ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1997
Seria: INFORMATYKA Z. 33 Nr kol. . 1377

Marek MERES
Politechnika Śląska, Instytut Informatyki

WYKORZYSTANIE OBIEKTOWEJ TECHNIKI PROGRAMOWANIA W PAKIECIE AMOK

Streszczenie. W artykule przedstawiono skrótowy opis pakietu analizy modeli kolejkowych AMOK w wersji dla stacji roboczych SUN. Zwrócono uwagę na korzyści wynikłe z użycia obiektowej techniki programowania do utworzenia modelu sieci i na sposób integracji części obiektowej z przejętą z poprzednich wersji częścią proceduralną, napisaną w języku Fortran.

UTILISATION OF OBJECT-ORIENTED PROGRAMMING TECHNIQUE IN AMOK

Summary. A short description of the queueing model analysis package AMOK dedicated for SUN workstations is presented in the paper. The advantages of using OOP techniques in network model creation phase are mentioned as well as integration of new object-oriented design with old procedural part written in Fortran.

1. Teoria kolejek

Modele kolejkowe służą do opisu i analizy różnych układów, które teoria kolejek traktuje jako sieci stanowisk obsługi [3]. Podstawowym elementem modelu jest stanowisko obsługi. Obsługuje ono nadchodzących klientów w pewnym określonym czasie, nazywanym czasem obsługi. Jeśli klient nie może być obsłużony natychmiast, bo stanowisko jest zajęte, ustawia się w kolejce, skąd przechodzi do stanowiska po jego zwolnieniu. Klienci mogą być generowani przez źródła klientów i opuszczać sieć po pewnym czasie (sieć otwarta), lub też mogą krążyć w sieci (sieć zamknięta). Po opuszczeniu jednego stanowiska, klienci mogą przechodzić do innych z określonym prawdopodobieństwem przejścia. Ze względu na różne

typy regulaminów obsługi, oraz różne prawdopodobieństwa przejścia, klienci mogą być podzieleni na klasy.

Prócz wyżej wymienionych, w sieci mogą występować także elementy synchronizujące (np. semafory), rozgałęziające i łączące klientów (np. konstrukcje 'fork-join'), służące do gospodarki zasobami i inne.

Podstawowymi parametrami sieci są średni czas obsługi klienta, regulamin kolejki i średni czas generacji nowego klienta przez źródło. Parametry te są charakterystyczne dla danej klasy klientów.

Szukanymi wartościami w takim modelu są przede wszystkim średnie długości kolejek, czasy oczekiwania na obsługę w poszczególnych stanowiskach, stopień ich wykorzystania, średni czas pobytu klienta w sieci otwartej lub średni czas obiegu klienta w sieci zamkniętej.

Istnieje wiele metod analizy sieci różniących się ze względu na możliwości modelowania sieci zamkniętych/otwartych, różną dokładność i złożoność obliczeń oraz dozwolony zakres parametrów układu. Wiele z tych metod jest zbyt złożonych obliczeniowo, by można było stosować je do analizy dużych, kompleksowych sieci, stąd potrzeba wspomagania tego procesu odpowiednim oprogramowaniem. Zastosowanie takiego oprogramowania umożliwia także symulację sieci, co pozwala określić jej parametry niezależnie od metod analitycznych.

2. Wcześniejsze wersje pakietu AMOK

Pakiet programowy AMOK (Analizator MOdeli Kolejkowych) [2, 8] jest narzędziem wspomagającym tworzenie i analizę modeli kolejkowych. Inne tego typu pakiety, to np.: RESQ (opracowany w firmie IBM), QNAP (opracowany w laboratoriach INRIA przy współpracy firmy Honeywell-Bull) i BEST autorstwa firmy BGS Systems. AMOK był pierwszym i w swoim czasie jedynym tego typu pakietem w krajach dawnego bloku wschodniego.

Pierwsza wersja pakietu powstała w połowie lat osiemdziesiątych z przeznaczeniem na komputer ODRA 1305 z systemem operacyjnym GEORGE 3. Została napisana w standardzie języka Fortran, z zastosowaniem koncepcji modułowej, aby zapewnić elastyczność i uniwersalność pakietu, przy jednoczesnym zachowaniu ograniczonych rozmiarów plików wykonywalnych. Podstawowymi elementami pakietu AMOK były wtedy:

- program CONV, kompilator języka opisu modelu sieci;
- programy obliczeniowe, różne dla różnych typów sieci i metod analizy;
- program prezentacji wyników;

Zadaniem programu CONV było tłumaczenie opisu sieci stanowisk obsługi (napisanego w specjalnie do tego celu opracowanym języku opisu modelu QNDEL) na postać binarną i utworzenie bazy danych (pliku dyskowego), zawierającej wspomniany opis sieci w postaci binarnej. W takiej postaci model był analizowany przez programy obliczeniowe, a wyniki były wyprowadzane na standardowe urządzenie wyjściowe. Zaimplementowanie wielu metod umożliwiło rozwiązywanie praktycznie dowolnych modeli sieci.

Całość struktury pakietu AMOK w jego najwcześniejszej wersji przedstawia rys.1.

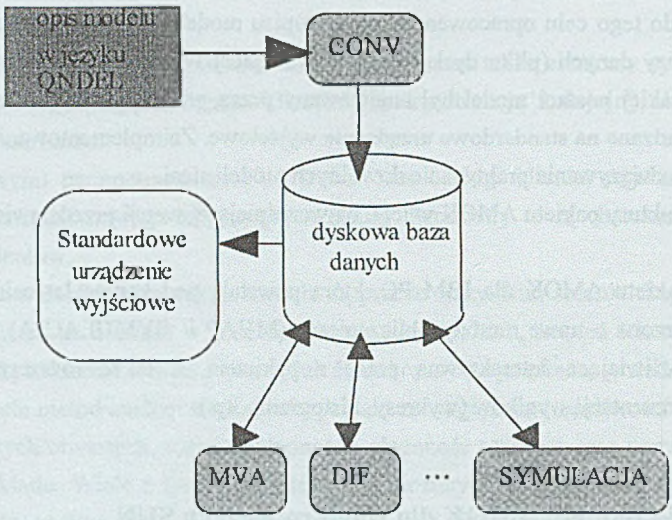
Wersja pakietu AMOK dla IBM PC, która powstała pod koniec lat osiemdziesiątych, została poszerzona o nowe moduły obliczeniowe (MVAP i SYMULACJA), dodano także moduły umożliwiające interaktywną pracę z pakietem, a także rozszerzono znacznie możliwości prezentacji wyników (wykresy, histogramy itp.).

3. Wersja pakietu AMOK dla stacji roboczych SUN

Stały postęp w dziedzinie rozwoju sprzętu komputerowego i technik programowania umożliwił opracowanie nowej wersji pakietu AMOK, przeznaczonej dla stacji roboczych SUN pracujących pod systemem operacyjnym UNIX.

Nowa wersja pakietu AMOK nie jest bezpośrednią konwersją wersji dla IBM PC. Została ona zaprojektowana od nowa, jako zestaw obiektów w języku C++. Zapewnia to elastyczną, rozwojową architekturę całości pakietu, dającą możliwość łatwej modyfikacji pakietu AMOK w przyszłości, gdyby zaszła potrzeba jego rozszerzenia lub zmiany przyjętych rozwiązań. Z drugiej strony, algorytmy analizy sieci nie są podatne na zmiany techniki implementacji i marnotrawstwem czasu i energii byłoby powtórne ich tworzenie. Dlatego w nowej wersji pakietu AMOK zastosowano przetestowane i poprawione w ciągu 10 lat eksploatacji procedury obliczeniowe napisane jeszcze dla pierwszej wersji pakietu w języku Fortran IV.

Taka koncepcja spowodowała podział całości projektu na dwie części: nową - obiektową, realizującą etap budowy modelu sieci i starą - proceduralną, odpowiedzialną za wykonanie obliczeń. Ich integracja stanowiła jeden z problemów przy opracowywaniu pakietu AMOK w nowej wersji i zostanie opisana w dalszej części artykułu.



Rys. 1. Struktura pierwszej wersji pakietu AMOK
Fig. 1. Structure of the first version of AMOK

3.1. Obiektowy model sieci stanowisk obsługi

Nowa koncepcja budowy modelu sieci stanowisk obsługi polega na utworzeniu w języku C++ zestawu obiektów reprezentujących poszczególne elementy sieci wraz z zestawem metod pozwalających parametryzować te obiekty i łączyć je ze sobą. Dzięki temu nie jest potrzebne definiowanie języka opisu modelu, a co za tym idzie, nie trzeba pisać programu tłumaczącego ten opis na postać binarną (dawny CONV), gdyż jego rolę spełnia niejako kompilator języka C++. Ma to niebagatelne znaczenie w przypadku planowania rozwoju formalnego opisu sieci, gdyż pozwala uniknąć modyfikacji (ponownego pisania) programu tłumaczącego.

Każdy element sieci jest reprezentowany przez obiekt właściwej klasy, który zawiera parametry tego elementu, zestaw metod umożliwiających przeglądanie i zmianę tych parametrów, a docelowo także czynności wykonywane przez ten element. Ta obiektowa konwencja zapewnia bezpośrednią łączność pomiędzy problemem i będącym jego rozwiązaniem

programem, pozwala także na wprowadzenie systematyki elementów sieci, poprzez zbudowanie odpowiedniej hierarchii klas.

Stworzona została podstawowa grupa klas definiujących sieć, dobrana pod kątem przyjętego algorytmu analizy modelu, którym jest metoda wartości średnich z priorytetami (MVAP) [7]. Poszczególne klasy tej grupy reprezentują: całą sieć ('Network'), stanowisko obsługi ('SimpleServer') oraz jego części wyodrębnione jako osobne elementy (rodziny klas 'Queue' i 'Service').

Oprócz klas reprezentujących typy elementów sieci, czyli służących do tworzenia modelu, zostały zbudowane klasy pomocnicze służące do analizy tego modelu i prezentacji wyników.

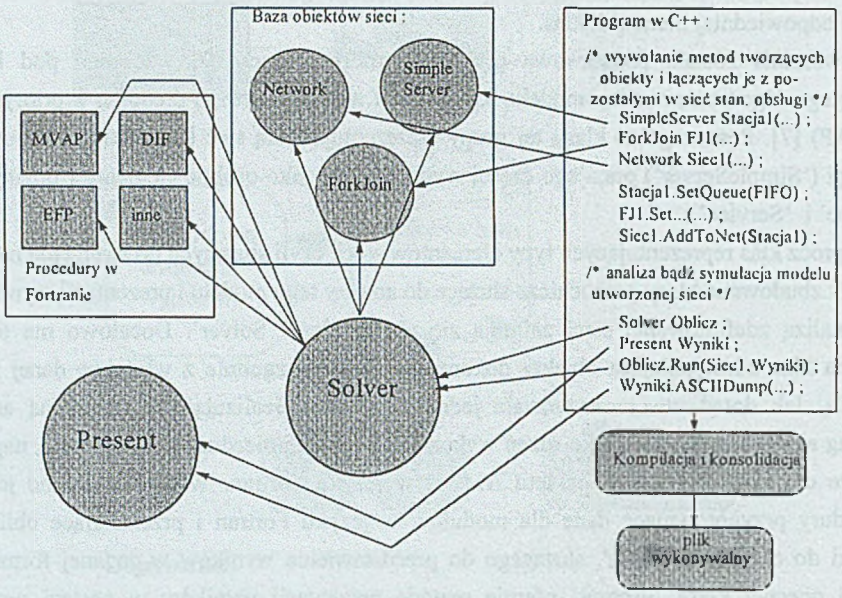
Analizą zdefiniowanej sieci zajmują się obiekt klasy 'Solver'. Docelowo ma to być rodzina klas udostępniająca obiekty dokonujące obliczeń zgodnie z właściwą danej klasie metodą. Jak dotąd utworzona została jedna taka klasa, realizująca wspomnianą analizę według algorytmu MVAP. Jako rdzeń wykorzystano stare procedury obliczeniowe, napisane jeszcze dla pierwszej wersji pakietu AMOK w języku Fortran. W C++ napisano jedynie procedury przygotowujące dane dla modułów w języku Fortran i przekazujące obliczone wyniki do obiektu 'Present', służącego do przedstawienia wyników w żądanej formie. W chwili obecnej klasa 'Present' oferuje metodę prezentacji wyników w postaci prostego wydruku znakowego. Pozwala ona ocenić poprawność przyjętych rozwiązań.

Tak przygotowana baza obiektów dostarcza elementów, z których użytkownik buduje model analizowanej sieci pisząc program w C++, następnie kompiluje go i uruchamia.

Całość struktury nowej wersji pakietu przedstawia rys. 2.

Przedstawienie koncepcji modelowania sieci za pomocą obiektów uzupełnia szkic hierarchii klas pakietu (rys. 3).

Szczegóły implementacyjne klas zostały przedstawione w pracy [1].



Objaśnienia:

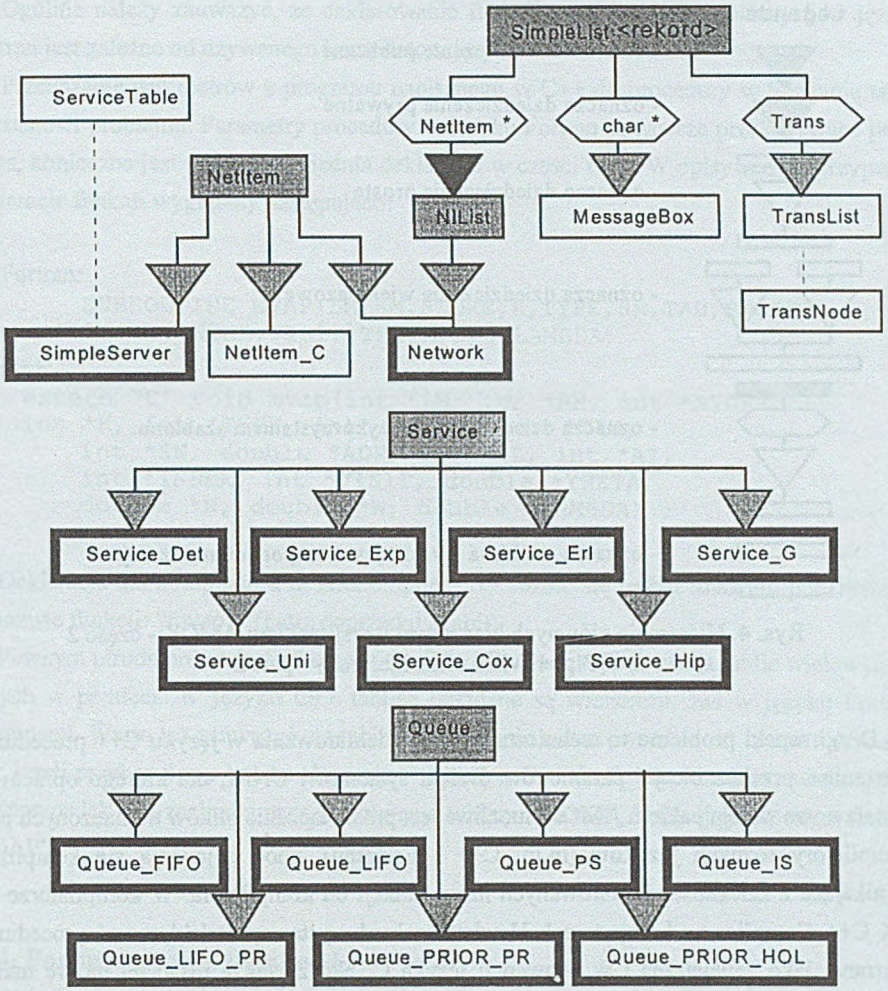
- Koła oznaczają obiekty zawierające metody służące do tworzenia sieci stanowisk obsługi (tylko obiekty reprezentujące elementy sieci) oraz do obliczenia parametrów użytkowych sieci.
- Strzałki oznaczają wykonanie metod danego obiektu lub wykonanie w języku Fortran.

Rys. 2. Struktura nowej wersji pakietu AMOK
Fig. 2. Structure of the new version of AMOK

3.2. Integracja procedur w Fortranie z resztą pakietu

Obliczeniem parametrów użytkowych sieci na podstawie zbudowanego modelu zajmują się w ramach nowej wersji pakietu AMOK procedury napisane w języku Fortran IV jeszcze dla pierwszej wersji pakietu, które następnie były poprawiane i rozbudowywane. Ponieważ sam model jest definiowany poprzez zbudowanie go z obiektów (patrz p. 3.1) napisanych w C++, powstał problem integracji tych dwóch różnych środowisk.

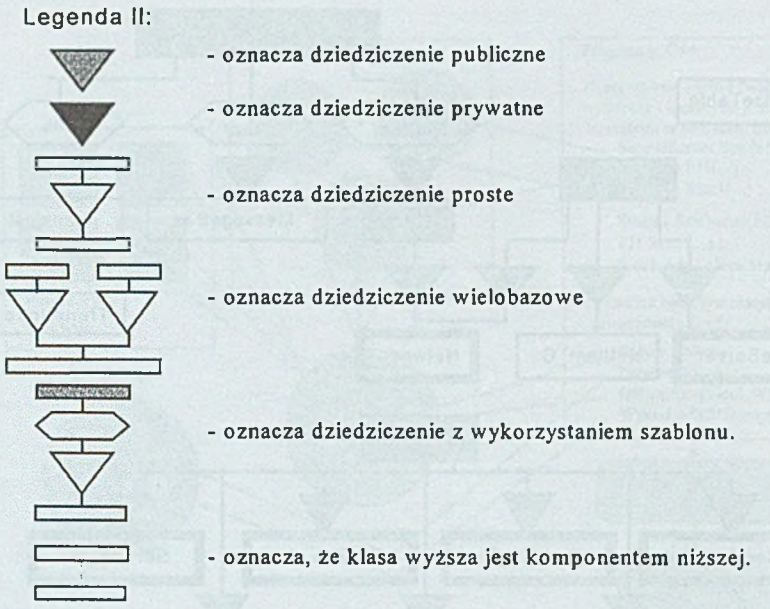
Pierwszy aspekt problemu polega na logicznym umiejscowieniu części napisanej w Fortranie w bazie obiektów stworzonych w C++. Utworzono klasę 'MVAP_Solver', realizującą analizę modelu. W klasie tej zdefiniowano metodę 'run', w której następuje wywołanie procedury 'mvap', napisanej w Fortranie. Procedura ta jest globalna dla całego programu, co jest rozwiązaniem najprostszym, choć może nie najbardziej eleganckim.



Objaśnienie:

- oznacza klasę abstrakcyjną.
- oznacza klasę pomocniczą, nieistotną dla użytkownika.
- oznacza klasę - komponent SSO (istotną dla użytkownika).
- oznacza klasę sparametryzowaną (wykorzystującą tzw. szablon).
- oznacza parametr - czyli wypełnienie szablonu - klasy sparametryzowanej.

Rys. 3. Hierarchia głównych klas nowej wersji pakietu AMOK - część 1
Fig. 3. Main class hierarchy - part 1



Rys. 4. Hierarchia głównych klas nowej wersji pakietu AMOK - część 2
Fig. 4. Main class hierarchy - part 2

Drugi aspekt problemu to techniczna strona zadeklarowania w języku C++ procedury w Fortranie i przekazanie jej parametrów. Dzięki systemowi UNIX, dla którego opracowana została nowa wersja pakietu AMOK, możliwe jest proste łączenie plików tłumaczonych przez kompilatory różnych języków (m.in. C++ i Fortranu), choć pojawiają się komplikacje wynikające z zależności deklarowanych nazw funkcji od kompilatora. W kompilatorze HP-UX C++ Compiler na komputerach Hewlett-Packard wystarczy zadeklarować procedurę w Fortranie jako zewnętrzną i w konwencji języka C. Należy także pisać jej nazwę małymi literami, gdyż tak zapisuje ją kompilator Fortranu. Deklaracja w tekście programu w C++ powinna wyglądać następująco:

```
extern "C" mvap(<parametry>) ;
```

Natomiast dla kompilatora SPARCompiler C++ na komputer SUN konieczne jest dodanie na końcu nazwy znaku podkreślenia, który jest dodawany przez kompilator Fortranu w tym systemie. W tym przypadku deklaracja musi wyglądać jak poniżej:

```
extern "C" mvap_(<parametry>) ;
```

Ogólnie należy zauważyć, że deklarowanie funkcji zewnętrznych napisanych w języku Fortran jest zależne od używanego kompilatora języków C++ i Fortranu.

Przekazanie parametrów z programu napisanego w C++ do procedury w Fortranie także nie stanowi problemu. Parametry procedury w języku Fortran są zawsze przekazywane przez adres, konieczna jest więc odpowiednia deklaracja w części C++. W opisywanym przypadku deklaracje funkcji wyglądały następująco:

Fortran:

```
      SUBROUTINE MVAP(LM,RM,STORE,K,TYPE,SR,TAU,SN,ADP,
     * AL,AI,INDEX,VISIT,THETA,N,W,LAMBDA)
```

C++:

```
extern "C" void mvap(int *LM, int *RM, int *STORE,
int *K, int *TYPE, int *SR, double *TAU,
    int *SN, double *ADP, int *AL, int *AI,
    int *INDEX, int *VISIT, double *THETA,
    double *N, double *W, double *LAMBDA) ;
```

Deklaracja dla kompilatora SPARCompiler C++ różnić się będzie znakiem podkreślenia po nazwie funkcji: "mvap_" (patrz poprzedni akapit).

Pewnym utrudnieniem jest odmienna dla C++ i Fortranu reprezentacja tablic wielowymiarowych w pamięci. W języku C++ tablice układane są wierszami, zaś w języku Fortran kolumnami. Warto też zauważyć, że tablice w Fortranie numerowane są od 1, a w C++ od 0, więc jeżeli przekazujemy indeks elementu w tablicy z C++ do Fortranu, należy zwiększyć go o 1 (przypadek taki zachodzi przy wypełnianiu tablicy 'INDEX', będącej parametrem funkcji 'MVAP').

4. Podsumowanie i wnioski

Ewentualne wnioski mogą zostać sformułowane jedynie na podstawie doświadczeń nabytych w procesie tworzenia nowej wersji pakietu AMOK, jako że trudno bowiem mówić o eksploatacji samego produktu w sytuacji, gdy jest on zaledwie prototypem o stosunkowo ograniczonych możliwościach. Niemniej jednak, można określić korzyści, jakie dało zastosowanie obiektowej koncepcji modelu sieci do budowy pakietu. Po pierwsze, znacznie uprościło to etap analizy zagadnienia i projektowania struktur danych, ponieważ programowy model problemu odpowiada dość dokładnie rzeczywistemu problemowi. Z tego samego powodu rozbudowa i ewentualne modyfikacje pakietu powinny być łatwiejsze do wykonania, gdyż czas potrzebny do poznania jego struktury jest krótszy, sama zaś struktura pewniejsza i odporniejsza na błędy niż w poprzedniej wersji. Po drugie, bezpośrednie przełożenie

elementów sieci na zespół obiektów w języku programowania zamiast definiowania specjalnego języka opisu modelu wyeliminowało konieczność pisania kompilatora tego języka, co znacznie upraszcza wprowadzanie zmian w formalnej części pakietu.

Ujemną stroną nowej koncepcji jest obsługa pakietu, szczególnie przez osoby nie znające języka C++. Stała się ona trudniejsza i bardziej kłopotliwa. Język opisu modelu QNDEL był specjalizowany pod kątem opisu modeli kolejkowych, w przeciwieństwie do języka C++, który ma własną składnię. Składnia ta wymusza także na użytkowniku deklarację zbędnych szczegółów, np. konieczność podawania liczby elementów w wywołaniach funkcji z niezdefiniowaną liczbą parametrów, co nie było konieczne w wersji opisu modelu za pomocą języka QNDEL. Sposób obsługi pakietu wymaga więc jeszcze przemyślenia i dopracowania.

Pewne obawy wiązały się z połączeniem dwóch zasadniczo różnych środowisk programowania - języków C++ i Fortran. Problem ten został rozwiązany w sposób zadowalający, choć nie zasługujący na miano "integracji". Możliwe więc jest wykorzystywanie w programach obiektowych starszych "wstawek", choć niewątpliwie ma to także ujemne strony w postaci np. globalnych procedur, istniejących poza strukturą obiektów programu.

Mimo wad nowa wersja pakietu AMOK jest niewątpliwie nowocześniejsza i bardziej przyszłościowa od takiej, w której powtórzono by koncepcję dawnej wersji dla komputera ODRA 1305, tak jak uczyniono to w przypadku wersji dla IBM PC. Wykorzystanie obiektowej techniki programowania przy jej konstruowaniu zapewniło łatwość modyfikacji i zwiększyło niezawodność pakietu, umożliwiając przy okazji autorowi doskonalenie swoich umiejętności i pogłębienie zrozumienia tego narzędzia tworzenia oprogramowania.

LITERATURA

[1] Meres M.: Implementacja programów obliczeniowych systemu AMOK dla stacji roboczych SUN, pod systemem operacyjnym UNIX. Praca dyplomowa magisterska, Politechnika Śląska, Gliwice 1995.

[2] Czachórski T., Kowalówka M., Szczerbiński Z., Tomasik J., Wilk A., Wołowiec W.: Modelowanie i ocena pracy systemów komputerowych za pomocą AMOK-u. Skrypt uczelniany Politechniki Śląskiej nr 1626, Gliwice 1991.

[3] Czachórski T.: Modele kolejkowe systemów komputerowych. Skrypt uczelniany Politechniki Śląskiej nr 1844, Gliwice 1994.

[4] Barteczko K.: Praktyczne wprowadzenie do programowania obiektowego w języku C++. Wydawnictwo 'Lupus', Warszawa 1994.

[5] Sun FORTRAN User's Guide (FORTRAN 1.3.1.). Sun Microsystems, Inc, Mountain View (USA). Revision: A of 5 July 1990.

[6] Trykozko A.: Fortran 77: podstawy programowania. Zakład Nauczania Informatyki 'MIKOM', Warszawa 1994.

[7] Tomasik J.: Opracowanie modułu programowego umożliwiającego modelowanie systemów komputerowych metodą analizy wartości średnich z priorytetami. Praca dyplomowa magisterska, Politechnika Śląska, Gliwice 1989.

[8] Dokumentacja pakietu AMOK. Opracowanie wewnętrzne IITiS PAN, Gliwice 1988.

Recenzent: Dr inż. Zdzisław Szczerbiński

Wpłynęło do Redakcji 18 marca 1996 r.

Abstract

Queue models are very useful in solving problems of complex real systems of many sorts. They help to determine parameters of these systems and make them faster and more effective. But large systems need large models and large models need many computing power and that's a point when programs like AMOK come in handy. AMOK is a group of programs which helps with defining and analysing queue models. The very first version was dedicated for ODRA 1305 computers and the next for IBM computers (Fig. 1.). It consisted of several separate modules: the QNDEL-language compiler, some numerical processing modules and the presentation module (QNDEL is a specially defined language of model description). Each module utilised common data which was stored in file. This architecture was good enough for small computers, but it lacked of flexibility. So, the new version was designed for UNIX operating system, especially SUN computers.

The new architecture consist of two major parts. The first is a set of objects designed for defining the model; it has been written in C++. The second part are some procedures in Fortran inherited by old version of AMOK (fig. 2.). Fortran procedures are called from within the C++ object called 'Solver', and results of computing are transfered to another object called 'Present'. User creates model by defining objects variables with appriorate parameters and calling methods of defined objects in ordinary C++ source text. After that method 'Run' of 'Solver' object must be called which means the model is analysed. Results are transfered to 'Present' automatically and user choose only the method of presentation. The source code can be compiled with standard C++ compiler so there is no need to define QNDEL-like language. Figure (3) shows main class hierarchy.

It was mentioned that new version of AMOK consist of two major parts. There was some troubles with integrating both the C++ object-oriented part and the Fortran procedural part of project. Finally, Fortran procedure calls were located within the method 'Run' of object 'Solver'. The procedures themselves were made global and declared as 'extern' in C++ source text. All parameters were referred by pointers and two dimensional arrays were converted to one dimension due to differences in their memory representation in C and Fortran.

The new version of AMOK designed for SUN computers is more flexible and reusable then previous versions. Object-oriented design decreases program vulnerability to errors and make it more durable. The program functions and structure are more obvious and it makes future changes easier to do. Although there are some problems still unsolved the new verion is a real improvement to former concepts.