Krystyna MACEK-KAMIŃSKA, Marcin KAMIŃSKI, Grzegorz Paweł KORBAŚ
POLITECHNIKA OPOLSKA, KATEDRA ROBOTYKI I ZASTOSOWAŃ INFORMATYKI,
ul. Luboszycka 7, 45-036 Opole

Programowanie obiektowe w symulacji układów elektromechanicznych i estymacji parametrów

Dr hab. inż. Krystyna MACEK-KAMIŃSKA

Studia na Wydziale Elektrycznym Politechniki Wrocławskiej ukończyła w 1979r, a doktorat obroniła tam w 1983r. Habilitację uzyskała w 1994r. w AGH Kraków. Jest autorem 120 publikacji w tym 2 monografii oraz promotorem 3 doktoratów. Specjalizuje się w modelowaniu i estymacji parametrów układów elektromechanicznych oraz w zagadnieniach automatyzacji napędu elektrycznego. Od 2005r. pełni obowiązki dyrektora Instytutu Układów Elektromechanicznych i Elektroniki Przemysłowej.

e-mail: k.macek-kaminska@po.opole.pl

Dr inż. Marcin KAMIŃSKI

Studia na Wydziale Elektroniki Politechniki Wrocławskiej ukończył w 1982r. W 2003 obronił doktorat na Wydziale Elektrotechniki, Automatyki i Informatyki Politechniki Opolskiej. Od 1984r. pracuje w Politechnice Opolskiej. Specjalizuje się w stosowaniu nowoczesnych techniki informatycznych w modelowaniu i estymacji parametrów układów elektromechanicznych. Jest członkiem PTETiS i SEP.

e-mail: m.kaminski@po.opole.pl

Streszczenie

W artykule zaprezentowano doświadczenia autorów w modelowaniu układów elektromechanicznych. Do przeprowadzania tych prac zostały opracowane autorskie aplikacje. Dzięki odpowiedniemu projektowi w oparciu o wykorzystanie techniki programowania obiektowego udało się w łatwy sposób przystosowywać ją do różnych wariantów obliczeń. Na jej bazie zbudowano specjalizowane narzędzia do obliczeń symulacyjnych napędów grupowych i do estymacji parametrów modelu matematycznego badanego obiektu. Pokazano też opis dedykowanego systemu do estymacji parametrów wraz z jego sieciowym rozszerzeniem.

Słowa kluczowe: układy elektromechaniczne, modelowania, języki programowania, programowanie obiektowe.

Object oriented programming in simulation of electromechanical systems and parameter estimation

Abstract

The paper presents the authors' experiences in modeling electromechanical systems. To carry out this work the original applications were developed. The project of application, based on object-oriented programming techniques, has allowed in an easy way adapting it to different variants of calculations and building on its basis the extended, specialized tools for numerical simulations of the group of drives and estimation of the object mathematical model parameters. The authors suggest that the choice of simulation software should satisfy the basic criterion - it should comply with standards. Not all tools used currently meet this criterion. With this approach it will be easy to prevent a situation where the new version of the tool will not work with source files developed in the previous versions. The second recommendation is separation of the code responsible for calculating from the code responsible for user's interaction, so that you can modify the application interface (as shown in Fig. 1). The paper presents also description of a dedicated system for parameter estimation and its network extension. The network version of this application can perform calculations in a flexible way by making use of any number of available computers.

Keywords: electromechanical systems, modeling, programming languages, object-oriented programming.

Dr inż. Grzegorz Paweł KORBAŚ

W 2001r. ukończył równolegle studia na Wydziale Elektrotechniki i Automatyki Politechniki Opolskiej oraz na Wydziale Matematyki, Fizyki i Chemii Uniwersytetu Opolskiego. W 2007r. obronił doktorat na Wydziale Elektrotechniki, Automatyki i Informatyki Politechniki Opolskiej. Specjalizuje się w badaniu metod optymalizacji oraz estymacji parametrów przy wykorzystaniu sieci komputerowej. Jest członkiem Polskiego Towarzystwa Fizycznego.

e-mail: g.korbas@po.opole.pl

1. Wstęp

Prowadzone od wielu lat w Katedrze Robotyki i Zastosowań Informatyki Instytutu Układów Elektromechanicznych i Elektroniki Przemysłowej Wydziału Elektrotechniki, Automatyki i Informatyki Politechniki Opolskiej prace z zakresu modelowania układów elektromechanicznych były związane z wykorzystywaniem i testowaniem szeregu narzędzi informatycznych. Początki tych prac sięgają czasów, kiedy tworzono oprogramowanie w języku FORTRAN na komputery typu SM-4, czy MERA 400. Pod koniec lat 80. XX wieku opracowane procedury były adaptowane na komputery klasy IBM-PC. Proces ten odbył się bez większych problemów dzięki temu, że język FORTRAN – jako nieliczny z dostępnych języków programowania w początkowej fazie rozwoju informatyki przypadającej na drugą połowę XX wieku – doczekał się standaryzacji. Zwiększające się możliwości obliczeniowe komputerów osobistych wraz z przełamywaniem kolejnych barier sprzętowych doprowadziły do powstania wielu atrakcyjnych narzędzi programistycznych. Niewątpliwie przełomem okazał się moment wprowadzenia do użytku systemu operacyjnego Windows 95, który, jako prekursor obecnie powszechnie wykorzystywanych kolejnych jego wersji, skutecznie zerwał z ograniczeniami systemu operacyjnego DOS.

Problem związany z wyborem najlepszego narzędzia do prowadzenia prac badawczych jest kluczowym i niesie szereg daleko idących konsekwencji, niejednokrotnie trudnych od razu do przewidzenia. Refleksja ta uzyskała dobitne potwierdzenie w momencie, kiedy wystąpiła konieczność porównania nowych wyników obliczeń numerycznych z podobnymi, przygotowanymi z użyciem kompilatora języka programowania FORTRAN w późnych latach 80. XX wieku (kompilator MS FORTRAN wersja 4). Wersje .exe przygotowanych ponad 20 lat temu aplikacji dają się uruchomić w oknie konsoli systemu Windows 7, a kod źródłowy jest możliwy do kompilacji współczesnym kompilatorem języka FORTRAN z minimalnymi poprawkami. Przykład ten pozwala krytycznie spojrzeć na popularne w ostatnim czasie narzędzia informatyczne, gdyż wiele z nich nie jest w stanie zapewnić swoim użytkownikom możliwości modyfikacji kodu źródłowego lub nawet uruchomienia wersji wykonywalnej aplikacji w dłuższej perspektywie czasowej. Dotyczy to na przykład niemożności wykorzystania plików źródłowych programu Matlab i jego przyborników, czy plików programu LabView przygotowanych w dużo wcześniejszych wersjach tych programów. Brak kompatybilności wstecznej powinien być podstawową przesłanką do krytycznej refleksji przy wyborze takiego narzędzia. To, że można z ich wykorzystaniem łatwo przygotować aplikację nie powinno przesądzać, że taki wybór jest najlepszy. Praca naukowa bowiem z reguły nie kończy się na przekazaniu wersji wykonywalnej użytkownikowi, ale charakteryzuje się ciągłością badań, a co za tym idzie, częstymi nawiązaniami do wcześniej opracowanych algorytmów i procedur.

2. Współczesne standardy tworzenia oprogramowania

Dynamiczny rozwój technologii informatycznej sprawia, że nawet aplikacje tworzone na własne potrzeby przez pracowników nauki powinny spełniać powszechnie akceptowane standardy.

Do tych standardów zaliczyć można:

- graficzny interfejs użytkownika,
- programowanie zdarzeniowe.

Wymóg uwzględnienia tych standardów wynika wprost z cech obecnie wykorzystywanych graficznych systemów operacyjnych oraz możliwości większości narzędzi programistycznych dostępnych aktualnie na rynku.

Problem więc nie sprowadza się do pytania *czy*, ale do rozstrzygnięcia problemu *jak*. Dodatkowy wymóg, by opracowana aplikacja zachowała swoją przydatność na długi okres użytkowania, istotnie zawęża wybór potencjalnych narzędzi. Pozytywne doświadczenia związane z wykorzystywaniem języka programowania FORTRAN, wskazują przesłankę, która powinna być wzięta pod uwagę przy wyborze narzędzia informatycznego. Tą przesłanką jest wybór języka programowania, który:

- posiada swój standard,
- nie jest wyłączną własnością firmy bądź korporacji.

Przyjęcie powyższych założeń powoduje, że bardzo popularne, zwłaszcza wśród studentów kierunku informatyka, narzędzia informatyczne produkcji firmy Microsoft – platforma .NET i język programowania C#, nie spełniają wskazanych wyżej przesłanek i ewentualna decyzja o ich wyborze powinna być podjęta z dużą rozwagą. Wcześniejsze doświadczenia autorów z próbą wykorzystywania języka C++ w środowisku Visual C++ firmy Microsoft nie były zachęcające, gdyż kolejne wersje tego środowiska wprowadzały znaczące zmiany, które utrudniały wykorzystywanie wcześniejszych wersji procedur. Innego rodzaju sytuacja związana jest z językiem Java – jedną z najpowszechniej wdrażanych do tej pory technologii, którego twórca – firma Sun Microsystems – została w kwietniu 2009 roku wykupiona przez innego potentata na rynku informatycznym, firmę Oracle. Szereg innych znanych producentów oprogramowania, w tym IBM i SAP, oparło wcześniej swoje rozwiązania na technologii Java i obecnie stali się jedynie biernymi obserwatorami poczynań ich głównego konkurenta na rynku – obecnego właściciela tej technologii.

3. Programowanie obiektowe

Kwestia wyboru języka programowania i platformy jego wdrażania jako narzędzi budowy aplikacji, nie jest jedyną decyzją, którą należy podjąć. Kolejny problem to wybór sposobu budowy swoich aplikacji. Obok proceduralnego (imperatywnego) paradygmatu programowania, charakterystycznego dla języków FORTRAN, Pascal i C, coraz większego znaczenia nabiera paradygmat programowania obiektowego, znany już od końca lat 60. XX wieku. Przez kilka dziesięcioleci technologia programowania obiektowego z trudem zdobywała sobie należne jej miejsce. Jej znaczenie wzrosło, kiedy na rynku pojawiły się nowe narzędzia informatyczne wykorzystujące ten paradygmat. W 1983 roku Bjarne Stroustrup przedstawił język C++. Język ten charakteryzuje się tym, że umożliwia korzystanie zarówno z paradygmatu programowania proceduralnego, jak i obiektowego, wysoką wydajnością kodu wynikowego, bezpośrednim dostępem do zasobów sprzętowych i funkcji systemowych, łatwością tworzenia i korzystania z bibliotek. Dodatkowo język ten zawiera mechanizmy programowania generycznego, dzięki któremu można, za pomocą uogólnionego podejścia, przygotować kod możliwy do wykorzystania w szerokiej gamie zastosowań. Praktyczną realizacją takiego podejścia jest włączona do standardowej biblioteki języka biblioteki szablonów (STL – Standard Template Library). Biblioteka ta daje programiście nowe, bardzo wygodne, w porównaniu z tradycyjnymi metodami zaimplementowanymi w językach C/C++, narzędzie do obsługi tablic dynamicznych dowolnych rozmiarów [1].

W roku 1995 pojawiła się Java – nowy język programowania opracowany pod kierunkiem Jamesa Goslinga z firmy Sun Microsystems. Język ten, mimo, że jego składnia jest zbliżona do składni języka C++, odróżnia się od niego dwoma zasadniczymi cechami: jest językiem w pełni obiektowym i jest językiem niezależnym od platformy, na której jest uruchomiony. Poza tym wyeliminowano w nim te cechy programowania, znane z języków C/C++, które były najczęstszym źródłem błędów.

Niewątpliwym impulsem rozwoju techniki programowania obiektowego był fakt, że realizowane projekty informatyczne stawały się coraz bardziej skomplikowane, a w związku z tym proces ich budowy, testowania i wdrażania wydłużał się, zaś współpraca w wieloosobowych zespołach programistów napotykała na trudności. Nowa technika programowania dawała szansę usprawnienia tego procesu. Struktura oprogramowania obiektowego stara się bowiem przybliżać świat rzeczywisty w znacznie większym stopniu, niż oprogramowanie tradycyjne. Praktycznie przejawia się to w tym, że w podejściu tradycyjnym rzeczywistość zostaje opisywana za pomocą niewielu dostępnych struktur danych, takich jak: zmienne proste podstawowych typów (całkowite, rzeczywiste, logiczne), tablice, rekordy. W programowaniu obiektowym programista może projektować na swoje potrzeby nowe typy zmiennych – nazywane klasami a następnie tworzyć zmienne – egzemplarze lub tzw. instancje tych klas. Sama klasa zawierać może zarówno elementy statyczne (pola) w postaci zmiennych prostych, tablic, innych klas jak i elementy dynamiczne (metody) w postaci funkcji i procedur operujących na polach. Dodatkowo istnieje możliwość ustawiania praw dostępu zarówno do pól, jak i metod, (obiekt/metoda: publiczna, ukryta), co pozwala je zabezpieczyć przed nieautoryzowanym dostępem. Klasa w technice programowania obiektowego może być wykorzystana do tworzenia nowych, bardziej specjalizowanych klas poprzez rozszerzanie ich funkcjonalności przez definiowanie w klasie pochodnej nowych pól i metod. Unika się w ten sposób problemu wielokrotnego powtarzania kodu. W programowaniu obiektowym nacisk jest położony na tworzenie elementów ogólnego przeznaczenia, w związku z tym łatwo jest je zastosować w innych programach.

Autorzy artykułu jako język programowania, wykorzystywany do tworzenia aplikacji symulacyjnych, wybrali język C++. Od końca lat 80. XX wieku dostępnych było kilka kompilatorów tego języka. W momencie pojawienia się graficznych systemów operacyjnych z rodziny Windows, kompilatory języka C++ zwiększyły swoją funkcjonalność o możliwość tworzenia graficznego interfejsu użytkownika i wykorzystywały paradygmat programowania zdarzeniowego. Kompilatory języków programowania zaczęły być częścią tzw. zintegrowanych środowisk programistycznych, umożliwiających zarówno tworzenie kodu aplikacji w wybranym języku programowania, budowę interfejsu użytkownika i łatwego programowania zdarzeń związanych z interakcją użytkownika z interfejsem. Komponenty, za pomocą których buduje się elementy interfejsu użytkownika, stały się dostępne w postaci specjalizowanych bibliotek, np. Visual Component Library (VCL) – biblioteki opracowanej przez firmę Borland na potrzeby środowiska Delphi i następnie zaadaptowanej do środowiska C++ Builder oraz biblioteki Microsoft Foundation Classes (MFC) – opracowanej na potrzeby środowiska Visual C++. Oprócz wymienionych powstało też wiele innych bibliotek (wxWidgets, Tk, Qt) dostępnych z poziomu różnych języków programowania. Obie wskazane wyżej biblioteki są zaprojektowane w sposób obiektowy, dzięki czemu możliwym stało się tworzenie elementów projektowanego interfejsu w sposób programowy i, co ważniejsze, można dynamicznie, w zależności od rzeczywistego kontekstu obliczeń, kształtować postać tego interfejsu.

W aplikacjach tworzonych w języku programowania C++ korzystano z zintegrowanego środowiska programistycznego Borland C++ Builder. W porównaniu do konkurencyjnego produktu firmy Microsoft – Visual C++ – charakteryzował się on znacznie większym komfortem pracy i przejrzystym projektem biblioteki komponentów VCL.

4. Aplikacja do modelowania układów elektromechanicznych

Omawiana aplikacja powstała około roku 2000. Przez ten okres czasu podlegała jedynie kosmetycznym poprawkom. Aplikacja pozwala wskazać model, za pomocą którego symulowany jest wybrany układ elektromechaniczny. Programowa definicja modelu składa się z dwóch procedur definiujących:

1. elementy modelu, które nie zależą od elementów obliczanego wektora stanu,
2. elementy modelu, których wartości zależą od aktualnie wyznaczanych wartości wektora stanu podczas procesu obliczeniowego.

Dodatkowe dwie procedury, które należy przygotować, służą do zdefiniowania wyrażeń służących do wyliczania funkcji wyjścia oraz procedury wykorzystywanej do całkowania układu równań różniczkowych.

Poniższa linia kodu tworzy egzemplarz klasy TSymulacja na podstawie zdefiniowanych procedur opisującymi konkretny model, sposób wyliczania funkcji wyjścia i wybraną metodę całkowania układu równań różniczkowych:

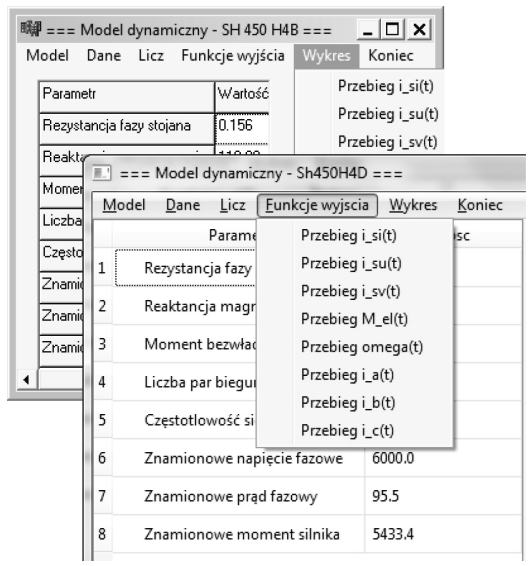
```
zadanie = New TSymulacja(ModelElementyStale, Model,
            FunkcjeWyjscia ,Fehlberg);
```

W klasie TSymulacja jest zdefiniowana odpowiedniej metoda, która, inicjowana poprzez wybranie opcji w menu głównym aplikacji, uruchamia proces obliczeniowy. Interfejs aplikacji, opracowany z użyciem komponentów dostępnych w środowisku programistycznym Borland C++ Builder, pozwala zaprogramować podstawowe etapy pracy: wybór pliku z danymi do obliczeń, sposób prezentacji wyników obliczeń (wykres, pliki tekstowe). Interfejs tej aplikacji przedstawiono na rysunku 1.

Aby uniezależnić się od konkretnej realizacji biblioteki komponentów graficznego interfejsu użytkownika, w projekcie aplikacji wyraźnie rozdzielono fragmenty kodu odpowiedzialne za obliczenia numeryczne od fragmentów, które odpowiedzialne są za procedury interakcji użytkownika z elementami interfejsu graficznego. Dzięki temu, w momencie, w którym została podjęta decyzja o zmianie biblioteki VLC na inną, operacja ta zostanie przeprowadzona z minimalną ingerencją w kod aplikacji. Powodem podęcia tej decyzji był fakt, że firma Borland zaniechała rozwoju swoich środowisk programistycznych opartych o własne, bardzo wydajne kompilatory. Na rysunku 1. pokazano wygląd formatek aplikacji przygotowanych z wykorzystaniem dwóch narzędzi: biblioteki VLC filmy Borland i biblioteki Qt4, obecnie wspieranej przez firmę Nokia.

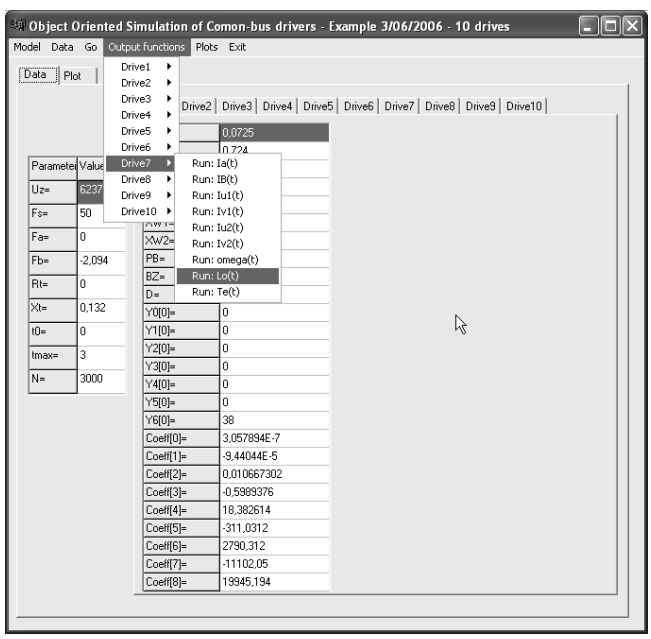
Omawiana aplikacja od wielu lat służy do przeprowadzania symulacji pracy układów elektromechanicznych. By rozszerzyć jej funkcjonalność o obliczenia dla nowego obiektu, należy do projektu dołączyć nowo opracowane procedury, dodać nową pozycję do menu Model wraz z oprogramowaniem zdarzenia związanego z kliknięciem myszką w tą pozycję menu i ponowna kompilacja całości projektu. Czynności te zajmują kilka minut. Dysponując zbiorem gotowych zestawów procedur można je wszystkie, w przejrzysty sposób, zawrzeć w jednej aplikacji.

Wykorzystując przetestowany obiektowy projekt aplikacji symulacyjnej, na jej bazie został opracowany program do modelowania stanów dynamicznych napędów grupowych [2]. Wykorzystując obiektowe cechy środowiska C++ Builder, udało się opracować jedną, uniwersalną aplikację umożliwiającą przeprowadzenie różnorodnych symulacji z uwzględnieniem dowolnej liczby współpracujących napędów. W zależności od liczby napędów uwzględnianych w obliczeniach aplikacja dynamicznie buduje swój interfejs, m.in. uwzględniając odpowiednią liczbę napędów w menu aplikacji oraz w komponencie PageControl, w którym wyświetlane są parametry poszczególnych napędów, co pokazane jest na rysunku 2.



Rys. 1. Interfejsy aplikacji opracowanej do symulacji układów elektromechanicznych przygotowanej odpowiednio w środowisku Borland C++ Builder i Qt4

Fig. 1. Application interfaces developed for simulation of electromechanical systems prepared in Borland C + + Builder and Qt4 development environment, respectively



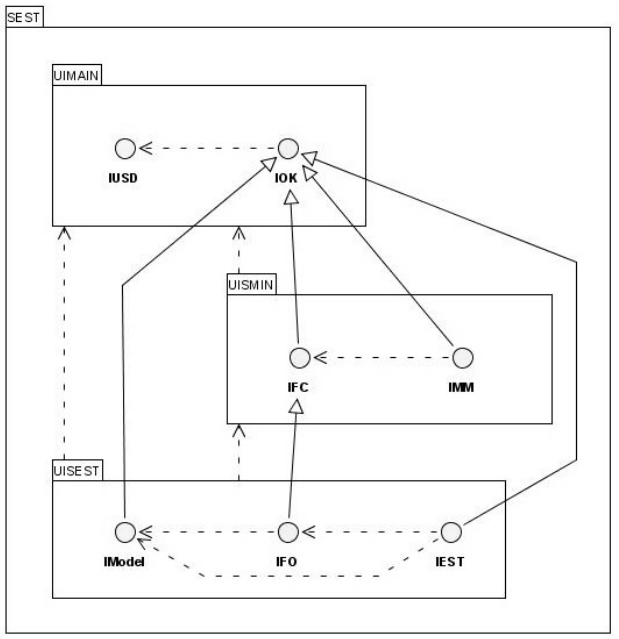
Rys. 2. Interfejs aplikacji przeznaczonej do symulacji napędów grupowych potrzeb własnych elektrowni

Fig. 2. Application interface designed for simulation of common-bus drives

5. Propozycja narzędzia do estymacji

W celu utworzenia systemu przeznaczonego do estymacji parametrów (nazwanego SEST) zastosowano jedną z typowych metod projektowania systemów informatycznych opartą o analizę przypadków użycia. W praktyce oznacza to, że na początku projektowania systemu tworzy się spis wymagań funkcjonalnych systemu, który następnie jest szczegółowo analizowany i uzupełniany. Doprowadziło to do powstania opisu abstrakcyjnej warstwy systemu obiektowego, którą stanowi zbiór interfejsów stosowanych w systemie oraz powiązań pomiędzy nimi, co pokazuje rysunek 3.

Otrzymano system składający się z trzech podsystemów: obiektów podstawowych, podsystemu minimalizacji i podsystemu estymacji. Dopiero na tym etapie nastąpiła decyzja związana z wyborem narzędzia, w którym taki system może być zaimplementowany.

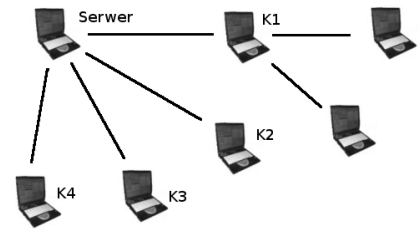


Rys. 3. Interfejsy w systemie estymacji parametrów
Fig. 3. Interfaces in the parameter estimation system

W krótkim czasie okazało się, że choć aplikacja działa poprawnie, to w wielu analizowanych procesach estymacji obliczenia są długotrwałe i warto znaleźć sposób na zwiększenie mocy obliczeniowej. Jednym z możliwych rozwiązań było zastosowanie do obliczeń istniejącej w Katedrze sieci komputerowej - wiele komputerów było bezczynnych poza zajęciami dydaktycznymi. Zastosowanie wcześniej współczesnych metod projektowania i programowania obiektowego umożliwiło bardzo łatwe i wartościowe rozbudowanie istniejącej aplikacji. Wystarczyło napisać rozszerzenie sieciowe, które w odpowiedni sposób pośredniczyło w komunikacji pomiędzy aplikacjami uruchomionymi na różnych komputerach. Warto tu jednak zauważyć, że rozszerzenie powstawało na podobnych zasadach jak cały system: począwszy od spisu wymagań funkcjonalnych, poprzez warstwę abstrakcyjną i ostateczną implementację. W normalnym przypadku aplikacja używa wbudowanego, wymiennego sterownika o interfejsie `ISterownik`, który zarządza procesami. W elastycznym systemie przewidziano jednak możliwość przesłaniania tego sterownika przez obiekty o interfejsie `ISterownikX`. Takim właśnie obiektem jest obiekt klasy `SterownikX_z_dll` który umożliwia odczyt opisu rozszerzenia sieciowego z biblioteki dołączanej dynamicznie (dll). Szczegółowy opis tych elementów znajduje się w [4]. Na rysunku 4 zaprezentowano schematycznie przykładową konfigurację sieci podczas pracy – generalnie stosowano topologię gwiazdy, przeznaczając na serwer komputer o największej wydajności.

Warto zauważyć, że ewentualna awaria serwera nie niszczyła działania reszty sieci – pozostałe komputery wykonywały wówczas estymację samodzielnie, a po ponownym włączeniu serwera wyniki były scalane. W przypadku niektórych procesów estymacji współpracowało ze sobą do czternastu komputerów.

Sieć była stosowana do estymacji parametrów różnych modeli układów elektromechanicznych zarówno w oparciu o dane symulacyjne jak i rzeczywiste pomiary. Największym wyzwaniem była równoczesna estymacja jedenastu parametrów modelu silnika indukcyjnego dwuklatkowego, który został zastosowany w przypadku pomiarów dynamicznych silnika indukcyjnego energooszczędnego [3]. Wprawdzie model nie dopasowuje się optymalnie do wejściowych przebiegów, jednak system zastosowany do estymacji definitywnie zdał egzamin, umożliwiając swobodną estymację w tym i innych przypadkach.



Rys. 4. Przykładowa konfiguracja sieci
Fig. 4. Exemplary configuration of the network

Narzędzie do estymacji zaprojektowane i zaimplementowane przy użyciu idei obiektowości sprawdziło się na każdym etapie i ze względu na swą elastyczność może być w dalszym ciągu rozbudowywane o nowe opcje.

6. Podsumowanie

Opisane badania, które ze względu na profil Katedry łączą elementy informatyczne z ich zastosowaniami w elektrotechnice wskazują, że zastosowanie obiektowych metod projektowania i programowania systemów informatycznych jest niezwykle wartościowe. Właściwe, współczesne podejścia umożliwia tworzenie wartościowych i elastycznych narzędzi, które można łatwo modyfikować i rozbudowywać dostosowując je do zmieniających się potrzeb. Jest ważne, że wspomniane zmiany i rozbudowa mogą być wykonywane nie tylko przez twórców systemów, ale przez wielu informatyków, dla których współczesna metodologia obiektowa jest już standardem.

Artykuł przedstawia dwa przykładowe narzędzia dedykowane obliczeniom symulacyjnym oraz estymacji parametrów. Podejście obiektowe zastosowane w obydwu przypadkach przez różnych autorów, umożliwiło nie tylko uzyskanie właściwie działających narzędzi, w postaci aplikacji. Warte uwagi są również cechy kodu, który te aplikacje tworzy: jest łatwy w konserwacji, elastyczny, można posłużyć się jego fragmentami w zupełnie innych systemach, ponadto – co jest wartością równie istotną – jest zrozumiały dla całej rzeszy współczesnych informatyków, którzy mogą go wspierać i rozwijać.

7. Literatura

[1] Baron B., Piątek Ł.: Metody numeryczne w języku C++, Wydawnictwo Helion, Gliwice, 2004
[2] Macek-Kamińska K., Kamiński M., Simulation of common-bus drivers with use of Object Oriented Programming, Int. Conf. on Electrical Machines, Chania, Greece, September 2-5, 2006.
[3] Macek-Kamińska K., Korbaś G.P.: Estymacja parametrów silnika energooszczędnego, Przegląd elektrotechniczny, Nr 3, 2009.
[4] Korbaś G.P.: Obiektowy system estymacji parametrów układów elektromechanicznych, Rozprawa doktorska, Opole, wrzesień 2007.

otrzymano / received: 25.11.2010
przyjęto do druku / accepted: 02.02.2011 *artykuł recenzowany*