STUDIA INFORMATICA
Nr 1(5) Systemy i technologie informacyjne 2005

KONCEPCJA PROGRAMOWEGO SYSTEMU DO TWORZENIA INTERAKTYWNYCH SYMULACYJNYCH APLIKACJI

Radosław Dropia[1], Piotr Czeberkus[2], Aleksander Timofiejew[3]

Abstract: Conception of program system creating of autonomously working interactive simulation applications is proposed

Key words: program simulation systems, interactive application generator, microcontroller projects

1. Wprowadzenie

Symulowanie obiektów z dziedziny przedmiotowej za pomocą interaktywnych programów komputerowych odgrywa poważną rolę dydaktyczną w procesie nauczania. Dla obiektów projektowanych symulowanie daje możliwość sprawdzenia nowego projektu. Autonomicznie pracujący program interaktywny zezwala na aktywne poznanie zasad funkcjonowania złożonego obiektu metodą gry (eksperymentowania) z modelem obiektu.

Zalety interaktywnych programów symulujących są dobrze znane, ale tworzenie symulatorów, zarówno urządzeń technicznych jak i zjawisk zachodzących w przyrodzie, nie jest rzeczą łatwą, ponieważ muszą one odzwierciedlać stany i przebieg zwykle złożonych procesów, które mają miejsce w symulowanym obiekcie. Dlatego do opracowania programu interaktywnego potrzeba dużo czasu i dobrej wiedzy o badanym obiekcie oraz trzeba dobrze opanować język programowania.

Rozwiązanie problemu szybkiego wytwarzania dobrej jakości programu autonomicznego dostosowanego do wymagań i potrzeb projektujących, nauczających lub uczących się w zasadzie jest możliwie, ale dzięki odpowiedniej metodyce i programowym narzędziom.

2. Analiza rozwiązań problemu

Do celu sprawdzania schematów zbudowanych z układów są wykorzystywane symulacyjne systemy - konstruktory. Przykładem systemu tego typu

[1] Akademia Podlaska, Koło Naukowe Informatyków GENBIT.
[2] Akademia Podlaska, Koło Naukowe Informatyków GENBIT.
[3] Akademia Podlaska Instytut Informatyki.

jest *Proteus* firmy *Labcenter Electronics*. Zawiera on *ProSPICE Simulator* oparty na powstałym na Uniwersytecie w Berkeley symulatorze *SPICE3F5*. System umożliwia symulację wielu (ponad 6000) układów, w tym mikrokontrolerów, w sposób bardzo dokładny i czytelny. Jego zaletą jest także współpraca z wieloma popularnymi kompilatorami i asemblerami.

Innym przykładem komercyjnego oprogramowania przeznaczonego dla mikroukładów firmy Microchip jest aplikacja pod nazwa *Oshon PIC Simulator*. Zawiera ona oprócz symulatora także prosty kompilator, asembler, deasembler oraz debugger. Program ten umożliwia ukazanie działania prawie 40 modeli układów firmy Microchip.

Producenci złożonych układów ofiarują też gotowe aplikacji symulujące schemat typowy dla danego układu lub grupy układów. Przykładem może być pakiet MPLAB firmy Microchip, który zawiera *Software Simulator* do symulacji działania zarówno jednostki centralnej układu mikrokontrolera, jak i wszystkich modułów peryferyjnych. Tworzenie go przez producenta mikrokontrolerów daje pewność, że w sposób dokładny odzwierciedla on zachodzące wewnątrz mikroukładu procesy, i pozwala mieć przekonanie, że program w rzeczywistym wykonaniu w mikrokontrolerze zachowa się tak jak podczas symulacji.

Szeroko wykorzystywany w procesie dydaktycznym system *MATLAB* poprzez pakiet *Simulink* umożliwia budowanie schematów z pojedynczych elementów, takich jak bramki logiczne, różnego rodzaju wejścia i wyjścia. System nie może być wykorzystany dla złożonych układów, ponieważ funkcjonalne schematy układów nie są dostępne.

Bliskim do rozwiązania problemu generacji aplikacji jest program *VisualSPARK* opracowany w *Berkley National Laboratory of the University of California*. Program generuje teksty programu w języku C, kompiluje, podłącza do biblioteki i uruchamia wytworzony program na polecenie użytkownika. Jednak *VisualSPARK* nie produkuje autonomicznego, interaktywnego programu wynikowego.

3. Wymagania do programowego generatora aplikacji

Większość obiektów przeznaczonych do symulacji zawiera części strukturalne (komponenty). Uniwersalny programowy generator aplikacji symulującej pewny obiekt musi łączyć moduły opisujące strukturalne komponenty obiektu. Dla funkcji, jakie realizuje każdy z komponentów, należy tworzyć ich odzwierciedlenie w języku programowania. Symulujący moduł powstający przy łączeniu modułów - komponentów musi być też modułem - komponentem dla obiektów z większą złożonością.

Każdy system symulowania musi pokazywać informację o stanie obiektu i jego komponentów. Znane systemy symulowania pokazują taką informację w jednym ze standardowych formatów stworzonych przez autorów systemu symulowania. Systemy symulowania oparte o języki typu VHDL pokazują użytkownikowi wartości zmiennych tylko w tabelach, na wykresach lub w opisach

nad liniami schematu. Dużo lepiej, jeżeli format informacji o stanie obiektu i jego komponentów mógł być stworzony przez nauczyciela, który lepiej wie, co pokazać uczniom w celu najlepszego poznania obiektu lub przez projektanta systemu, który dobrze wie, jaki właściwości systemu są ważne. Byłoby dobrze nie nakładać ograniczeń na złożoność modeli komponentów, a mianowicie niech model „sam" liczy wartości zmiennych, czasy zdarzeń, przekazuje komunikaty, wywołuje dialogi albo inne programy, generuje sygnały na wyjściach komputera lub w dowolny inny sposób komunikuje użytkownikowi swój stan, reaguje na polecenia użytkownika. Możliwość wizualizacji danych w postaci grafiki, pseudografiki i tekstów komentarza jest niezbędna zarówno w procesie nauczania, jak i w procesie projektowania.

4. System *Amethyst*

Przedstawione wymagania są realizowane w systemie *Amethyst* - generatorze interaktywnych aplikacji symulujących urządzenia techniczne.

System *Amethyst* i wytwarzane interaktywne programy przeznaczone są do działania w środowisku *MS Windows*. System korzysta z możliwości *MS Windows* do przedstawienia w przyjazny sposób dużej ilości informacji.

System generuje teksty interaktywnego programu w języku C++, dodaje do projektu niezbędne biblioteki kodów i teksty procedur i za pomocą kompilatora środowiska (np. *Borland C++ Builder*) produkuje programy wynikowe.

Zaletą proponowanego systemu jest to, że użytkownik może pracować z wyprodukowanymi tekstami jako z pełnym kompletem plików autonomicznego projektu środowiska (np. *Borland C++ Builder*), co umożliwia późniejsze modyfikowanie programu bez korzystania z aplikacji systemu.

System dostosowuje się do rozwiązywanych problemów przez włączenie innej biblioteki komponentów bazowych.

Umieszczany w interaktywnym programie wynikowym monitor symulowania umożliwia eksperymentowanie z badanym obiektem. W procesie eksperymentu można wielokrotnie ustawiać obiekt w stan początkowy, robić kroki z zadaną liczbą zdarzeń lub odstępów czasowych, w sposób ciągły przechodzić do końca wyznaczonego przedziału czasowego. W każdym momencie eksperymentu można oglądać stan dowolnego komponentu w jego oknie prezentacji stanu. W procesie symulowania system wpisuje do okna prezentacji stanu komponentu aktualne wartości zmiennych. Dane w oknie prezentacji parametrów można redagować przed symulowaniem oraz w procesie symulowania (w trybie symulowania krokami).

Tworzony model adekwatnie odzwierciedla statyczne funkcje urządzenia oraz jego właściwości dynamiczne (opóźnienie sygnału, filtracja krótkich sygnałów itp.). Możliwe jest symulowanie urządzeń cyfrowych, analogowych i złożonych, ponieważ system przekazuje między elementami urządzenia zarówno symbole (teksty) jak i prądy elektrycznego napięcia.

Wejściowymi danymi do systemu *Amethyst* są schematy obiektu (w przypadku strukturalnego obiektu) i ujednolicone opisy do każdego z komponentów obiektu. System produkuje i zapisuje do osobnego katalogu ujednolicony opis symulowanego obiektu. Dla niestrukturalnego obiektu, tj. najprostszego komponentu, którego brakuje w bibliotece, trzeba przygotowywać ujednolicony opis.

Ujednolicony opis dowolnego komponentu składa się z następujących części:

- tekstowego pliku z parametrami i wzorcem prezentacji komponentu (w postaci tekstu sterującego),
- plików programu w języku C++ do symulowania funkcjonowania komponentu.

Opis funkcji modelu obiektu musi być w języku C++ w postaci ujednoliconej klasy zapisanej do odrębnych plików: nagłówkowego (z rozszerzeniem „h") i definiującego (z rozszerzeniem „cpp").

Przed programowaniem funkcji komponentu zalecane jest opisanie modelu obiektu za pomocą aparatu matematycznego tzw. *F - modelu* („funkcjonalnego modelu") [1].

5. Struktura systemu

System *Amethyst* zawiera dwa podstawowe narzędzia programowe:

- generator komputerowych programów symulacyjnych,
- edytor okien prezentacji stanu i parametrów modelu obiektu.

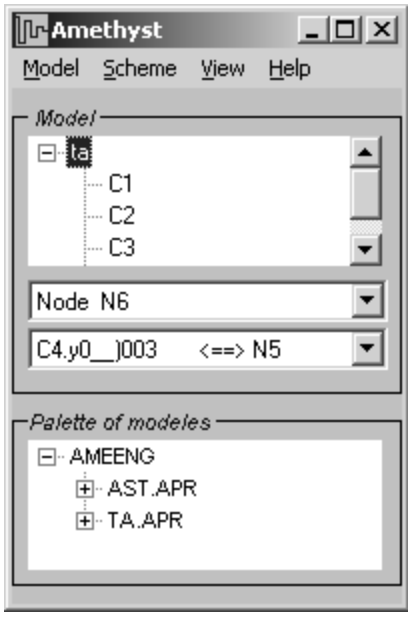
Generator programów (*amethyst.exe)* umożliwia budowanie modelu złożonego obiektu z prostszych komponentów bibliotecznych oraz definiowanie parametrów komponentów. W końcowym efekcie powstaje autonomiczny program zawierający monitor symulowania i rejestrator stanu badanego modelu obiektu. Tak wyprodukowany program (moduł wywoławczy) stanowi autonomiczny, niezależny produkt, który w sposób interaktywny umożliwia testowanie badanego obiektu.

Autor modeli komponentu pracujący nad ujednolicony opisem modelu może skorzystać ze specjalnego edytora okien prezentacji stanu i parametrów *(ameedit.exe).* Za pomocą edytora okien prezentacji, autor modelu może umieszczać w oknie grafikę, tekst, liczby, wartości z jednostkami czasu, prądu, napięcia itp. Takie możliwości odzwierciedlania cech obiektów zezwalają na symulowanie dowolnego obiektu technicznego.

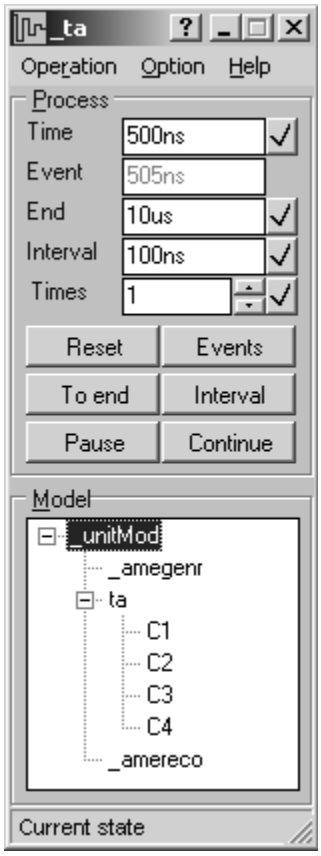
6. Główne okno systemu

Główne okno (rys. 1) systemu *Amethyst* służy do interaktywnego sterowania budowaniem modelu.

W oknie są opisane struktura modelu, dane o komponentach i połączeniach oraz paleta już zbudowanych modeli.



Rys. 1. Główne okno systemu *Amethyst*



Rys. 2. Panel sterowania

7. Sterowanie symulacją

Rys. 2 pokazuje panel sterowania symulacją, który system *Amethyst* dodaje do każdego interaktywnego programu.

Jak wynika z rys. 2, symulacja jest możliwa w następujących trybach:

- do końca czasowego okna symulowania (tryb *To end*),
- od zdarzenia do zdarzenia (tryb *Events*),
- w granicach czasowego przedziału (tryb *Interval*).

Możliwe jest wyzerowanie procesu (tryb *Reset*), powstrzymanie (tryb *Pause*) i przedłużenie (tryb *Continue*) procesu.

Czasowe parametry z klawiszami zaznaczonymi „ptaszkiem” można redagować bezpośrednio na panelu sterowania.

8. Wyświetlanie wyników

Wartości zmiennych wynikowych można pokazać w specjalnym oknie - oknie wizualnego rejestratora, który jest dodawany systemem do każdego modelu. List zmiennych i format rejestracji można ustawić przy symulacji albo opisać w tekstowym pliku z rozszerzeniem *„.arc”*.

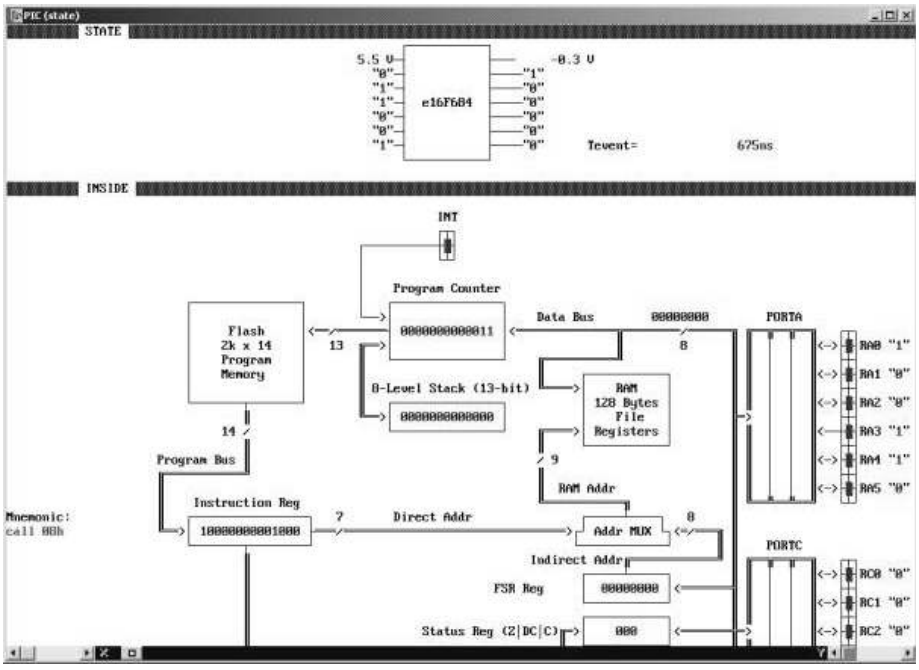
Można zaprojektować i dodać do systemu inny rejestrator lepiej pasujący do zagadnienia z dziedziny przedmiotowej.

Rozpatrzymy przykłady aplikacji wyprodukowanych systemem *Amethyst*.

9. Modeli mikrokontrolerów

Jako przykład aplikacji otrzymanej za pomocą *Amethysta* posłuży symulator mikrokontrolera *PIC16F684* firmy *Microchip*.

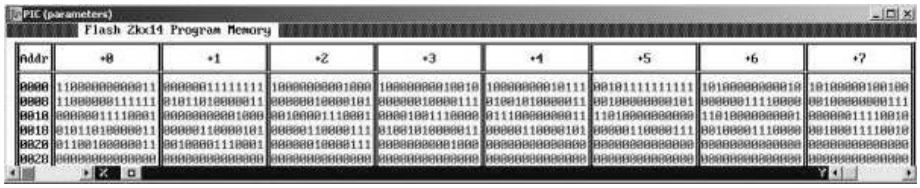
Jak już było wspomniane wygenerowany za pomocą systemu program pozwala obserwować stany komponentów urządzenia podlegającego symulacji oraz ustawiać ich poszczególne parametry. Dostęp do aktualnego, zmienianego dynamicznie przez aplikację, stanu mikrokontrolera *PIC16F684* uzyskujemy poprzez okno prezentacji stanu, którego fragment jest przedstawiony na rys. 3.



Rys. 3. Okno prezentacji stanu mikrokontrolera *PIC16F684*

Ametyst daje możliwość swobodnego decydowania o formacie informacji pokazywanych w oknie prezentacji. Mogą więc być one w pełni dostosowywane do potrzeb, jakie spełniać ma aplikacja i zależą jedynie do inwencji twórcy symulatora. Na powyższym rys. 3 widzimy dwa schematy: zewnętrzny schemat oraz fragment blokowego schematu mikrokontrolera pozwalające poznać jego budowę i zasady działania. Dodatkowo wartości podlegające dynamicznym zmianom, w celu ich czytelniejszego ukazania, wyświetlane są na kolorowo.

Do wyświetlania, jak również zmiany parametrów symulowanego układu przeznaczone jest okno prezentacji parametrów, którego fragment jest przedstawiony na rys. 4.



Rys. 4. Okno prezentacji parametrów mikrokontrolera *PIC16F684*

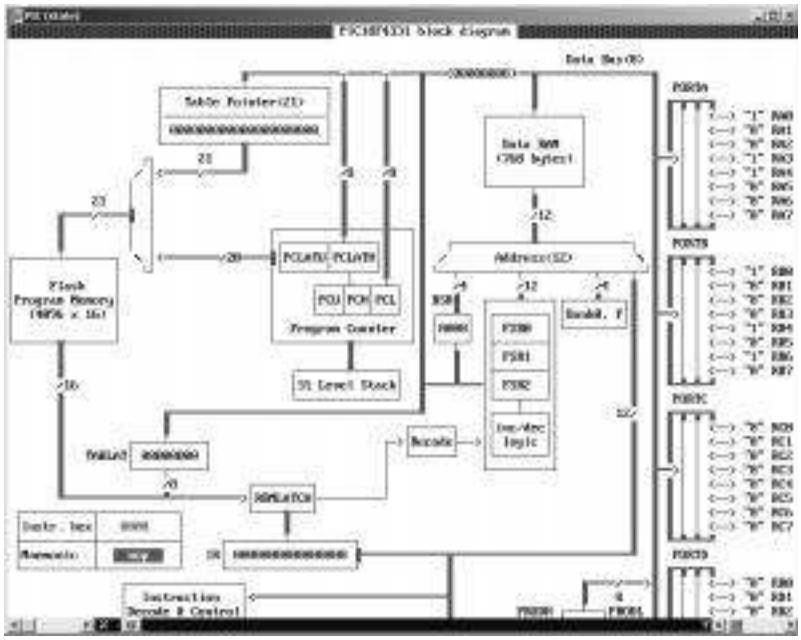
Symulator stworzony w systemie *Ametyst* pozwala poprzez okno prezentacji parametrów na łatwy dostęp i zmianę parametrów według jakich przebiegać ma symulacja urządzenia. W przypadku omawianego symulatora mikrokontrolera głównymi parametrami są instrukcje umieszczone w jego pamięci programu (widoczne na rys. 4. w postaci ciągu zer i jedynek). Parametry umieszczone w tym oknie można zmieniać w każdej chwili, także w trakcie przebiegu symulacji, a wynik przeprowadzenia modyfikacji natychmiast jest widoczny w oknie prezentacji stanu. Daje to możliwość zapoznania się ze wszystkimi skutkami zmiany parametrów oraz pozwala w pełni kontrolować przeprowadzanie symulacji urządzenia.

Drugim przykładem symulowanego urządzenia posłuży mikrokontroler *PIC18F4331* firmy *Microchip*, którego model zbudowano w systemie *Amethyst*. Jego zmieniający się w trakcie symulacji stan można obserwować w oknie prezentacji stanu, którego fragment jest przedstawiony na rys. 5.

Widoczne na rys. 5 pola oznaczone kolorem stanowią zmieniające się dynamicznie podczas symulacji elementy modelu. Pozwala to na zaobserwowanie stanu symulowanego urządzenia w każdej chwili, będącej najmniejszym odcinkiem czasu, w trakcie którego następuje zdarzenie w systemie. Dzięki odpowiednio dobranym parametrom modelu zgodnymi z charakterystykami rzeczywistego urządzenia prezentowane wyniki są dokładne i odzwierciedlają prawdziwe procesy zachodzące wewnątrz mikroukładu.

Fragmenty okna prezentacji parametrów mikrokontrolera *PIC18F4331* są przedstawione na rys. 6.

Do pamięci programu pokazanej na rys. 6,a można wprowadzać własne programy i testować poprawność ich wykonania.



Rys. 5. Okno prezentacji stanu mikrokontrolera *PIC18F4331*

(a)

PIC (parameters)

Flash Program Memory (409E

Address	01-00	03-02	05-04	07-06	09-08	0B-0A	0D-0C	0F-0E
0000h	5080	6F00	5081	6F01	5100	6301	EF14	F000
0020h	F000	5F01	EF08	F000	6A94	6E82	0000	0000
0040h	0000	0000	0000	0000	0000	0000	0000	0000
0060h	0000	0000	0000	0000	0000	0000	0000	0000
0080h	0000	0000	0000	0000	0000	0000	0000	0000
00A0h	0000	0000	0000	0000	0000	0000	0000	0000
00C0h	0000	0000	0000	0000	0000	0000	0000	0000
00E0h	0000	0000	0000	0000	0000	0000	0000	0000

(b)

PIC (parameters)

Generator

```
Generator: 1/2 period =    25ms
Delays: '0'=        5ms:  '1'=        10ms:
'0->1'_voltage =    1.4 V:
Input parameters:     (-3.85mA -1.5V)(-0.4mA 0.8V)(50mA 2V)(0.3mA 7V)
'0'_output parameters:(-8.5mA -1.5V)(0mA 0.2V)(16mA 0.4V)(17mA 7V)
'1'_output parameters:(-7.27mA -1.5V)(0A 3.5V)(0.1mA 7V)
```

Rys. 6. Fragmenty okna prezentacji parametrów mikrokontrolera *PIC18F4331*: pamięć programu (a) i parametry wbudowanego generatora (b)

Symulator wykonany w systemie *Amethyst* wykazuje znaczną przewagę nad zwykłymi wykrywaczami usterek (*debuggers*), które potrafią jedynie wykonywać instrukcje w trybie krokowym. Pozostałe elementy mikroukładu, na które wpływają wykonane rozkazy, pozostają niewidoczne dla programisty. Symulator natomiast daje pełną kontrolę poprzez możliwość obserwowania wszystkich skutków działania programu.

10. Zakończenie

Komputerowy system *Amethyst* może być wykorzystany do celu nie tylko poznawczego ale i dydaktycznego. Generator programów pozwala na bezpieczne nauczanie zasad projektowania, dlatego że każdy z wytworzonych programów może być jednym z wariantów rozwiązania pewnego problemu i jego funkcjonowanie może być szczegółowo sprawdzone i porównane z innymi wariantami.

Komputerowy system *Amethyst* pozwoli na kształtowanie i testowanie modeli różnych obiektów, stwarzanie odpowiednich warunków nauki w różnych okolicznościach, pozwoli na rozwijanie metod kształcenia, uwzględniając indywidualne potrzeby kształcących się i w pełni zaspakajając wymagania nauczycieli.

Musi być opracowana metoda wytwarzania komputerowych dydaktycznych programów do celów kształcenia i szkolenia zawodowego na zasadzie korzystania z systemu *Amethyst*. Metoda będzie zawierała kolejność i treść kroków do wytwarzania dydaktycznych programów. Uwaga musi być skoncentrowana na wytwarzaniu modeli bibliotecznych komponentów. Dzięki łatwości wytwarzania różnorodnych wariantów programów dydaktycznych metoda może koncentrować się w większym stopniu na indywidualnych potrzebach osób uczących się.

Dzięki systemowi *Amethyst* nauczyciele i wykładowcy będą mogli wykonać całokształt prac dydaktycznych w zakresie swojego przedmiotu, przeprowadzić całokształt programów symulujących wirtualne i rzeczywiste obiekty (sytuacje, instytucje, układy scalone oraz urządzenia zbudowane na bazie tych układów).

Bibliografia

1. **Timofeev Alexander O.**, 2004: *Computer production of programs for simulation of dynamic systems.* Proceedings of the 15th International Conference on Systems Design (7-10 September 2004, Wroclaw, Poland). Vol. 2. Wrocław, Oficyna Wydawnicza Politechniki Wrocławskiej, 2004. s. 91-95.