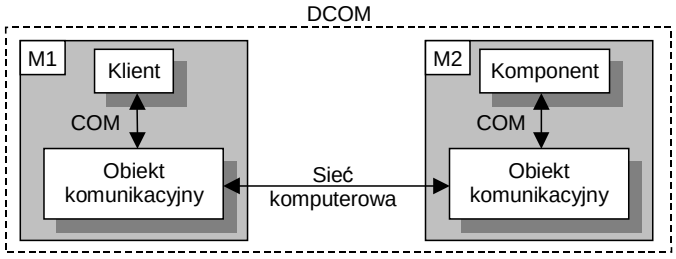
do innych programów i wymianę danych między poszczególnymi obiektami za pomocą interfejsów, przy czym:

- interfejs stanowi zbiór wskaźników do funkcji składowych komponentu COM (metod), zgromadzonych w tabelach vtable (ang. *Virtual Function Pointer Table*),
- interfejs nie jest obiektem, nie posiada własnej implementacji, to komponent COM implementuje interfejs,
- każdy komponent może implementować wiele interfejsów – oferować wiele zestawów usług,
- komponenty odwołują się do interfejsów za pośrednictwem wskaźników,
- każdy interfejs posiada własny, unikalny 128-bitowy identyfikator GUID (ang. *Globally Unique IDentifier*s). Nowa wersja interfejsu (np. z rozszerzonym zestawem funkcji) nie powoduje konfliktu ze starszą wersją – otrzymuje ona inny GUID.



Rys.2. Schemat interfejsowy obiektu technologii COM.

Mechanizm interfejsów pozwala na łatwe dodawanie nowej funkcjonalności do istniejących obiektów poprzez implementację nowych metod lub całych interfejsów przy zachowaniu kompatybilności z obiektami korzystającymi ze starszych metod/interfejsów. Równie proste jest usprawnianie istniejącej funkcjonalności przez wprowadzanie poprawek w implementacji obecnych metod.



Rys.3. Wymiana informacji pomiędzy obiektami informatycznymi; M1, M2 – jednostki komputerowe.

Ze względu na potrzebę wymiany informacji pomiędzy jednostkami komputerowymi, został poszerzony obszar technologii COM o obiekty komunikacyjne pracujące w sieci komputerowej (rys. 3), którą nazwano technologią DCOM (ang. *Distributed Component Object Model*). Łączy ona technologię COM, w której wykorzystuje się mechanizmy przekazywania komunikatów systemu operacyjnego Windows w obrębie lokalnego komputera oraz wymianę danych za pośrednictwem sieci komputerowej. W systemie operacyjnym pracują obiekty komunikacyjne, które pośredniczą w transmisji informacji pomiędzy aplikacją klienta a komponentem świadczącym określone usługi. Technologia DCOM zastępuje protokołem sieciowym lokalną komunikację między procesami, korzystając z technologii DCE RPC (ang. *Distributed Computing Environment / Remote Procedure Call*). Z punktu widzenia klienta nie ma różnicy, czy serwer znajduje się na tej samej jednostce komputerowej lub innej.

2.2 Technologia OPC

Na bazie technologii COM/DCOM powstała technologia OPC. Jej specyfikacje pozwoliły na zdefiniowanie standardu wymiany informacji w aplikacjach komputerowych pracujących w schemacie klient-serwer. Definiują one dla serwerów OPC oddzielne zadania pod względem ich funkcjonalności i obejmują:

- OPC Data Access (OPC DA) – umożliwia dostęp do aktualnych danych procesowych w trybie rzeczywistym,
- OPC Historical Data Access (OPC HDA) – umożliwia dostęp do danych archiwalnych,
- OPC Alarms & Events (OPC A&E) – rozgłasza zaistniałe zdarzenia w systemie oraz zgłaszane alarmy,
- OPC Security – definiuje sposób dostępu do danych,
- OPC Batch – jest wymagana podczas zarządzania wsadami,
- OPC and XML – integruje OPC i XML (ang. *eXtensible Markup Language*) w celu budowy aplikacji internetowych.

Specyfikacja OPC DA pozwala na dostęp do pojedynczej zmiennej procesowej (ang. *OPC Item*) z możliwością odczytu lub zapisu, z których każda posiada wartość (ang. *Value*), znacznik czasowy (ang. *Timestamp*), typ oraz jakość (ang. *Quality*). Znacznik czasowy może być generowany przez węzeł sieci lub przez serwer OPC, jeżeli węzeł nie ma takiej możliwości. Przy pomocy tej specyfikacji można przeglądać jedynie wartości aktualnych zmiennych procesowych lub zmieniać jej wartość. Ze względu na złożoność procesów realizowanych przez serwer OPC, dokonano logicznego podziału zmiennych procesowych na grupy (ang. *OPC Group*). W grupach tych zmienne charakteryzują się różnymi czasami skanowania oraz trybem odczytu.

Wprowadzenie standardu komunikacji OPC przyczyniło się do: standaryzacji komunikacji i wymiany danych przemysłowych, dużej uniwersalności i skalowalność rozwiązań oraz znacznego obniżenia kosztów integracji dużych systemów przemysłowych. W warunkach pracy sieciowej ujawniają się następujące wady komunikacyjne oparte na technologii DCOM, tj.: trudności w skonfigurowaniu połączenia, problemy z nawiązaniem połączenia, problemy z utrzymaniem połączenia pomiędzy klientem a serwerem.

Obecnie Fundacja OPC odchodzi od standardu komunikacyjnego opartego na technologii DCOM. Przyczyną tego faktu jest wieloletnia współpraca liderów

przemysłowych, których celem było stworzenie otwartego standardu wymiany informacji w systemach zarządzania procesem w sposób bogatszy i pełniejszy, zorientowany usługowo i bezpieczny w porównaniu do aktualnie wykorzystywanych standardów bazujących na platformie DCOM. Nowy standard nazwano OPC UA (*Unified Architecture*) i nie związano go z żadną istniejącą technologią komunikacyjną. Pierwotną technologię OPC wykorzystująca technologię COM/DCOM nazywano OPC Clasic.

Specyfikację OPC UA (Unified Architecture) wprowadzono w styczniu 2007r, która pod względem funkcjonalnym określa sposób realizacji trzech starszych specyfikacji: OPC DA, OPC HDA oraz OPC A&E. Bazuje ona na ogólnie przyjętych protokołach komunikacyjnych takich jak TCP/IP (ang. *Transmission Control Protocol* / *Internet Protocol*), HTTP (ang. *Hypertext Transfer Protocol*), SOAP (ang. *Simple Object Access Protocol*). Specyfikacja OPC UA umożliwia przesyłanie danych za pośrednictwem różnych formatów m.in. formatu opartego o usługi sieciowe Web Services i formatu binarnego. Serwer OPC zbudowany w oparciu o Unified Architecture definiuje swoim klientom zestaw usług, jakie oferuje oraz format danych procesowych za pośrednictwem którego ma odbywać się komunikacja.

Usługi sieciowe Web Services implementują rozproszone komponenty programowalne udostępniane za pośrednictwem protokołu SOAP. Komponenty usługowe Web Services mogą być implementowane z użyciem różnych języków programowania, platform sprzętowych i operacyjnych. Opisuje się je w skrypcie WSDL (ang. *Web Services Description Language*), który oparty jest na języku XML, w celu ułatwienia implementacji aplikacji klienckich. Dalszym rozwinięciem tego rozwiązania jest specyfikacja baz danych UDDI (ang. *Universal Description, Discovery and Integration*) umożliwiających gromadzenie informacji o dostępnych w sieci usługach Web Services.

2.3 Technologia RSE

Mając na uwadze wady powszechnie znanej technologii OPC Clasic, oraz tendencje rozwojowe OPC Unified Architecture, wykorzystano sprawdzone mechanizmy zdalnego pozyskiwania danych występujących w systemach baz danych do opracowania informatycznej technologii Zdalnego Wykonania Skryptu RSE (ang. *Remote Script Execution*). Technologia ta jest autorską i oryginalną technologią informatyczną przeznaczoną do zdalnego zarządzania:

- jednostką komputerową,
- jej zasobami (np. bazami danych),
- urządzeniami peryferyjnymi (np. kamera internetowa, pilot radiowy),
- urządzeniami automatyki przemysłowej (np. przemienniki częstotliwości, przyrządy pomiarowe, sterowniki programowalne itp.) pracującymi w komputerowej sieci przemysłowej, tj. Modbus, do której dostęp realizowany jest za pomocą komunikacji szeregowej przez interfejs RS 485 / RS 232 jednostki komputerowej,
- innymi obiektami.

Implementacja technologii RSE znalazła zastosowanie w budowie podsystemu systemu operacyjnego Microsoft Windows, który nazwano Informatycznym Systemem Skryptowym ISS (ang. Information Script System). Wymiana informacji w tym systemie realizowana jest za pomocą modelu komunikacyjnego RSEP (ang. *Remote Script Execution Protocol*) opartego o protokół komunikacyjny TCP/IP w sieci Ethernet. W modelu tym, podobnie jak

w systemach baz danych, można transmitować dwa rodzaje informacji: tekst oraz tabelaryczne obiekty danych. Tekstem kierowanym do serwera ISS jest skrypt obiektowego języka programowania PL#, którego struktura i składnia zbliżona jest do języka Pascal i Delphi Language. W treści skryptu wywołuje się instrukcje z należnymi jej parametrami, których implementacja należy do modułu wchodzącego w skład serwera RSE. Wywoływane instrukcje stanowią usługi sieciowe do wykonania przez serwer, od którego oczekuje się informacji zwrotnej w postaci wiadomości tekstowej oraz tabeli.

Do podstawowych usług serwera RSE zalicza się:

- zarządzanie drzewiastą bazą danych KDB,
- zarządzanie uruchomionymi procesami (zadaniami),
- zarządzanie systemem plików systemu operacyjnego Windows - zdalne lub lokalne kopiowanie plików, usuwanie oraz zakładanie nowych folderów lub plików itp.,
- zarządzanie strumieniem danych - operacje na zawartościach plików i pamięci operacyjnej.

3. WIELOZADANIOWA PRACA SERWERA RSE

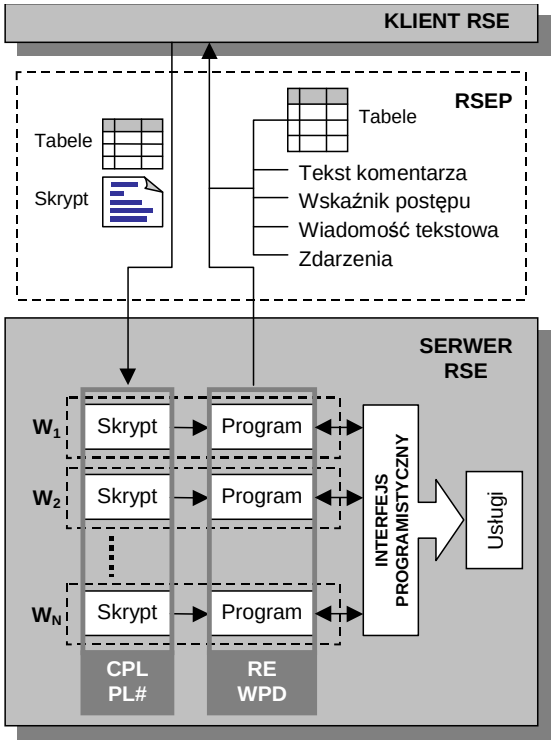
Serwer RSE jest aplikacją komputerową, w której każde zlecenie od klienta powoduje utworzenie samodzielnego wątku (zadania). Zadanie wyposażone jest w kompilator (CPL) obiektowego języka programowania PL# oraz wirtualny procesor Delphi (WPD), który wykonuje pośredni kod wynikowy (program) wygenerowany w wyniku procesu kompilacji skryptu (rys. 4). W trakcie wykonywania programu, może zostać przesłany komunikat zwrotny do klienta RSE w postaci: komentarza tekstowego, wiadomości tekstowej, wartości liczby całkowitej postępu wykonywania określonej czynności programu, tabelarycznego obiektu danych oraz zdarzeń pracy programu, takich jak: rozpoczęcie i zakończenie programu oraz ewentualnych błędów kompilacji lub wykonania programu.

Współpraca zadań w serwerze RSE odbywa się poprzez jego środowisko uruchomieniowe. Podobnie jak dla technologii COM (rys. 1), główną funkcjonalnością serwera jest możliwość pobrania adresu zadania przechowywanego w pamięci operacyjnej, na podstawie którego odczytuje się adres do odpowiedniej metody. Przykładem dwóch zadań wpływających na wzajemną pracę są programy z listingu 1 i 2.

Listing 1. Zadanie serwera o nazwie „test1“.

```
01  program test1;
02  {$APPTYPE CONSOLE}
03  var Stop: Boolean;
04  procedure DoStop;
05  begin
06    Stop := True;
07  end;
08  exports DoStop;
09  begin
10    Stop := False;
11    repeat
```

```
12      Sleep(1000);
13    until Stop;
14    PrintLn(Stop);
15  end.
```



Rys.4. Schemat serwera RSE; CPL – kompilator języka PL#; RE – środowisko uruchomieniowe; WPD – wirtualna procesor Delphi; W1..Wn – wątki serwera (zadania); RSEP – protokół zdalnego wykonania skryptu.

Listing 2. Zadanie serwera o nazwie „test2“.

```
01  program test2;
02  {$APPTYPE CONSOLE}
03  uses process;
04  var h: LongWord;
05      proc: procedure;
06  begin
07    h := prc_Addr(PChar('test1'));
08    if h > 0 then
09    begin
10      proc := libMethod(h, 'DoStop');
11      if Assigned(proc) then
```

```
12        proc else
13          PrintLn('No method found!');
14      end else
15       PrintLn('No process found!');
16    end.
```

Do komunikacji programów uruchomionych w osobnych procesach środowiska serwera używa się eksportowanych metod (tak jak w przypadku projektu biblioteki). Program „test1" (listing 1) wykonuje instrukcje w pętli co 1 sek. dopóki zmienna „Stop" nie będzie posiadać wartości logicznej „True" (wiersz 13). Osiągnięcie celu może być zrealizowane tylko przez wykonanie eksportowanej (wiersz 8) metody „DoStop" (wiersz 4), w której znajduje się instrukcja powodująca przypisanie zmiennej „Stop" wartości prawdy (wiersz 6). Program „test2" (listing 2) z kolei, odczytuje adres w pamięci operacyjnej programu „test1" (wiersz 7). W następnym etapie pobierany jest adres do eksportowanej metody „DoStop" (wiersz 10) w programie „test1". Wykonanie tej metody spowoduje przerwanie pętli w programie „test1" i jego zakończenie.

4. WNIOSKI

Przedstawiona w punkcie 3 niniejszego artykułu technologia zdalnego wykonania skryptu RSE została zaprojektowana w celu możliwości implementacji mechanizmu wielozadaniowości w procesie sterowania nie tylko urządzeniami automatyki przemysłowej. Z zastosowania skryptowych programów w serwerze ZWS wynikają następujące korzyści:

- klient systemu ISS uruchamia zadanie, w którym instrukcje programu wykonują określone usługi,
- każdy program zapisywany jest w postaci skryptu obiektowego języka programowania PL#, który można modyfikować przed wysłaniem do serwera,
- język PL# pozwala na wykorzystanie technik kompilacji do zarządzania urządzeniami pracującymi w przemysłowej sieci komputerowej.

Przedstawiony przykład sposobu współpracy zadań w środowisku uruchomieniowym serwera RSE pokazuje możliwości do organizowania całego procesu technologicznego. Pewne zadanie może realizować akwizycję danych procesowych, inne zadanie może analizować działanie pierwszego, aby na podstawie jego zmiennych podejmować decyzje do organizowania dalszej pracy serwera, np. dodanie kolejnego lub usunięcie odpowiedniego procesu.

5. BIBLIOGRAFIA

[1] Grochowski L., Rozproszone systemy informatyczne, *Dom Wydawniczy ELIPSA*, Warszawa 2003.

[2] Kwiecień R., Komunikacja serwerów SDC w systemie sterowania w komputerowej sieci przemysłowej. *SENE 2007*, Łódź, str. 539-544, ISBN 978-83-912711-4-8.

[3] Kwiecień R., Zastosowanie serwerów SDC w komputerowych systemach automatyki przemysłowej. *Komputerowe systemy wspomagania nauki, przemysłu i transportu „TRANSCOMP"*, Zakopane 2007, str. 447-452, ISSN 1230-7823.

[4] Kwiecień R., Sterowanie urządzeniami przemysłowymi. *Prace naukowe ELEKTRYKA*, NR 1(19) 2005, Radom, str. 143-148, ISSN 1507-3025.
[5] Kwiecień R., Szychta E., Szychta L., Data acquisition in OPC-based industrial IT systems, *The 4TH international conference on electrical and control technologies*, ECT 2009, ISSN 1822-5934.
[6] Postół M., Platforma integracji systemów zarządzania z produkcją (cz. 2). Głównie dla orłów, *Control Engineeging Polska*, Październik 2008, str. 16-24.
[7] Skura K., Zagadnienia integracji systemów informatycznych w automatyzacji procesów produkcyjnych w oparciu o technologię OPC. *Napędy i sterowanie*, nr 10, Październik 2007r.
[8] Postół M., Platforma integracji systemów zarządzania z produkcją (cz. 1). W poszukiwaniu złotego środka, *Control Engineeging Polska*, Wrzesień 2008, str. 16-22.
[9] Chrupek R., Akwizycja Danych w systemach przemysłowych, *Napędy i sterowanie*, nr 4, kwiecień 2008r.
[10] Kwiecień R., Szychta L., Figura R., Skryptowy informatyczny system sterowania urządzeniami automatyki przemysłowej, Przegląd Elektrotechniczny 2'2010, ISSN 033-2097, str. 285-288.
[11] Kwiecień R., Szychta E., Szychta L., Figura R., Wykorzystanie komputerowego systemu pomiarowego do badania silnika jednofazowego, Logistyka 6/2010, ISSN 1231-5478.