7 1989

informatyka

Felix Redmill
o niezawodności i bezpieczeństwie
Uruchamianie urządzeń mikroprogramowanych
Język symulacji na PC

WYDAWNICTWO NOT SIGMA

Cena 300 zł

JACEK KURAŚ
Centrum Podstawowych Problemów Gospodarki Surowcami Mineralnymi i Energią Polska Akademia Nauk Kraków

JACEK LEMBAS
MAREK SKOMOROWSKI
Katedra Informatyki Uniwersytet Jagielloński Kraków

Język symulacji systemów o działaniu ciągłym ze zdarzeniami dyskretnymi

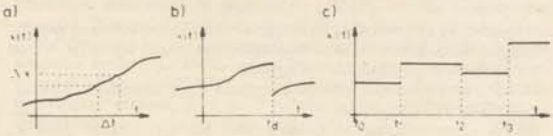
Jedną z podstawowych właściwości systemu, którą bada się w procesie symulacji, jest jego działanie, czyli zachowanie się zmiennych stanu w czasie. Codzienne odczucie czasu fizycznego nakazuje traktować go jako zmienną ciągłą o wartościach należących do zbioru liczb rzeczywistych. Takie traktowanie czasu jest słuszne dla makroskopowego ujęcia zjawisk w otaczającym nas świecie i przestaje być słuszne dopiero na poziomie zjawisk, którymi zajmuje się fizyka kwantowa. Istnieje jednak wiele systemów, których działanie ma taki charakter, że zasadnicze, istotne zmiany stanu zachodzą w nich jedynie w pewnych dyskretnych momentach czasu, między którymi wartości zmiennych stanu nie zmieniają się lub zmieniają się bardzo niewiele. Przy tworzeniu modeli takich systemów posługiwanie się ciągłą zmienną czasową wydaje się nieuzasadnione. Tak więc, z punktu widzenia działania, systemy możemy podzielić na dwie klasy:

- systemy o działaniu ciągłym, w których w każdym, dowolnie małym odcinku czasu zachodzą zmiany stanu mające istotny wpływ na dalsze zachowanie systemu, a ponadto – małym zmianom czasu odpowiadają małe zmiany stanu,
- systemy o działaniu dyskretnym, w których zmiany stanu, istotne z punktu widzenia celu badania systemu, zachodzą w dyskretnym (przeliczalnym) zbiorze chwil czasowych.

Na pograniczu tych dwu klas można wyróżnić jeszcze trzecią:

- systemy o działaniu ciągłym ze zdarzeniami dyskretnymi, w których zasadniczo ciągłe działanie systemu jest zaburzane w dyskretnych chwilach czasowych przez zjawiska powodujące skokowe zmiany stanu.

Ilustrację powyższego podziału stanowi rys. 1.



Rys. 1. Ilustracja zachowania pewnej zmiennej stanu x(t): a) w systemie o działaniu ciągłym, b) w systemie o działaniu ciągłym ze zdarzeniami dyskretnymi, c) w systemie o działaniu dyskretnym

Języki do symulacji komputerowej systemów o działaniu ciągłym bazowały początkowo na tradycji modelowania analogowego i z tego powodu nazwano je cyfrowymi symulatorami maszyn analogowych. Do tej grupy języków należą m.in. język 1130/CSMP [1] oraz GODYS–1 i 2 [2, 4]. W latach 1965–1967 powstał język CSSL (ang. *Continuous Systems Simulation Language*), uznany później za wzorcowy język do symulacji systemów o działaniu ciągłym. Autorzy tego języka odeszli w znacznym stopniu od tradycji modelowania analogowego, pozostawiając użytkownikowi znaczną swobodę w doborze formy zapisu równań modelu. W ogólności, forma ta ma postać równań podobnych syntaktycznie do instrukcji przypisania w językach programowania. Translator języka CSSL–3, będącego pewną wersją języka CSSL w systemie Cyber–72, został skonstruowany jako makroprocesor. Tłumaczy on program w języku CSSL–3 na program w języku Fortran, który jest następnie kompilowany i ładowany do pamięci z użyciem tradycyjnych środków. Zaletą takiego rozwiązania jest możliwość stosowania makrodefinicji i różnego rodzaju bibliotek. Rozwiązanie to charakteryzuje się jednak bardzo długim czasem translacji. Ponadto diagnostyka błędów na etapie wykonania jest szczególnie trudna, ponieważ standardowe procedury diagnostyczne odwołują się do programu pośredniego zapisanego w Fortranie, a nie do programu źródłowego napisanego przez użytkownika w języku CSSL–3.

Językami zbliżonymi zewnętrznie do CSSL–3 są języki GODYS–5 [5, 6] i GODYS–6 [8] zaimplementowane na komputerach serii Odra i Riad oraz przedstawiony w niniejszym opracowaniu język GODYS–PC, zrealizowany na mikrokomputery kompatybilne z IBM PC/XT/AT. Języki GODYS–5 i 6 jak również język GODYS–PC zostały zaprojektowane w taki sposób, że nie posiadają wymienionych wyżej wad języka CSSL–3.

Popularność i dostępność mikrokomputerów zgodnych z IBM PC spowodowała, że wiele obliczeń wykonywanych dotychczas na dużych komputerach realizuje się na mikrokomputerach. W bogatej ofercie oprogramowania mikrokomputerów IBM PC odczuwa się brak języków służących do symulacji zarówno procesów o działaniu ciągłym, jak i dyskretnym. Spotyka się rzadko zarówno języki przeniesione wprost z dużych komputerów (np. Dynamo), jak też uwzględniające specyfikę pracy z mikrokomputerem (np. ISIM firmy Simulation Sciences).

Należy nadmienić, że procesy symulacyjne są procesami czasochłonnymi i symulacja wielkich złożonych modeli na mikrokomputerach 16-bitowych może okazać się nieopłacalna lub wprost niemożliwa. Symulacja małych i średnich modeli z wykorzystaniem mikrokomputera jest możliwa, a częstokroć bardziej elastyczna niż w wypadku dużych komputerów.

Badania symulacyjne przeprowadzone porównawczo na identycznych modelach na komputerze Riad–32 (GODYS–6) i na mikrokomputerze zgodnym z IBM PC/AT z zegarem 10 MHz (GODYS–PC) wykazały 15% wzrost szybkości obliczeń na korzyść mikrokomputera. Powyższe uwagi uzasadniają celowość wprowadzenia języka GODYS–PC do oprogramowania mikrokomputerów kompatybilnych z IBM PC.

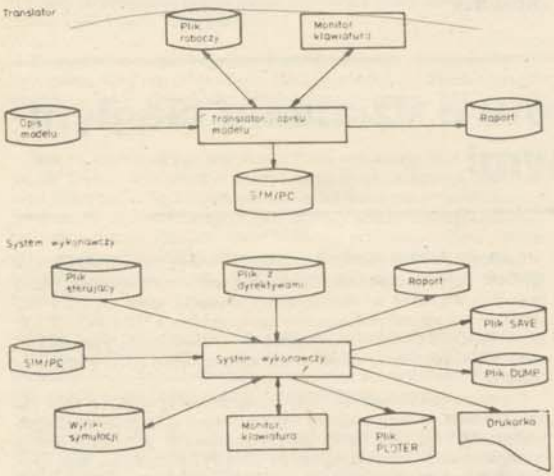
ZAŁOŻENIA I MOŻLIWOŚCI JĘZYKA GODYS–PC

Modelem matematycznym systemu o działaniu ciągłym jest, w najprostszym przypadku, układ równań różniczkowych zwyczajnych. Jeżeli układ ten jest podany w postaci kanonicznej, to nadaje się bezpośrednio do całkowania numerycznego. W praktyce trudno jednak żądać od użytkownika sprowadzania jego modelu do postaci kanonicznej, ponieważ ta forma może być dla niego niedogodna i nienaturalna. Język GODYS–PC pozwala na zapisanie modelu użytkownika w formie równań o praktycznie dowolnej postaci. Numeryczne metody całkowania równań różniczkowych zwyczajnych wymagają jednak przeprowadzania obliczeń w pewnej standardowej kolejności (tzw. scentralizowane całkowanie). Powstaje więc problem ustalenia tej kolejności w modelu użytkownika. Zadanie to musi wykonać translator języka symulacyjnego. Do ustalenia prawidłowej kolejności obliczeń w języku GODYS–PC wykorzystano formalną metodę, opartą na grafach funkcyjnych, przedstawioną w [3].

Podstawą przygotowania programu w języku GODYS–PC jest zapis modelu matematycznego badanego systemu w postaci równań różniczkowo-całkowo-logicznych. Zasadniczą częścią programu jest opis tego modelu, odwzorowany w procesie translacji na model cyfrowy. Na modelu tym można dokonywać wielu eksperymentów symulacyjnych. Instrukcje opisu modelu mogą być pisane w dowolnej kolejności, ponieważ translator języka GODYS–PC dokonuje ich automatycznego sortowania zapewniając właściwą kolejność obliczeń. Algorytm wyznaczania sekwencji obliczalnej (tj. określenia kolejności wykonywania instrukcji) został przedstawiony w [7].

Do całkowania równań modelu można wykorzystać jedną z sześciu dostępnych w języku metod całkowania: pięć metod stałokrokowych

i jedną zmiennokrokową. Wyniki eksperymentów mogą być prezentowane w postaci odpowiednich tabel i wykresów. Procesor języka GODYS–PC stanowią dwa programy: translator języka opisu modelu i system wykonawczy (rys. 2).



Rys. 2. Struktura systemu GODYS-PC

Procesor ten można wykorzystywać zarówno w tradycyjnym trybie wsadowym, jak i w trybie konwersacyjnym. Kompilator języka opisu modelu analizuje program wcześniej zapisany w całości na pliku. Nie ma więc znaczenia, w jaki sposób jest wprowadzany jego tekst. Różnica jakościowa występuje przy wsadowym i konwersacyjnym trybie wykorzystania systemu wykonawczego. W trybie wsadowym należy przewidzieć (przynajmniej w zarysie) wyniki poszczególnych dyrektyw, aby następne dyrektywy były sensowne. Tryb konwersacyjny daje możliwość bezpośrednich zmian w doborze kolejnych dyrektyw symulacji w zależności od dotychczasowych wyników. Jest to szczególnie ważne przy przypadkowym dobieraniu aktualnych parametrów kolejnego eksperymentu symulacyjnego. Można również prześledzić na monitorze kilka wersji dyrektywy wydawniczej, a do składania i wydruku wybrać tę najlepszą. Trzeba ponadto zaznaczyć, że dyrektywy te korzystają z procedur graficznych, czego brak było w poprzednich wersjach języka GODYS, a co znacznie polepsza jakość systemu. Sposób użytkowania systemu wykonawczego w trybie konwersacyjnym jest prosty, zaopatrzony w podstawowe mechanizmy przerwań i nie wymaga dodatkowej instrukcji.

STRUKTURA PROGRAMU W JĘZYKU GODYS–PC

Program w języku GODYS–PC składa się z dwu zasadniczych części:
– zapisu modelu matematycznego systemu,
– sekwencji dyrektyw sterujących eksperymentami symulacyjnymi na modelu i wyprowadzaniem wyników.

W procesie translacji zapis modelu symulowanego systemu jest przekształcany na równoważny program w języku maszyny abstrakcyjnej SIM/PC (rys. 2), implementowanej przez efektywny interpreter, którego pracą sterują wymienione wyżej dyrektywy.

Program w języku GODYS–PC ma następującą strukturę:

```
MODEL nazwa-modelu
    sekcja deklaracji (PARAMT, CONST, PREPAR, DECLARE,
                       MAP, NOMAP, RENAME)
INITIAL
    sekcja inicjująca (opcjonalnie)
END
DYNAMIC
    instrukcje opisu modelu
END
LOAD nazwa-modelu
    ciąg dyrektyw (DATA, EXECUTE, CONTINUE, HDR, PRINT,
                   PRPLOT, GRAPH, PLOTXY, DUMP, SAVE)
FINISH
```

Dyrektywa MODEL podaje symboliczną nazwę modelu. Po dyrektywie MODEL następuje sekcja deklaracji, w której mogą występować następujące deklaracje:

PARAMT –deklaruje identyfikatory parametrów modelu. Parametrom tym, w trakcie kolejnych eksperymentów, można nadawać wartości (za pomocą dyrektywy DATA), odwołując się do nich przez nazwę. Za pomocą dyrektywy PARAMT deklarowane są parametry proste i tablice jednowskaźnikowe.

CONST – definiuje identyfikatory stałych liczbowych modelu.

PREPAR – deklaruje zmienne obserwowane modelu, tzn. zmienne rejestrowane w trakcie eksperymentów symulacyjnych. Zmienna niezależna, o standardowym identyfikatorze T, jest rejestrowana automatycznie.

DECLARE – określa liczbę argumentów i parametrów funkcji zdefiniowanej przez użytkownika.

MAP – deklaracja ta umożliwia wydruk pełnej mapy modelu (standardowo wyprowadzana jest tylko mapa identyfikatorów).

NOMAP – deklaracja ta wstrzymuje wyprowadzanie mapy.

RENAME – deklaracja ta zmienia identyfikator zmiennej niezależnej modelu (standardowo T). Na przykład, użycie RENAME T = CZAS powoduje, że nazwą zmiennej niezależnej będzie CZAS, a identyfikator T może być użyty do oznaczenia innej zmiennej.

Sekcja inicjująca ujęta w ograniczniki INITIAL – END powoduje nadanie wartości parametrom wewnętrznym modelu. Identyfikatory występujące w sekcji inicjującej po lewej stronie znaku przypisania są identyfikatorami parametrów wewnętrznych. Parametrom tym przypisywane są odpowiednie wartości przed rozpoczęciem eksperymentu symulacyjnego.

Właściwy opis modelu tworzy sekwencja instrukcji opisu modelu ujęta w ograniczniki DYNAMIC – END. Instrukcja opisu modelu ma postać standardowej instrukcji przypisania, przy czym zmienna występująca po lewej stronie znaku przypisania może wystąpić w tym miejscu tylko jeden raz. Wyrażenie występujące po prawej stronie znaku przypisania jest konstruowane przy użyciu operatorów arytmetycznych, relacyjnych, logicznych i funkcji standardowych języka. Funkcje standardowe języka mają ściśle określone zbiory argumentów i parametrów. Parametrami aktualnymi funkcji standardowych mogą być jedynie wielkości stałe w trakcie danego eksperymentu symulacyjnego (stałe, nazwy stałych, parametry wewnętrzne, parametry proste i tablice). Język GODYS–PC ma 75 operatorów i funkcji standardowych. Użytkownik może w bardzo prosty sposób rozszerzyć język GODYS–PC o własne funkcje napisane w języku Fortran–77.

Dyrektywy systemu wykonawczego sterują eksperymentami symulacyjnymi na modelu, a także sterują wyprowadzaniem wyników. Wśród tych dyrektyw wyróżnia się następujące:

LOAD – powoduje załadowanie programu w języku maszyny abstrakcyjnej SIM/PC do pamięci.

DATA – pozwala na przypisanie wartości parametrom prostym, a tablicom zbiorów wartości. Wartości te mogą być zmienione tylko kolejną dyrektywą DATA, lub w trakcie procesu optymalizacji parametrycznej.

EXECUTE, CONTINUE – powodują wykonanie pojedyńczego eksperymentu symulacyjnego na modelu o określonych aktualnie parametrach. Eksperyment jest autonomiczny w wypadku EXECUTE lub jest kontynuacją eksperymentu poprzedniego w wypadku CONTINUE. Sposób prowadzenia eksperymentu symulacyjnego determinują parametry wykonania. Parametry te określają odpowiednio:

TMIN, TMAX – modelowy czas początku i końca symulacji,
METHOD – algorytm całkowania numerycznego,
DT – krok całkowania numerycznego,
ABSERR, RELERR – tolerancję dla bezwzględnego i względnego błędu całkowania w metodzie zmiennokrokowej,
COMDEL – przyrost zmiennej niezależnej (okres komunikacyjny), determinujący częstotliwość zapisywania wartości zmiennych obserwowanych na plik roboczy,
BEGTR, ENDTR – modelowy czas początku i końca śledzenia zmiennych modelu,
CLKTIME – limit rzeczywistego czasu trwania eksperymentu symulacyjnego (w sekundach),

OPT = (lista) – optymalizacja parametryczna (dozwolona tylko w dyrektywie EXECUTE); lista parametrów optymalizacji zawiera identyfikator funkcji celu jako przyjęty wskaźnik jakości, identyfikatory optymalizowanych parametrów (maksymalnie 8) wraz z zakresem ich występo-

wania oraz informacje o niestandardowej obsłudze zakończenia optymalizacji. Wystąpienie błędu wykonania w danym eksperymencie powoduje jego zakończenie. Kolejne eksperymenty mogą być realizowane tylko w wypadku podania parametru wymuszającego wykonanie FORCE. W wypadku niepodania wszystkich lub tylko pewnych parametrów w dyrektywach EXECUTE lub CONTINUE przypisywane są im wartości standardowe lub zastępcze, implikowane przez inne, podane wcześniej przez użytkownika.

Wygodnym elementem języka są dyrektywy wydawnicze. Umożliwiają one prezentowanie na monitorze, drukarce, pliku tekstowym lub ploterze zmiennych obserwowanych modelu, rejestrowanych w trakcie eksperymentu symulacyjnego, w postaci:
PRINT – tablica wartości,
PRPLOT – wykresy (do 4 zmiennych równocześnie),
GRAPH – tablica wartości i wykres dla jednej zmiennej,
PLOTXY – wykres zależności dwóch zmiennych.
Przy wykonywaniu wykresów wykorzystywana jest skala określana automatycznie lub też ustalana na podstawie podanych przez użytkownika wartości granicznych danej zmiennej. Inne dyrektywy są następujące:
HDR – podaje nagłówek,
SAVE – powoduje składowanie wyników symulacji (wartości zmiennych obserwowanych) na pliku, który może być później wykorzystany przez inne programy do obliczeń postsymulacyjnych.
DUMP – powoduje kopiowanie na plik zawartości pamięci maszyny abstrakcyjnej SIM/PC.
FINISH – powoduje zakończenie programu.

Język GODYS–PC umożliwia pracę zarówno w tradycyjnym trybie wsadowym, co jest szczególnie pożyteczne w wypadku dłuższych obliczeń, jak również w trybie konwersacyjnym, pozwalającym na bieżące śledzenie procesu symulacyjnego i korygowanie parametrów na podstawie obserwacji.

PRZYKŁADOWY PROGRAM

Jako przykład programu w języku GODYS–PC rozważmy następujący model sterowania zapasami [6]. Detalista utrzymuje zwykle pewien poziom zapasów sprzedawanych towarów, który ustala jako kompromis między kosztem składowania a stratami, jakie mogłyby być poniesione wskutek braku towaru. Różnica między żądanym poziomem zapasów (ZPZ) a bieżącym poziomem zapasów (BPZ) jest wykorzystywana do sterowania (system ze sprzężeniem zwrotnym) natężeniem zamówień składanych u producenta (NZSP) w celu uzupełnienia zapasów. Bieżące natężenie produkcji (BNP) jest proporcjonalne do liczby zamówień nie zrealizowanych przez producenta (ZNPP). Bieżące natężenie sprzedaży (BNS), będące zmienną zewnętrzną modelu (wymuszeniem) ma postać BNS = $f(t)$, gdzie $f(t)$ jest dowolną funkcją spełniającą warunek $f(t) \geqslant 0$ (warunek ten wynika z fizycznej interpretacji bieżącego natężenia sprzedaży.

W rozważanym modelu zmiennymi stanu są następujące zmienne: bieżący poziom zapasów (BPZ), zamówienia nie zrealizowane przez producenta (ZNPP) i średnie natężenie sprzedaży (*SNS*), dla których równania stanu przyjmują postać:

$$\frac{d(\mathrm{BPZ})}{dt} = \mathrm{BNP} - \mathrm{BNS}$$

$$\frac{d(\mathrm{ZNPP})}{dt} = \mathrm{NZSP} - \mathrm{BNP}$$

$$\frac{d(\mathrm{SNS})}{dt} = \frac{\mathrm{BNS} - \mathrm{SNS}}{T2}$$

T2 reprezentuje jednostkę czasu, w jakiej następuje uśrednianie sprzedaży. Program w języku GODYS-PC dla opisanego modelu przedstawiono na rys. 3.

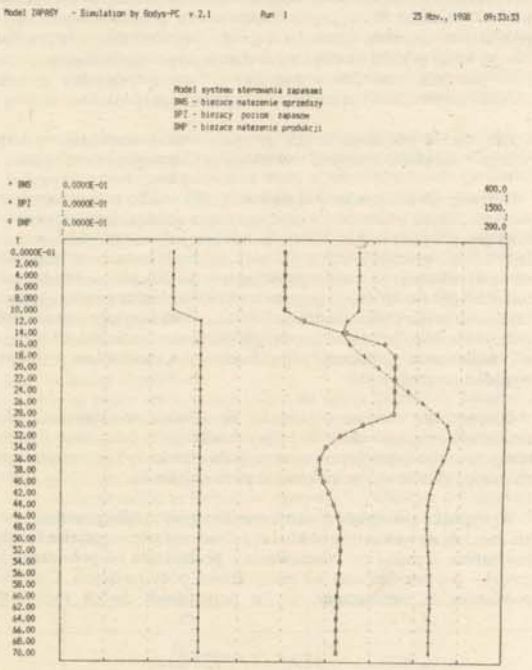
W rozważanym modelu przyjęto pewne ograniczenia. Bieżący poziom zapasów (BPZ) nie może przyjmować wartości ujemnych. Wynika to z fizycznej interpretacji zapasów. Przyjęto również, że bieżące natężenie produkcji (BNP) może wzrastać tylko do pewnej wartości maksymalnej. Ograniczenia te zostały zrealizowane za pomocą funkcji standardowych języka GODYS-PC (MAX, BOUND). W przedstawionym programie bieżące natężenie sprzedaży (BNS) ma postać funkcji

```
model zapasy
nosap
 prepar bns, bpz, bnp
 param t1, t2, t3, k1, k2, k3
 param bpz0, znpp0, sns0, r, a1, a2, t0, b
dynamic
   z = integ(bnp-bns;bpz0)
   bpz = max(z,0)
   znpp = integ(nzsp-bnp;znpp0)
   sns = integ((bns-sns)/t2;sns0)
   bnp = k3*bound(znpp;1,b)/t1
   zpz = r*sns
   nzsp = k1*sns+k2*(zpz-bpz)/t3
   bns = a1*step(t)+a2*step(t-t0)
end

load zapasy
data t1=4,t2=4,t3=4,k1=1,k2=1,k3=1,bpz0=1000,znpp0=400,
     sns0=100,r=10,a1=100,a2=25,t0=10,b=600
hdr "              Model systemu sterowania zapasami"
hdr "              BNS - biezace natezenie sprzedazy"
hdr "              BPZ - biezacy poziom zapasow"
hdr "              BNP - biezace natezenie produkcji"
execute(dt=0.2,tmax=70,conde)*2,method=trapez)
prplot(bns*(0,400),bpz*(0,1500),bnp*(0,200))
finish
```

Rys. 3. Program symulacyjny w języku GODYS-PC dla modelu systemu sterowania zapasami

skoku jednostkowego (funkcja standardowa STEP). Wyniki eksperymentu symulacyjnego dla przedstawionego modelu systemu sterowania zapasami zostały przedstawione na rys. 4.



Rys. 4. Wyniki eksperymentu symulacyjnego dla modelu systemu sterowania zapasami; realizacja dyrektywy PRPLOT z rys. 3

* * *

Język GODYS-PC jest eksploatowany od jesieni 1987 r. zarówno do celów badawczych, jak i dydaktycznych. Dotychczasowe doświadczenia uzyskane w trakcie eksploatacji języków GODYS-5 i GODYS-6, a także języka GODYS-PC pozwalają twierdzić, że GODYS-PC jest prostym i wygodnym narzędziem do symulacji szerokiej klasy systemów dynamicznych o działaniu ciągłym, jak również systemów o działaniu ciągłym ze zdarzeniami dyskretnymi. Szczególnym walorem odróżniającym go od poprzedników są wszechstronne możliwości wydawnicze, a także możliwość śledzenia na bieżąco eksperymentów symulacyjnych przez użytkownika i reagowanie w wypadku podania błędnych parametrów (zerwanie symulacji, zakończenie zbyt długiego procesu wydawniczego).

dokończenie na s. 3

Zalecenia uzasadniają, że dokładna ocena powinna obejmować całe przedsięwzięcie, po pierwsze, aby możliwie najwcześniej wykryć zaistniałe problemy, a po drugie, ze względu na ograniczenia testowania i niedoskonałość metod wytwarzania oprogramowania. W celu umożliwienia dokładnej oceny, podzielono ją na cztery fazy: propozycja przedsięwzięcia, opracowanie, przyjęcie i zatwierdzenie, eksploatacja i konserwacja. Podczas oceny powstaje model będący podstawą przyjęcia (lub odrzucenia) systemu, zwany modelem bezpieczeństwa (ang. *safety case*).

W zaleceniach podkreśla się znaczenie zdefiniowania kryteriów odbioru ocenianego systemu. Te kryteria dotyczą całego okresu istnienia systemu i dotyczą stosowanych norm, kwalifikacji personelu, procedur zapewnienia jakości i wyników testów. Analizy kryteriów odbiorczych dokonuje się na dwóch poziomach – na poziomie środowiska i samego systemu komputerowego. W zaleceniach wyjaśniono rolę obu poziomów oraz omówiono czynniki istotne w ustalaniu kryteriów. Szczegółowo przedstawiono techniki oceny oraz model bezpieczeństwa. W zakończeniu, podano dokładne wskazówki dotyczące dokumentowania oceny.

Kwestionariusz dzieli się na pięć części odpowiadających poszczególnym fazom okresu istnienia systemu oraz trzy inne części dotyczące planowania i zarządzania, zatwierdzania, eksploatacji i konserwacji. Każda z nich dzieli się z kolei na rozdziały. Pytania w poszczególnych rozdziałach zawierają listy czynności dla osób oceniających i wykonawców, dotyczące odpowiednich części okresu istnienia projektu. Taki kwestionariusz jest bardzo dobrym narzędziem, ponieważ wykorzystując wskazówki z głównej części zaleceń osoba oceniająca może użyć go jako podstawy do planu oceny.

Konserwacja systemów komputerowych z uwzględnieniem bezpieczeństwa

W przeszłości, niewłaściwe specyfikowanie i złe metody projektowania i zarządzania przedsięwzięciami prowadziły do powstawania systemów, które wymagały poprawek, natychmiast po wprowadzeniu do eksploatacji, a później – znacznego wysiłku związanego z konserwacją. Jednocześnie, niewłaściwe zasady i procedury konserwacji powodowały pogorszenie jakości dokumentacji i zwiększenie złożoności systemu. Z tego względu, istotne jest stosowanie w konserwacji ścisłych zasad inżynierskich i procedur formalnych.

W zaleceniach podkreśla się, że każda proponowana zmiana w eksploatowanym systemie musi być najpierw poddana dokładnej analizie, a następnie zostać zaakceptowana na odpowiednim poziomie zarządzania. Jeżeli zmiana ma być zrealizowana, to należy ją traktować jako niezależne przedsięwzięcie i stosować sprawdzone metody projektowania i wytwarzania. Na zakończenie należy dokonać ponownej atestacji systemu i zaktualizować jego dokumentację.

Zalecenia oparto na użyciu tzw. diagramów pęcherzykowych (ang. *bubble diagrams*), które wyjaśniają istotę procesu konserwacji. Ilustrują cztery rodzaje konserwacji: zapobiegawczą, korekcyjną, doskonalącą i adaptacyjną (ang. *preventive, corrective, perfective, adaptive*). Każdy rodzaj konserwacji prowadzi jednak do zmiany w systemie i dlatego w stosowaniu wymaga tych samych zasad i kontroli.

Ponadto, zalecenia zawierają obszerny słownik wyjaśniający znaczenie wszystkich nazw użytych w diagramach oraz tzw. minispecyfikacje stanowiące opisy wszystkich czynności konserwacyjnych. Na opis czynności składają się wskazówki dotyczące sposobu zarządzania i szczegółowe procedury postępowania. Choć treść zaleceń dotyczy głównie systemów z uwzględnieniem bezpieczeństwa, co znalazło odbicie w stosowanej terminologii, podane reguły mogą być stosowane szerzej, a dokument można łatwo dostosować do potrzeb konserwacji innych rodzajów systemów w zastosowaniach krytycznych.

* * *

Dokonany powyżej przegląd czterech najnowszych zaleceń dotyczących odpowiednio pomiarów oprogramowania oraz projektowania, oceny i konserwacji systemów komputerowych z uwzględnieniem bezpieczeństwa może posłużyć jako wstęp do bardziej szczegółowej lektury odpowiednich dokumentów. Zainteresowani podstawami wytwarzania systemów komputerowych do zastosowań krytycznych mogą sięgnąć do zbioru pierwszych sześciu zaleceń. Wszystkie zalecenia opracował Komitet Techniczny nr 7 organizacji *European Workshop on Industrial Computer Systems*.

Tłumaczył i opracował
JANUSZ ZALEWSKI

LITERATURA

[1] Daniels B. K.: Proc. Safety and Reliability Society Symposium, Altrincham (UK), 10–12 November 1987. Elsevier, London, 1987
[2] Ehrenberger W.: Safety Related Computers in an Expanding Market. Proc. SAFECOMP'88. Pergamon Press, London, 1988
[3] International Electrotechnical Commission: Software for Computers in the Safety Systems of Nuclear Power Stations. IEC Publication 880, Geneva, 1986
[4] Rata J. M. A.: The Work of the Technical Committee on Reliability, Safety and Security of Industrial Computer Systems European Workshop on Industrial Computer Systems [1]
[5] Redmill F. J. (Ed): Dependability of Critical Computer Systems I. Elsevier, London, 1988
[6] Redmill F. J. (Ed.): Dependability of Critical Computer Systems II. Elsevier, London, 1989
[7] Zalewski J., Ehrenberger W.: Hardware and Software for Real Time Process Control. Proc. IFIP/IFAC/EWICS Working Conf., Warszawa, 30 May – 1 June 1988, North-Holland, Amsterdam, 1989.

Język symulacji systemów...

dokończenie ze s. 11

Planowana jest dalsza rozbudowa języka GODYS-PC. W kolejnej wersji przewiduje się wygodne w obsłudze menu i możliwość współpracy z myszą. Na życzenie użytkownika możliwe jest wygenerowanie wersji ze zmienionym zestawem funkcji standardowych.

LITERATURA

[1] Gordon G.: Symulacja systemów. WNT, Warszawa, 1974
[2] Jakubowski R., Król J.: Implementation of the simulation language based on functional graphs. Podstawy Sterowania, T. 2, 1972
[3] Jakubowski R., Król J.: Grafy funkcyjne w zastosowaniu do opisu i symulacji złożonych systemów dynamicznych. Podstawy Sterowania, T. 3, 1973
[4] Król J.: Symulacja cyfrowa złożonych systemów elektromechanicznych. Rozprawa doktorska. Politechnika Śląska, Gliwice, 1974
[5] Król J., Kuraś J., Lembas J.: Język symulacyjny dla komputerów ODRA 1300. Informatyka, nr 11, 1979
[6] Pofelski L., Skomorowski M.: Symulacja komputerowa. Uniwersytet Jagielloński. Kraków, 1983
[7] Król J., Kuraś J., Lembas J., Ślusarek M.: Implementacja języka do symulacji układów ciągłych ze zdarzeniami dyskretnymi. Podstawy Sterowania, T. 10, 1980
[8] Król J., Kuraś J., Lembas J., Rosek J.: Implementacja języka GODYS-6 dla komputera Riad-32. Raport roboczy. Instytut Informatyki, Uniwersytet Jagielloński, Kraków, 1985.

I Krajowa Konferencja Naukowa pn.

INŻYNIERIA WIEDZY i SYSTEMY EKSPERTOWE

odbędzie się we Wrocławiu w dniach 6–8 czerwca 1990 r.
Organizowana jest przez Instytut Sterowania i Techniki Systemów Politechniki Wrocławskiej i Komitet Automatyki PAN, przy współudziale Podkomitetu Automatyki PKPAiR NOT oraz Komisji Informatyki i Automatyki PAN Oddział Wrocław.

Tematyka konferencji:
- problemy i metody sztucznej inteligencji,
- metodologiczne podstawy inżynierii wiedzy,
- metody statystyczne w inżynierii wiedzy i systemach ekspertowych,
- metody i komputeryzacja badań eksperymentalnych,
- podstawy budowy systemów ekspertowych,
- ekspertowe systemy sterowania, identyfikacji i rozpoznania,
- zastosowania techniczne i biomedyczne.

Termin nadsyłania zgłoszeń oraz streszczeń referatów (o obj. 2 s. maszynopisu) upływa 30 listopada 1989 roku.

Adres organizatorów:
Dr Adam Grzech
Instytut Sterowania i Techniki Systemów Politechniki Wrocławskiej
ul. Janiszewskiego 11/17, 50–370 Wrocław