ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: AUTOMATYKA z.114 Nr kol. 1250

Joanna JÓZEFOWSKA, Piotr PIESIK, Jan WĘGLARZ
Politechnika Poznańska

MODELOWANIE I WERYFIKACJA DECYZJI PROJEKTOWYCH I OPERACYJNYCH W ELASTYCZNYCH SYSTEMACH PRODUKCYJNYCH ZA POMOCĄ PROGRAMU SYMULACYJNEGO RUNSIM

Streszczenie: W pracy przedstawiono program symulacyjny RUNSIM, służący do modelowania elastycznych systemów produkcyjnych. Modułowa struktura programu pozwala definiować wiele różnych systemów produkcyjnych na poziomie wprowadzania danych wejściowych. Ten sposób definiowania systemu pozwala łatwo dostosować program do rzeczywistego systemu produkcyjnego bez zmiany kodu źródłowego. W konsekwencji uzyskuje się znaczne zmniejszenie kosztów badań symulacyjnych, co istotnie zwiększa zakres ich stosowania.

MODELLING AND VERIFICATION OF THE DESIGN AND OPERATIONAL DECISIONS IN FLEXIBLE MANUFACTURING SYSTEMS USING SIMULATION PROGRAM RUNSIM

Summary: A simulation package RUNSIM which is designed for modelling and control of Flexible Manufacturing Systems is presented. The modular structure of the program enables an easy description of many different FMS using the input data. This way of defining a production system makes it possible to change modelled systems without changing the program code. Consequently a substantial decline in the simulation costs is achieved.

DIE MODELLIERUNG UND VERIFIKATION VON ENTWURFS- UND LAUFZEITENTSCHEIDUNGEN IN FLEXIBLEN FERTIGUNGSSYSTEMEN MIT HILFE DES SIMULATIONSPROGRAMMS RUNSIM

Zusammenfassung: Wir stellen die Simulationsprogramm RUNSIM vor, welches zur Modellierung und Kontrolle flexibler Fertigungssysteme (FMS's) entwickelt wurde. Die Modulstruktur des Programmes erlaubt es, verschiedene FMS ausschließlich mit Hilfe der Eingabedaten zu beschreiben. Daher ist es möglich, das zu modellierende System zu wechseln, ohne den Programmcode zu modifizieren. Diese Eigenschaft des Programms bringt eine wesentliche Verringerung der Simulationskosten mit sich.

1. Wstęp

Projektowanie elastycznych systemów produkcyjnych (ESP) jest etapem decydującym o efektywności przygotowywanej inwestycji. Wysokie koszty każdego z elementów projektowanego systemu (maszyn, urządzeń transportowych, palet, uchwytów) powodują, że głównym zadaniem optymalizacyjnym na tym etapie staje się problem minimalizacji nakładów dla osiągnięcia założonego celu. Złożoność tego problemu wynika z następujących cech elastycznych systemów produkcyjnych:

- założony cel jest na ogół zdefiniowany jakościowo i za pomocą wielu kryteriów,
- liczba elementów systemu, które należy wziąć pod uwagę, jest duża,
- pewne decyzje mogą wywoływać efekt synergii,
- asortyment produkcji jest zmienny w pewnym zakresie,
- programy produkcji są zmienne nawet w krótkich okresach czasu.

Jak dotąd najbardziej skuteczną metodą pozwalającą ocenić projekt ESP jest symulacja. Równocześnie jednak, budowa symulatora jest możliwa, gdy pewne decyzje projektowe zostały już podjęte (np. liczba maszyn, typ ESP) i jest zadaniem bardzo pracochłonnym. Biorąc pod uwagę fakt, że taki symulator może modelować tylko ten konkretny ESP, nietrudno uzasadnić, że zastosowanie tego narzędzia pociąga za sobą bardzo wysokie koszty. Powyższe ograniczenia są przyczyną niewielkiego wykorzystania symulacji w projektowaniu ESP. Skutkiem tego efektywność wielu wdrożonych systemów okazała się mniejsza niż oczekiwano. W literaturze proponuje się wprowadzenie narzędzi w postaci symulatorów o budowie modułowej do szybkiego konstruowania modeli symulacyjnych konkretnych ESP. Zastosowanie tego rozwiązania wymaga dobrej znajomości proponowanego oprogramowania i nie gwarantuje dostatecznej elastyczności przy budowie modelu.

W niniejszej pracy przedstawiono symulator ESP, w którym liczba i konfiguracja elementów systemu są wprowadzane w postaci danych wejściowych, dzięki czemu zmiana typu ESP nie wymaga żadnej ingerencji w kod źródłowy programu. Ponadto algorytmy szeregowania zadań zastosowane do sterowania systemem można wybrać spośród załączonych procedur lub dołączyć dowolną skompilowaną procedurę, zgodną z modułami wejścia/wyjścia symulatora. Należy się spodziewać, że koszty symulacji ESP przy wykorzystaniu systemu RUNSIM będą zaniedbywalnie małe w stosunku do kosztów inwestycyjnych. RUNSIM może być również wykorzystywany przy weryfikacji decyzji operacyjnych podejmowanych w fazie eksploatacji systemu produkcyjnego (wybór algorytmu sterującego, usuwanie zagrożeń i zakłóceń itp.).

2. Koncepcja modelowania elastycznego systemu produkcyjnego

Celem opracowania symulatora RUNSIM było uzyskanie możliwości definiowania praktycznie dowolnego elastycznego systemu produkcyjnego za pomocą danych wejściowych wprowadzanych w sposób łatwy dla użytkownika. W tym celu opracowano ogólny model ESP, w którym wyróżniono następujące obiekty: maszyny, stanowiska, palety, uchwyty, narzędzia, system transportowy, system magazynowy.
Każdy obiekt jest scharakteryzowany przez wartości przypisanych mu atrybutów np.:

palety (nazwa, pojemność, liczba)
wózki (liczba, prędkość).

Zbiór wartości atrybutów wszystkich obiektów stanowi opis konkretnego systemu. Dla każdego atrybutu przewidziano wartość domyślną, na wypadek gdyby użytkownik nie umiał określić wartości niektórych atrybutów. Wartości domyślne są tak przyjęte, aby nie wprowadzały dodatkowych ograniczeń (np. brak danych odnośnie do systemu transportowego powoduje zaniedbanie czasu wykonania operacji transportowych podczas symulacji i dopuszcza wszystkie żądane operacje transportowe). Dane dotyczące procesów technologicznych detali oraz przewidywanej wielkości produkcji mogą być wprowadzane przez użytkownika lub generowane losowo. Na dane te składają się: nazwa (typ detalu), wielkość partii do jednoczesnej obróbki, wymagany typ palety, operacje: numer kolejny operacji, nazwa operacji, czas potrzebny na wykonanie operacji, wymagany typ uchwytu, alternatywne maszyny, na których operacja może być wykonana, narzędzia potrzebne do wykonania operacji. W ten sposób możliwe jest określenie (implicite) typu systemu (linia, gniazdo, itp.) oraz możliwość wystąpienia alternatywnych procesów technologicznych. Opisany schemat danych narzuca oczywiście pewne ograniczenia, ale jest wystarczająco ogólny dla modelowania praktycznie wszystkich typów elastycznych systemów produkcyjnych.

Program posiada również moduł sprawdzania poprawności danych, zabezpieczający przed wprowadzeniem danych sprzecznych (np. technologii wymagającej systemu typu job-shop z jednokierunkowym systemem transportowym).

3. Koncepcja wyboru algorytmu szeregowania zadań

Umożliwienie łatwego definiowania różnych typów systemów produkcyjnych pociąga za sobą konieczność uwzględnienia tej różnorodności na poziomie algorytmów sterowania systemem. Ponieważ kryterium optymalności uszeregowania w różnych systemach przybiera różną postać, funkcję celu w systemie RUNSIM może również zdefiniować użytkownik.

Zależnie od potrzeb może on wykorzystać algorytmy szeregowania zadań pracujące w trybie on-line i off-line. Ponadto zestaw algorytmów załączonych do symulatora można w dowolny sposób rozszerzać o skompilowane procedury szeregowania, zgodne z modułami wejścia/wyjścia symulatora.
Wśród załączonych algorytmów szeregowania znajdują sie m.in. FIFO (First In First Out), LPT (Longest Processing Time), SPT (Shortest Processing Time), EDD (Earliest Due Date).
Możliwość elastycznego wyboru algorytmu szeregowania zadań jest istotna zarówno na etapie projektowania ESP, jak i podczas jego eksploatacji. Na etapie projektowania celowe jest porównanie różnych wariantów konfiguracji systemu przy zastosowaniu różnych algorytmów sterujących jego pracą. Przy niewielkich czasach przebiegu symulacji można tą drogą wspomagać również decyzje operacyjne, jak np. rozwiązanie problemu awarii obrabiarki, czy zbyt długiej kolejki zadań na jakimś stanowisku.
Większość problemów szeregowania zadań występujących w praktyce to problemy obliczeniowo trudne, co jest przyczyną stosowania algorytmów heurystycznych. Wybór najlepszej heurystyki zależy od typu systemu, charakterystyki zadań oraz kryterium optymalności. Ponieważ RUNSIM pozwala modelować różne systemy produkcyjne oraz różne kryteria, konieczne jest włączenie różnych algorytmów szeregowania. Liczba znanych heurystyk jest jednak bardzo duża i ciągle powstają nowe. Aby umożliwić rozwój programu zgodnie z potrzebami użytkownika, dopuszcza się zatem dołączanie dodatkowych algorytmów szeregowania. Algorytmy te mogą być zaprogramowane w dowolnym języku i po kompilacji dołączone do biblioteki programu RUNSIM. Szczegóły implementacyjne tego rozwiązania przedstawiono w następnym rozdziale.

4. Opis implementacji

4.1. Środowisko i narzędzia

Program symulujący elastyczne systemy produkcyjne jest przeznaczony do uruchomienia na komputerze IBM PC 386, w środowisku Microsoft Windows 3.1, przy użyciu kompilatora Borland Pascal 7.0 wraz z biblioteką Object Windows. Środowisko Windows zostało wybrane z kilku powodów. Po pierwsze daje ono możliwość opracowania bardzo dobrego interfejsu programu z użytkownikiem. Windows jest tu niekwestionowanym standardem środowiska graficznego. Ponadto jest systemem wielozadaniowym, co pozwala na jednoczesne uruchomienie wielu symulacji z różnymi parametrami, nie blokuje komputera na czas przebiegu programu oraz ułatwia wymianę danych z innym oprogramowaniem, np. z systemem eksperckim (z którego mogą pochodzić dane wejściowe dla symulacji) czy

z arkuszem kalkulacyjnym lub programem graficzno - prezentacyjnym (które mogą służyć do obróbki danych wyjściowych). Zarządzanie pamięcią komputera 386 przez Windows 3.1 jest wielkim ułatwieniem dla programisty i pozwala zapomnieć o niedogodnościach korzystania z pamięci w systemie DOS. Windows daje programom dostęp do znacznej ilości pamięci dzięki mechanizmowi pamięci wirtualnej, której wielkość może być definiowana przez użytkownika stosownie do faktycznych potrzeb.

4.2. Ogólna struktura programu

Program składa się z następujących modułów:

- moduł definiowania elastycznego systemu produkcyjnego,
- moduł definiowania zadań dla systemu produkcyjnego,
- interfejs do podłączania wymiennych algorytmów sterujących,
- moduł sprawdzający poprawność danych,
- właściwy symulator,
- moduł obróbki danych wyjściowych,
- system pomocy (help).

Wymienione moduły są napisane w języku obiektowym i połączone w jedną aplikację pracującą pod Windows. Jedynie algorytmy sterujące nie są programami wykonywalnymi, lecz dynamicznie łączonymi bibliotekami (DLL, Dynamic Link Library) współpracującymi z programem głównym. Poniżej scharakteryzujemy poszczególne moduły.

4.3. Charakterystyka modułów programu

Moduł definiowania elastycznego systemu produkcyjnego służy do wprowadzania, edycji i zapamiętywania danych opisujących badany system. Na dane te składają się:

- typy maszyn (nazwa, pojemność głowicy, liczba buforów wejściowych i wyjściowych),
- stanowiska (nazwa, typ, odległość od stacji za- i wyładowczej, zamontowane narzędzia),
- palety (nazwa (typ), pojemność (liczba detali), liczba dostępnych palet danego typu),
- uchwyty (nazwa (typ), liczba dostępnych uchwytów danego typu),
- narzędzia (nazwa (typ), liczba dostępnych narzędzi danego typu),
- inne zasoby,
- wózki (liczba, prędkość),
- transportery narzędzi (liczba, prędkość),

- stacje za- i wyładowcze (liczba, czas operacji),
- możliwe połączenia transportowe pomiędzy maszynami (maszyna 1, maszyna 2, odległość).

Moduł definiowania zadań dla systemu służy do wprowadzania, edycji i zapamiętywania danych opisujących zadania (tzn. detale do obróbki). Na dane te składają się:

- nazwa (typ) detalu,
- wielkość partii do jednoczesnej obróbki,
- wymagany typ palety,
- operacje (numer kolejny operacji, nazwa operacji, czas potrzebny na wykonanie, wymagany typ uchwytu, alternatywne maszyny, na których operacja może być wykonana, narzędzia potrzebne do wykonania operacji)

Interfejs do podłączania wymiennych algorytmów jest dość rozbudowany. Wynika to z konieczności podłączania, oprócz algorytmów off-line, również algorytmów on-line. Dlatego opracowano koncepcję dwóch poziomów interfejsu: interfejs ogólny oraz interfejs do algorytmów off-line, oba dostosowane do współpracy z formatem biblioteki DLL.

Dane wejściowe dla algorytmu obejmują:

- informacje o maszynach,
- informacje o systemie transportowym,
- informacje o zasobach,
- dane na temat obrabianych detali,
- czasy przejazdu, obróbki, za- i wyładunku.

Przekazanie tych danych do algorytmu polega na wywołaniu przez program procedury z DLL, z parametrem określającym wskaźnik (adres w pamięci) do miejsca, gdzie przechowuje swoje dane. Jednak biblioteka powinna zachować te dane niezmienione, tzn. korzystać z nich wyłącznie do odczytu.

Algorytm off-line na podstawie tych danych wyznacza uszeregowanie, przy czym wyróżnia się następujące dwa przypadki:

- algorytm wyznaczył wyłącznie uszeregowanie maszyn, co oznacza, że program powinien zarządzać wózkami w czasie rzeczywistym,
- algorytm wyznaczył zarówno uszeregowanie maszyn, jak i odpowiadające mu dopuszczalne uszeregowanie wózków - wówczas program korzysta z wyznaczonego uszeregowania, rezygnując ze sterowania wózkami w czasie rzeczywistym.

W przypadku awarii program zwraca się do algorytmu z kolejnym zadaniem szeregowania, uwzględniając brak urządzeń, które uległy uszkodzeniu. Jeśli algorytm wyznaczy dopuszczalne uszeregowanie, to program kontynuuje pracę (chyba że użytkownik explicite zaznaczy, że system ma się zatrzymać po wykryciu awarii).

Algorytm on-line potrzebuje dotatkowych informacji:

- adres (w programie głównym) procedur wykonujących podstawowe czynności w systemie: ładowanie detalu na wózek, przejazd wózka do wyznaczonego miejsca, rozpoczęcie wykonywania zadania na maszynie, załadowanie narzędzi do maszyny, diagnostyka maszyny,
- adres procedur wspomagających symulację, tzn.: czas bieżący, zatrzymanie symulacji.

Moduł badania poprawności danych sprawdza kolejno, czy spełnione są następujące warunki:

- istnieje choć jedna maszyna,
- istnieje choć jedna paleta,
- istnieje choć jeden uchwyt,
- istnieją jakiekolwiek narzędzia,
- zdefiniowano detale do obróbki,
- żaden detal nie wymaga nie istniejącej palety,
- istnieje choć jedna operacja dla każdego detalu,
- żadna operacja nie wymaga nie istniejącego uchwytu bądź narzędzia,
- istnieją maszyny potrzebne do wykonania wszystkich operacji,
- żadna wielkość partii do równoczesnej obróbki nie przekracza pojemności odpowiedniej palety,
- podłączono odpowiedni algorytm sterujący,
- marszruty technologiczne odpowiadają możliwościom systemu transportowego.

Program ma dwa (wybierane przez użytkownika) tryby reagowania na błędy wykryte w czasie testowania poprawności danych:

- zatrzymuje się na pierwszym wykrytym błędzie,
- sprawdza całość, po czym pokazuje błędy po kolei na liście.

W obu trybach możliwe jest automatyczne wejście w to miejsce w danych, w którym wystąpił błąd.

Symulator służy do uruchomienia symulacji zdefiniowanego systemu produkcyjnego. W tym celu potrzebne jest (oprócz zdefiniowania systemu) także zdefiniowanie zadań oraz podłączenie algorytmu sterującego (algorytmy sterujące powinny być zgodne z formatem DLL, opisanym w literaturze [8]).

Metoda działania symulatora jest uzależniona od rodzaju algorytmu sterującego. Wyróżniane są dwa zasadnicze przypadki:

A. W przypadku podłączenia algorytmu off-line symulator najpierw wywołuje algorytm (a konkretnie zawartą w DLL procedurę szeregowania), a następnie wykonuje symulację, uwzględniając otrzymane uszeregowanie oraz dane na temat systemu i zadań. Podczas

symulacji generowane są (jeśli użytkownik włączył odpowiednią opcję) awarie (proces o rozkładzie normalnym i parametrach zdefiniowanych przez użytkownika). W przypadku niesprawności maszyny lub urządzenia, które jest akurat potrzebne w procesie obróbki, symulator zatrzymuje system i wykonuje jedną z dwóch możliwych akcji (zależnie od polecenia użytkownika): ponownie wywołuje procedurę szeregowania z uwzględnieniem nowych warunków albo zatrzymuje system na zadany czas w celu zasymulowania naprawy urządzenia, po czym znów szuka uszeregowania i kontynuuje pracę. Oczywiście w przypadku pierwszym może się zdarzyć, że nie istnieje dopuszczalne uszeregowanie. Wtedy jedynym możliwym rozwiązaniem jest naprawa uszkodzonych elementów systemu.

B. W przypadku podłączenia algorytmu on-line rola symulatora jest ograniczona do cyklicznego wywoływania zawartej w DLL procedury sterowania systemem i generowania losowych awarii. Algorytm przejmuje właściwe sterowanie symulowanym systemem produkcyjnym. Oznacza to między innymi, że reagowanie na awarie jest obowiązkiem algorytmu, a nie symulatora. Program główny ogranicza się wówczas jedynie do informowania użytkownika o aktualnym stanie rzeczy. Planowane jest jednak znaczne rozbudowanie funkcji programu głównego w przypadku obsługi systemu przez algorytm on-line.

Symulacja zrealizowana jest przy użyciu metody przeglądania zdarzeń, przy czym jest tu ona dostosowana do obiektowej filozofii programu. Poszczególne elementy systemu są reprezentowane za pomocą odpowiadającej im hierarchii obiektów, co znacznie ułatwia ich organizację. Program główny (będący również obiektem) komunikuje się z nimi za pomocą komunikatów (message). Jest to metoda znana i zalecana w programowaniu obiektowym [8]. Każdy rodzaj elementów systemu (maszyny, wózki, palety itd.) jest reprezentowany przez odpowiednią klasę (typ) obiektu. Tworzone są odpowiednie podklasy do reprezentacji np. maszyn czy zasobów określonego typu. Odpowiednie obiekty (instancje) reprezentują konkretne maszyny, wózki transportowe i zasoby (tzn. każdy pojedynczy element systemu jest reprezentowany przez instancję obiektu odpowiedniej klasy). Każdy obiekt reaguje na dwa rodzaje komunikatów: polecenia (np. dla maszyny załadowanie detalu z bufora na stół roboczy jest poleceniem) oraz zapytania (np. maszynę można "zapytać", czy ma zajęty bufor, czy jest gotowa do rozpoczęcia wykonywania zadania itp.). Wszystkie dane na temat stanu (zarówno aktualnego, jak i "historii" od początku obróbki) każdego elementu systemu są pamiętane w polach odpowiadającego mu obiektu (instancji). Zatem po zakończeniu symulacji możliwe jest przeprowadzenie analizy całego przebiegu. Poniżej opisano zdarzenia, które każdy obiekt pamięta w swoich polach (wraz z momentem ich wystąpienia):

- dla maszyn: załadowanie detalu, rozpoczęcie obróbki, zakończenie obróbki, oddanie detalu do bufora wyjściowego, załadowanie narzędzi, wystąpienie awarii, usunięcie awarii,
- dla buforów: przyjęcie palety, oddanie palety,

- dla wózków: załadowanie detalu (detal i miejsce), dostarczenie detalu (miejsce), wystąpienie awarii, usunięcie awarii,
- dla palet i uchwytów: załadowanie detalu, wyładowanie detalu, wyjazd z magazynu, powrót do magazynu,
- dla narzędzi: załadowanie do głowicy maszyny, wymontowanie z głowicy, wyjazd z magazynu, powrót do magazynu,
- dla detali: załadowanie z magazynu na paletę, załadowanie na stół roboczy w maszynie, załadowanie z maszyny na paletę, powrót do magazynu.

Moduł obróbki danych wyjściowych służy do przetworzenia danych ("historii") pamiętanych w poszczególnych obiektach. Wstępne przetworzenie danych polega jedynie na zebraniu informacji z ww obiektów. Oprócz tego możliwe jest zapamiętywanie danych w pliku oraz generowanie raportów tekstowych i wykresów graficznych. Następnym etapem jest analiza porównawcza wybranych przebiegów symulatora. Przebiegi te mogą być wykonane w różnym czasie (dlatego potrzebne jest wspomniane wyżej zapamiętywanie danych w pliku) albo równolegle przez dwie instancje programu symulującego (należy pamiętać, że program jest przeznaczony do uruchomienia w wielozadaniowym systemie operacyjnym). Możliwe też jest zaprogramowanie wielu przebiegów symulacji ze zmieniającym się zadanym parametrem (np. w celu zbadania wpływu liczby palet na pracę systemu). Wówczas najodpowiedniejszym sposobem przedstawienia wyników jest wykreślenie wydajności systemu (lub wykorzystania zasobów) w funkcji tego parametru.

W celu dostarczenia użytkownikowi danych do prowadzenia własnych analiz możliwe jest przekazywanie danych do innych programów (np. arkuszy kalkulacyjnych).

5. Podsumowanie

Przedstawiony program symulacyjny RUNSIM został zaprojektowany w celu umożliwienia modelowania różnych typów elastycznych systemów produkcyjnych na etapie ich projektowania i eksploatacji. Moduł wprowadzania danych pozwala w łatwy sposób zdefiniować żądany FMS, a wymienność algorytmów szeregowania zadań umożliwia analizę efektywności decyzji na poziomie sterowania pracą systemu. Program został zaimplementowany na komputerze klasy PC. Wykazanie użyteczności programu dla celów przemysłowych wymaga intensywnych testów dla różnych parametrów wejściowych, co przy dużej możliwości zmian tych parametrów jest zadaniem pracochłonnym. Dalsze prace nad systemem RUNSIM będą obejmowały wyżej wspomniane testy oraz rozszerzenie zbioru algorytmów szeregowania zadań dostępnych w podstawowej wersji symulatora. Obserwacja

systemu sterowanego "na gorąco" może być bardzo ciekawym doświadczeniem dla użytkownika i dlatego warto skonstruować graficzny interfejs informujący o stanie poszczególnych elementów systemu. Bardzo ciekawym i pożytecznym doświadczeniem byłoby stworzenie dodatkowego modułu spełniającego funkcję programu śledzącego (debugger) w stosunku do zawartego w DLL algorytmu on-line. Dla twórców algorytmów sterujących mogłoby to być bardzo przydatne narzędzie, ułatwiające pracę nad tym ambitnym i trudnym do sprawdzenia oprogramowaniem.

LITERATURA

[1] Błażewicz J., Ecker K., Schmidt G., Węglarz J., Scheduling in Computer and Manufacturing Systems, Springer Verlag, Berlin, 1993

[2] Carrie A.S., Adhami E., Introducing FMSs by Simulation, *Working Paper*, University of Strathclyde, 1984

[3] Hurrion R.D. (ed.), Simulation. Applications in Manufacturing, Springer Verlag, Berlin, 1986

[4] Józefowska J., Piesik P., Węglarz J., RUNSIM - a Tool for Simulation of Flexible Manufacturing Systems, Proceedings of the Twelfth European Meeting on Cybernetics and System Research, Wiedeń, 1994 (przyjęte do druku)

[5] Ravi T., Lashkarl R.S., Dutta S.P., Selection of Scheduling Rules in FMSs - a Simulation Approach, *The International Journal of Advanced Manufacturing Technology*, 6, 1991, pp.246-262

[6] Stecke K.E., Design, Planning , Scheduling and Control Problems of Flexible Manufacturing Systems, *Annals of Opns. Res.*, vol.3, Baltzer, Basel,1985

[7] Suri R., Dille J.W., A Technique for On-line Sensitivity Analysis of Flexible Manufacturing Systems, *Annals of Opns. Res.*, vol.3, Baltzer, Basel, 1985

[8] Turbo Pascal for Windows: Users Guide, Programmers Guide, Windows Programming Guide, Borland International, Inc, 1991

[9] Swain J.J., Flexible Tools for Modelling, OR/MS Today, Dec. 1993.

[10] Voss A., Managing Advanced Manufacturing Technology, Springer Verlag, Berlin, 1986.

Recenzent: Prof. dr inż. Henryk Kowalowski

Wpłynęło do Redakcji do 30.04.1994 r.

Abstract

The design of Flexible Manufacturing Systems (FMS's) is a complex decision-making process. One of the possibilities to improve the quality of the design of FMS's is to use

simulation tools to test the system under various operating conditions. However, developing a dedicated simulation program requires much effort and is usually too costly to be applied effectively.

A simulation package RUNSIM designed for modelling and control of different FMS's is presented. These systems may differ in type (flexible manufacturing line, flexible manufacturing cell, FMS), in the storage system (central buffer, local buffers), in the transportation system (one- or multi-directional), in the system layout and in the availability of alternative processing plans. The modular structure of the program enables an easy description of a FMS using the input data. This way of defining a production system makes it possible to change the modelled systems without changing the program code. Consequently a substantial decline in the simulation costs is achieved.

Another characteristic of the presented simulation program is the possibility of defining a system even if some data are not available. In this situation some default values are used, relaxing all possible constraints imposed by the undefined variables. It enables a stepwise definition of the system and running the simulation even in early stages of the design process. One of the postulates for RUNSIM was to make it a useful tool at all stages of the design and to simulate the operation of a possibly broadest range of FMS's. Consequently a flexible choice of scheduling algorithms has been introduced. The user may choose an arbitrary scheduling algorithm from a set of algorithms (e.g. SPT, LPT, EDD, etc.) or include an original compiled procedure.

The program has been implemented on IBM PC 386 in Borland Pascal 7.0 with Object Windows in the MS Windows 3.1. environment.