POZNAN UNIVERSITY OF TECHNOLOGY ACADEMIC JOURNALS
No 82 Electrical Engineering 2015

Jarosław M. SZYMAŃDA*

# MODELOWANIE SYMULACYJNE Z PAKIETEM DYDAKTYCZNYM *SiSTLab20*

W referacie przedstawiono propozycję implementacji algorytmów w zakresie programowania podstawowego. Pod pojęciem programowania podstawowego przyjęto nabytą umiejętność programowania niezależnie od poziomu semantycznego technik programowania, innymi słowy poziom zaawansowania projektów odnoszono do indywidualnych możliwości studentów. Szczególną uwagę zwrócono na korelację umiejętności indywidualnych z działaniami w zespołach projektowych. W kontekście implementacji algorytmów, przede wszystkim algorytmów numerycznych, przedstawiono pakiet dydaktyczny *SiSTLab20* opracowany w celu rozszerzenia możliwości prezentacyjnych aplikacji realizowanych w technologii programowania w trybie konsoli tekstowej i uruchamianych w 32/64 bitowych wersjach systemu operacyjnego Microsoft Windows.

SŁOWA KLUCZOWE: zespoły projektowe, kształcenie problemowe, programowanie podstawowe, dydaktyka szkoły wyższej, aplikacje zintegrowane

## 1. WPROWADZENIE

Umiejętność implementacji algorytmów w zakresie programowania podstawowego jest jednym z ważniejszych elementów kształcenia problemowego. Kształcenie problemowe i jego szerokie spektrum kształcenia wariantywnego jest przedmiotem wielu publikacji [1, 2]. W zależności od treści i form realizacji oraz kategorii problemów, można w nich wyróżnić następujące istotne procesy, takie jak: wytwarzanie sytuacji problemowej (zadania projektowe), formułowanie celów szczegółowych (*podproblemy* i zdania), określanie warunków koniecznych i wystarczających rozwiązania, zgłaszanie propozycji rozwiązań (argumenty uzasadniające), weryfikacja zgłoszonych propozycji (kontrargumenty) oraz ocena uzyskanych rozwiązań w aspektach praktycznych i teoretycznych. Cechą charakterystyczną tej metody jest samokształcenie, a w połączeniu z pracą w zespole projektowym, pozytywna rywalizacja. Zatem z punktu widzenia zamierzonego celu dydaktycznego, wybór właśnie metody kształcenia problemowego jest jak najbardziej uzasadniony. Dodatkowym walorem jest wymagana i nieunikniona ciągła interakcja nie tylko między nauczycielem i zespołem projektowym, ale również wśród wszystkich uczestników procesu

* Politechnika Wrocławska.

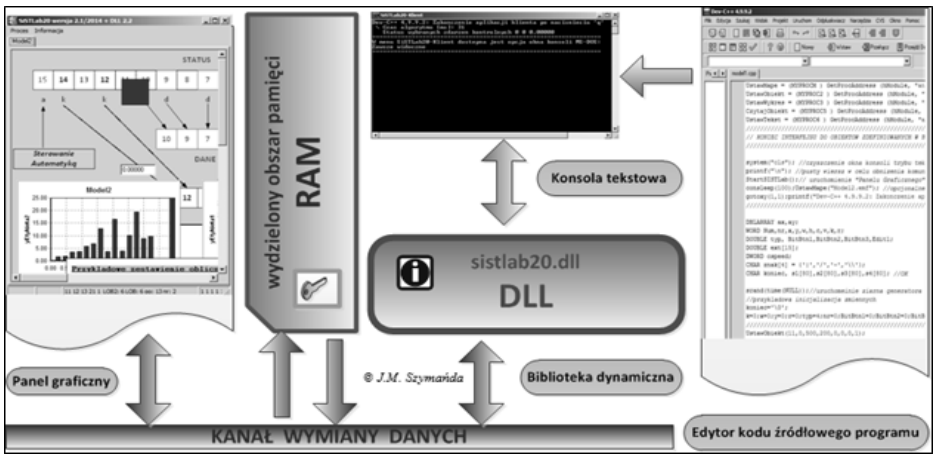
projektowania i to na każdym jego etapie. Efektywność samokształcenia, która może być zwiększana poprzez odpowiednio dobrane elementy z innych metod, takich jak: metoda przypadków (*case study*), giełda pomysłów (*brainstorming*), czy gry symulacyjne (*simulation game, mixed game*) [1, 2]. Przyjmuje się także, że rozwiązanie postawionego problemu jest jedynie możliwe, a przynamniej optymalnie skuteczne, po zastosowaniu odpowiednio wybranych procedur postępowania z bogatego repozytorium metodologii grupowego rozwiązywania problemów [1, 2]. W artykule pod pojęciem programowania podstawowego przyjmuje się nabytą umiejętność programowania niezależnie od poziomu semantycznego technik programowania, innymi słowy poziom zaawansowania określany jest w odniesieniu do indywidualnych możliwości studentów, którzy także mogą wchodzić w skład kilkuosobowych zespołów projektowych. W przypadku zespołów projektowych wymagane jest zapewnienie odpowiedzialnej koordynacji działań i odpowiedni do indywidualnych predyspozycji oraz zainteresowań przydział ról (funkcji). Szczegółowa analiza z tego zakresu przedstawiona jest między innymi w pracy twórcy koncepcji opartej na samoocenie Mereditha Belbina [1]. Współczesna technologia informatyczna (IT) oraz dostępne techniki programowania z jednej strony umożliwiają na bardzo kreatywne i sugestywne rozwiązywanie problemów, ale z drugiej ograniczają możliwość oceny wykorzystywanych algorytmów i zagrożeń w obliczeniach numerycznych. Jako przykład mogą być wymienione bardzo zaawansowane systemy typu CAD – czyli systemy wspomagające etapy projektowania, symulacji i eksperymentów numerycznych. Wysoka jakość większości z tych produktów wyznacza również uznawane standardy wymiany informacji oraz odniesień referencyjnych. Niestety w pewnych sytuacjach ich stosowane może prowadzić do poważnych błędów metodologicznych i merytorycznych. Poziom zagrożenia jest tym większy im mniejsze jest rozpoznanie i wiedza w danym zakresie. Czasami nawet w opracowaniach naukowych można dostrzec bezkrytyczne stosowanie programów obliczeniowych, efektem czego błędy obliczeniowe wynikające z arytmetyki liczb zmiennopozycyjnych o skończonej rozdzielczości (epsilon maszynowy) stają się przedmiotem wnioskowania merytorycznego o występujących korelacjach i obserwacjach eksperymentu symulacyjnego (!). Nawet najlepsze systemy, pakiety komercyjne i programy obliczeniowe nie zwalniają od istotnego rozpoznania zagadnienia w ramach którego zmierzamy do rozwiązania zadanego problemu. Podobny kontekst można także zaobserwować podczas opracowywania własnych programów obliczeniowych, w których poszczególne etapy kodowania algorytmu są lub nie są (!) oceniane pod względem stabilności obliczeń numerycznych. Często odwołujemy się do zewnętrznych bibliotek ułatwiających programowanie np. z zakresu algebry (rozwiązywanie układów równań, poszukiwanie wartości własnych, itp.), rachunku różniczkowo-całkowego, statystyki czy ukierunkowanych zagadnień technicznych jak optymalizacja, przetwarzanie sygnałów i wielu innych. Warto tutaj podkreślić, iż profesjonalnie opracowane oraz dostępne na rynku

produkty z tego zakresu (np. biblioteki numeryczne NAG [3]) są bardzo cenne i korzystanie z nich jest wysoce zalecane. Zawsze jednak po uprzednim dokładnym zapoznaniu się z odpowiednią dokumentacją, która powinna stanowić nieodłączny element zestawów dystrybucyjnych. Bez rozpoznania zastosowanych w danych procedurach algorytmów obliczeniowych i właściwej interpretacji parametrów wejściowo-wyjściowych włączenie ich do opracowywanego programu może być całkowicie błędnym podejściem. Należy także zauważyć, że prawidłowe działanie procedury i wykonane obliczenia nie oznaczają z definicji prawidłowego rozwiązania postawionego przez nas problemu. Z teorii algorytmów wynika, że algorytm numerycznie poprawny wcale nie musi być algorytmem numerycznie stabilnym, innymi słowy nasze rozwiązanie może być powtarzalne podczas każdego uruchomienia programu, ale powtarzane ze skorelowanym błędem numerycznym. Jest to sytuacja o największym stopniu zagrożenia, ponieważ bez świadomej analizy może doprowadzić do fałszywych wniosków. Nie można oczekiwać prawidłowego przybliżonego rozwiązania danego zagadnienia jeśli znany jest tylko jeden algorytm postępowania. W związku z powyższym, jeżeli jest to tylko możliwe, weryfikację obliczeń należy zawsze przeprowadzać wykorzystując inny niezależny algorytm. Z oceną niezależności algorytmów i zagadnieniami pokrewnymi można zapoznać się między innymi we wskazanej bibliografii [3, 4]. Znacznie „bezpieczniejsze" błędy to błędy krytyczne, które pojawiają się wtedy kiedy w pierwszej fazie programowania wprowadzony algorytm będzie algorytmem numerycznie niestabilnym i podczas jego uruchomienia zostanie zgłoszony wyjątek z zatrzymaniem programu włącznie. Podsumowując, rozpoczynając naukę w kontekście implementacji algorytmów obliczeniowych preferowane jest zatem programowanie podstawowe, ze szczególnym uwzględnieniem weryfikacji algorytmów pod względem ich stabilności numerycznych. Nabywane w tym zakresie doświadczenia powinny pomóc w podejmowaniu decyzji projektowych i we właściwej interpretacji otrzymywanych wyników.

## 2. PAKIET DYDAKTYCZNY *SiSTLab20*

Pakiet opracowano w celu rozszerzenia możliwości prezentacyjnych aplikacji realizowanych w technologii programowania w trybie konsoli tekstowej i uruchamianych w 32/64 bitowych wersjach systemu operacyjnego MS Windows. W skład pakietu wchodzą: panel graficzny o takiej samej nazwie wraz z biblioteką dynamiczną wymiany danych oraz aplikacją klienta. Pakiet jako zestaw dydaktyczny dedykowany jest do wspomagania zajęć laboratoryjno-projektowych z programowania podstawowego z elementami metod numerycznych i sieci komputerowych. Uproszczony schemat funkcjonalny organizacji pakietu przedstawiono na rysunku 1. Komunikacja

aplikacji klienta uruchomionej w trybie konsoli tekstowej z panelem graficznym realizowana jest poprzez dwa kanały komunikacyjne. Pierwszy, to dedykowana dynamiczna biblioteka DLL z przygotowanym interfejsem procedur realizujących zadania prezentacji i dwukierunkowej wymiany danych *klient-panel-klient*. Drugi, to wydzielony obszar pamięci RAM systemu operacyjnego dostępny wyłącznie dla aplikacji pakietu *SiSTLab20*. W tym obszarze wykonywane są niedostępne dla klienta wewnętrzne procedury pakietu. Jednym z podstawowych elementów opracowywania pakietu było założenie, że studenci uczestniczący w zajęciach nabyli już elementarną umiejętność programowania w języku programowania z możliwością kompilowania kodu źródłowego do postaci uruchomieniowej (*runtime*) w trybie konsoli tekstowej przy zachowaniu wszystkich standardowych własności tego trybu [4].




Rys. 1. Schemat funkcjonalny organizacji pakietu SiSTLab20

Uwzględniając wymagania programowe i plany dydaktyczne obowiązujące na Wydziale Elektrycznym Politechniki Wrocławskiej, opracowany pakiet został przygotowany do współdziałania z programami pisanymi w językach programowania: ANSI C w połączeniu z aplikacją Dev-C++ [4] oraz PASCAL w połączeniu z aplikacją FreePascal [4]. Obie aplikacje należą do grupy produktów tzw. wolnego oprogramowania i nie wymagają posiadania licencji na ich użytkowanie w celach niekomercyjnych oraz edukacyjnych. Aktualna dydaktyczna wersja pakietu *SiSTLab20* dystrybuowana jest tylko z interfejsami dla wyżej wymienionych wersji kompilatorów. Przyjęte rozwiązania otwartej architektury umożliwiają również na współpracę z aplikacjami z innych wersji kompilatorów akceptujących połączenia z bibliotekami dynamicznymi DLL, np. Borland Delphi, Microsoft Visual C++, C#, Java, VB a także programowania skryptowo-interpretacyjnego np. PHP i VBA [4].

### 2.1. Instalacja pakietu

Pakiet nie wymaga „typowej", znanej w systemie MS Windows instalacji. Wszystkie niezbędne do uruchomienia pliki należą tylko do jednego dedykowanego katalogu użytkownika. Aplikacje wchodzące w skład pakietu nie dokonują żadnych wpisów do rejestrów systemowych. Do korzystania z pakietu nie jest również wymagane posiadanie systemowych uprawnień administracyjnych, wskazane jest natomiast, ze względu na kompilacje programów, posiadanie uprawnień *użytkownika zaawansowanego*.

### 2.2. Ogólne zasady programowania klienta

Aplikacja klienta, niezależnie od wariantu języka programowania, podlega jednakowym zasadom organizacji programu. Poglądowy schemat funkcjonalny przedstawiono na rysunku 2. Można tutaj wyróżnić trzy podstawowe bloki programu:

a) blok inicjacji, w którym umiejscowione powinny być obiekty programu z wymaganym dostępem globalnym;
b) blok definicji funkcji, który jest opcjonalnym blokiem przeznaczonym na lokalne definicje funkcji wymagane podczas organizacji działania algorytmu;
c) blok główny programu zawierający wszystkie niezbędne informacje umożliwiające poprawne działanie algorytmu i współdziałanie programu z pozostałymi elementami pakietu.

Powiązanie z biblioteką dynamiczną *sistlab.dll* w zależności od wersji kompilatora kodu źródłowego, ustanawiane jest poprzez odpowiednie deklaracje typów i zmiennych funkcyjnych oraz uruchomienie procedur dedykowanych otwierających dostęp do biblioteki w trybie runtime (ładowanie dynamiczne DLL/ Windows). Właściwe powiązanie aplikacji *klient-panel* graficzny realizowane jest poprzez wywołanie w pierwszych liniach bloku głównego funkcji *StartSiSTLab*. Uruchomienie programu i wykonywanie zasadniczych elementów algorytmu określone jest przez pętlę główną programu oznaczoną na rysunku kolorem czerwonym. Do realizacji tej pętli w zależności od zastosowanego wariantu programowania wykorzystywane są instrukcje: *while* {}, *do* {} *while* , *while begin end*, *repeat until* lub *goto label*. Pętla główna jest powtarzana do czasu wykrycia zdarzeń kończących działanie klienta. Do zdarzeń takich mogą być zaliczone np.: naciśnięcie klawisza, testowanie statusu zmiennych algorytmu klienta i/lub modyfikacja statusu zmiennych w panelu graficznym *SiSTLab20*. W sekwencjach zamykających *działanie* klienta wywoływana jest funkcja zwalniająca wykorzystywane zasoby pakietu *StopSiSTLab*.

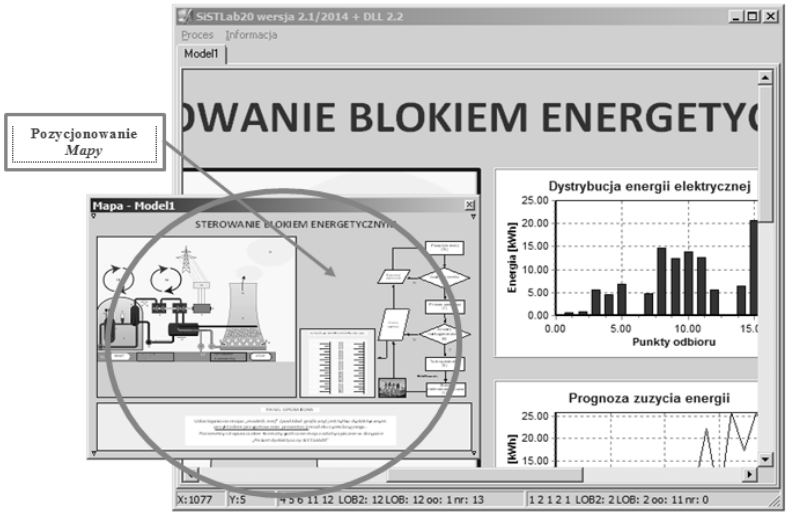
Z przykładami kodów źródłowych programowania klienta wraz z komentarzami można zapoznać się w dystrybuowanym zestawie plików *model1.pp [.cpp]* [4]. Po kompilacji kodu źródłowego program uruchamiany jest w trybie konsoli tekstowej MS-DOS systemu MS Windows bezpośrednio pod nadzorem procesora poleceń CMD (*command line procesor*). Program klienta oprócz możliwości prezentacyjnych panelu graficznego, może także wyprowadzać i wprowadzać informacje bezpośrednio do/z konsoli tekstowej. W tym celu biblioteka *sistlib. dll* uzupełniona została pomocniczymi funkcjami trybu konsolowego. Dla kompilatorów Delphi i FreePascal są to: *ClrScr, ConSleep, ReadKey, GotoXY, KeyPressed, SFormat* oraz dla kompilatora Dev-C++: *consleep(), gotoxy()*. W celu ułatwienia pracy nad projektem, do menu systemowego konsoli tekstowej MS-DOS dołączana jest opcja „*Zawsze widoczne*" poprzez którą można włączyć lub wyłączyć tryb *StayOnTop*


Rys. 2. Schemat funkcjonalny klienta

## 2.3. Atrybuty panelu graficznego

Możliwości prezentacyjne pakietu określone są poprzez system sterowania panelem graficznym *SiSTLab20*. W założeniach projektowych przyjęto, że wszystkie elementy graficzne, jako obiekty predefiniowane pakietu powinny być dostępne z poziomu programowania podstawowego klienta. Takie założenie eliminuje potrzebę dokształcania studentów (na tym etapie nauki) w zakresie kolejnych poziomów programowania grafiki oraz programowania obiektowego w kontekście systemu operacyjnego i programowania MS Windows API (*Application Programming Interfaces* – interfejs programowania graficznego aplikacji MS Windows). Można zatem większą uwagę zwrócić na zagadnienia stabilności algorytmów bez konieczności rozbudowywania programów o elementy i obiekty dodatkowe w danym wariancie języka programowania. Panel graficzny umożliwia przygotowanie prezentacji działania algorytmu wykorzystując

statyczne i dynamiczne obiekty graficzne. Do statycznych należy schemat prezentacji stanowiący podkład graficzny (mapę) w postaci rysunku lub rysunków wykonanych przy pomocy programu graficznego Microsoft Visio (dostępny dla studentów i pracowników w ramach subskrypcji MSDN AA –informacje na stronie internetowej Wydziału Elektrycznego http://www.weny.pwr.edu.pl) i/lub innych programów graficznych z możliwością zapisu plików w formatach: **.emf, *.gif* lub **.jpg*. Każda mapa w dowolnej chwili działania programu klienta może zostać podmieniona na inną. Należy zaznaczyć, iż zmiana taka może być także efektem działania algorytmu. Zalecane jest projektowanie prezentacji z wczytaną mapą określającą planowaną szerokość i wysokość panelu. Do obiektów dynamicznych zaliczamy wszystkie dostępne dla aplikacji klienta funkcje np.: *UstawObiekt()*, *CzytajObiekt()*, *UstawWykres()*, *UstawTekst()*. Każdy obiekt dynamiczny w pakiecie *SiSTLab20* oznaczony jest przez swój unikalny identyfikator (indeks), który przypisywany jest w bloku głównym klienta. Przykładowy widok panelu graficznego przedstawiono na rysunku 3. Większość zaimplementowanych funkcji posiada także parametry formalne, które precyzują zachowywanie się obiektu podczas komunikacji klienta z panelem graficznym. Istotnym elementem prawidłowego programowania obiektów dynamicznych jest zasada właściwej interpretacji parametrów formalnych funkcji i nadawanie parametrom aktualnym (podczas wywoływania) funkcji tylko dozwolonych wartości. W panelu graficznym oraz w bibliotekach dynamicznych pakietu większość parametrów aktualnych jest kontrolowana także w zakresie poprawności syntaktycznej. Aktualny zestaw dostępnych obiektów z opisem ich parametrów formalnych został zamieszczony w dokumentacji pakietu [4].




Rys. 3. Okno wirtualne z możliwością przeglądania panelu graficznego przy pomocy narzędzia Mapa

## 3. PODSUMOWANIE

Zaprezentowany pakiet wprowadzono do realizacji w semestrze letnim w roku ak. 2013/2014 w ramach kursu „Metody numeryczne w technice" dla studentów II roku studiów stacjonarnych na kierunku Elektrotechnika. Studenci podczas zajęć z projektowania w 3-4 osobowych grupach, z odpowiednio określonymi zadaniami i rozdziałem kompetencji, byli oceniani w kontekście założonych w „Karcie przedmiotu" tzw. „Przedmiotowych efektów kształcenia" w tym efektu: „*PEK_W02* - jest w stanie zaproponować odpowiedni algorytm numeryczny do rozwiązania zadania inżynierskiego". Przeprowadzona po zakończeniu semestru analiza wszystkich efektów potwierdziła poprawę nabywania przez studentów umiejętności umotywowanej diagnozy problemu w odniesieniu do zajęć realizowanych bez wymienionych innowacji. Innymi słowy, studenci w tym wariancie kształcenia mogli większą swoją uwagę skierować na istotę realizacji algorytmu numerycznego i jego prawidłowego zaprogramowania bez konieczności rozbudowywania programu i wykorzystywania systemowych procedur (obiektów) *prezentacyjno-graficznych* MS Windows API.

## LITERATURA

[1] Belbin R.M.: Management team: why they succeed or fail: London: Heimann 1981.

[2] Szymańda J.M: Wyznaczniki kształcenia problemowego inżynierów elektryków: XXXVI Międzynarodowa Konferencja z Podstaw Elektrotechniki i Teorii Obwodów, IC-SPETO2013, Gliwice-Ustroń, 22-25.05.2013, Politechnika Śląska, 2013, s.137-138.

[3] NAG- Numerical Algorithms Group (http://www.nag.co.uk/)Hyperlink: 14.07.2014.

[4] Szymańda J.M.: Pakiet dydaktyczny *SiSTLab20* – instrukcja użytkownika: Skrypt dydaktyczny PWR/W5/K1, 2014.

### MODELING SIMULATION WITH PACKAGE *SiSTLab 20*

Article draws attention to the elements of the problem in terms of learning basic programming. *SiSTLab20* educational package designed to expand opportunities for the presentation of applications implemented in programming technology in text console mode and run in 32/64 bit versions of MS Windows operating system is presented. The package includes: graphic panel, along with a library of dynamic data exchange and the client application. The package is dedicated to support laboratory-design classes with basic programming elements of numerical methods and computer networks.