ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1998
Seria: AUTOMATYKA z. 124 Nr kol. 1390

Wiesław PIERZCHAŁA
Politechnika Krakowska

SKALUJĄCY SIĘ MODEL DO STEROWANIA PRZEPŁYWEM PRODUKCJI W ZAUTOMATYZOWANYM SYSTEMIE WYTWARZANIA

Streszczenie. W referacie przedstawiono oryginalną metodę sterowania przepływem produkcji w oparciu o *skalujący się model macierzowy* systemu wytwarzania. Model ten stanowi rozwinięcie używanego dotychczas *modelu macierzowego*: zachowując wszystkie jego zalety posiada szereg istotnych nowych cech, które zostały krótko opisane. Najważniejszą jest możliwość dynamicznej zmiany jego rozmiarów, co umożliwia zachowanie ciągłości sterowania niezależnie od zmian asortymentu produkcji.

SCALABLE MODEL FOR PRODUCTION CONTROL IN AUTOMATED MANUFACTURING SYSTEM

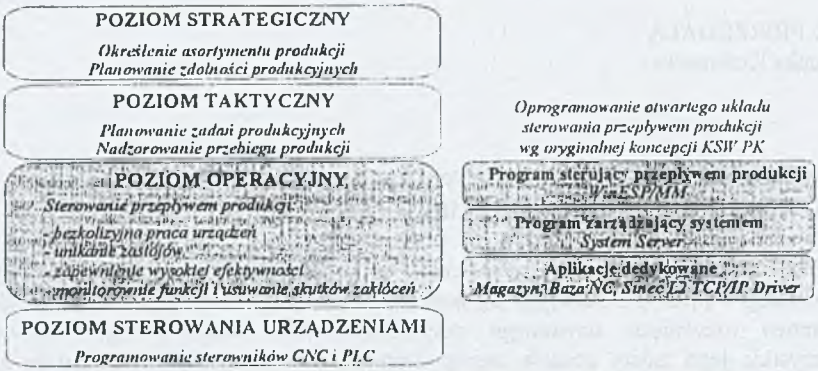
Summary. The paper presents the original method of automated production control, based on *scalable matrix model* of manufacturing system. Since all advantages of known *matrix model* have been maintained, the new features are described above all. The scalable model can be dynamically resized, so it enables the control apart from assortment changes.

1. Wprowadzenie

System zarządzania produkcją jest zazwyczaj postrzegany jako układ o strukturze hierarchicznej, w którym wyróżnia się co najmniej cztery poziomy przedstawione na rys. 1 wraz z głównymi zadaniami.

Efektem kilkuletnich badań prowadzonych w Katedrze Systemów Wytwarzania Politechniki Krakowskiej (KSW PK) było opracowanie i zweryfikowanie [2,3] oryginalnej metody komputerowego sterowania przepływem produkcji na poziomie operacyjnym, wykorzystującej model macierzowy (MM) [1] realizowanego procesu wytwarzania, przetwarzany współbieżnie z jego przebiegiem i decydujący o kolejności i terminach wykonywania poszczególnych czynności. Oprogramowanie metody ma strukturę

trójwarstwową (rys. 1) i składa się z szeregu aplikacji, wymieniających pomiędzy sobą informacje za pośrednictwem sieci Ethernet (protokół TCP/IP) [4].



Rys. 1. Hierarchiczna struktura systemu zarządzania produkcją
Fig. 1. Hierarchical structure of production management system

Łatwo zauważalną niedogodnością omawianej metody jest brak prostego i skutecznego sposobu wprowadzania do MM zmian asortymentu produkowanych równocześnie wyrobów, bez przerywania procesu wytwarzania. W referacie jest prezentowany *skalujący się model macierzowy* (SMM) systemu wytwarzania, umożliwiający płynny (bez przerywania sterowania) przepływ produkcji zautomatyzowanej, niezależnie od dynamiki zmian asortymentowych wynikających z bieżącego zapotrzebowania na wyroby. Model ten samoczynnie zmienia swoje rozmiary (wymiary macierzy) w przypadku wprowadzenia do produkcji nowych przedmiotów lub zakończenia (zaprzestania) produkcji któregoś z dotychczas wytwarzanych.

2. Idea modelu

Zadanie projektanta MM polegało na utworzeniu całego modelu w postaci zbioru macierzy, których kolumny odpowiadały obiektom systemu, a wiersze arbitralnie wyodrębnionym czynnościom elementarnym, które one realizują [1]. Zgodnie z nową, oryginalną koncepcją, projektant buduje bazę SMM złożoną z *komponentów modelu* reprezentujących poszczególne obiekty systemu wytwarzania (np. obrabiarki, palety, manipulatory) i opisujących ich funkcje.

W SMM wyróżnia się *obiekty stałe* oraz *obiekty przepływające*. Do pierwszej grupy zaliczają się wszystkie maszyny, urządzenia i inne elementy (takie jak np. palety), które stanowią integralną, niezmienną część struktury systemu (tzn. nie są do niego wprowadzane z zewnątrz i nie opuszczają go w trakcie jego działania), wykazując przy tym działanie cykliczne.

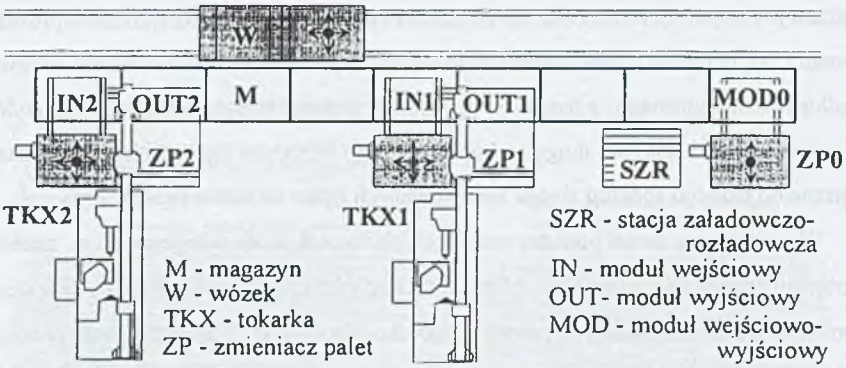
Obiektami przepływającymi z kolei są wszystkie te obiekty, które są do systemu wprowadzane z zewnątrz, są przemieszczane pomiędzy poszczególnymi stanowiskami według określonego porządku technologicznego, a następnie opuszczają system (najbardziej typowym przykładem są przedmioty obrabiane, ale mogą to być też palety, przyrządy itp.). Obiekty te muszą być przypisane do jednego spośród zbioru zdefiniowanych *typów obiektów przepływających*.

Wprowadzony został ponadto podział obiektów stałych na *operacyjne* (tzn. zmieniające stan obiektu przepływającego – np. obrabiarki, stacja załadowcza, myjnia), *przemieszczające* (zmieniające położenie obiektów przepływających – np. wózki, podajniki), oraz *bierne* (tzn. nie wykazujące samodzielnej aktywności – np. palety, magazyn regałowy). Obiekty bierne mogą być *wieloliczne*, tzn. każda kolumna *macierzy stanu* [1] modelu macierzowego może zawierać informację o stanie kilku identycznych obiektów (np. palet). Z kolei wszystkie *obiekty aktywne* (tzn. przemieszczające i operacyjne) mogą być tylko *jednoelementowe*.

Jak już wspomniano, *baza* SMM składa się ze zbioru *komponentów*, czyli obiektów systemu wraz z opisem ich działania. W bazie znajdują się wszystkie obiekty stałe, z których składa się system, oraz wszystkie przedmioty (i ewentualnie inne pomocnicze obiekty przepływające), które potencjalnie mogą być produkowane (wykorzystywane) przez system.

Funkcje obiektów stałych opisują tzw. *szablony działania obiektu*. Składają się one ze *wzorców czynności elementarnych*, w których obiekt bierze udział. We wzorcu czynności wszystkie uczestniczące w niej obiekty przepływające są zastępowane *symbolami* ich typów. Symbol taki ma postać `_typ_`. Do tak zapisanego wzorca *pasuje* każda czynność elementarna, która w miejscu symbolu typu obiektu przepływającego zawiera nazwę obiektu tego typu, wraz z opcjonalnym opisem jego *stanu* (w nawiasach prostokątnych). Na przykład, w modelu systemu do obróbki wałków pokazanego na rys. 2, do szablonu `_pdm_,PAL1;IN1»ZP1` (pobranie palety `PAL1` z obiektami przepływającymi typu _pdm_ z modułu wejściowego `IN1` przy TKX1 na zmieniacz palet `ZP1`) pasują czynności `PO1[po SZR],PAL1;IN1»ZP1` (pobranie przedmiotu `PO1`) oraz `PO2[po SZR],PAL1;IN1»ZP1` (pobranie przedmiotu `PO2`), jeśli przedmioty `PO1` i `PO2` są obiektami przepływającymi typu _pdm_.

Szablon może również, oprócz wzorców, zawierać konkretne czynności elementarne, w których nie uczestniczą żadne obiekty przepływające. Na przykład, w szablonie dla wózka `W` może występować czynność `PAL1;M»W` (pobranie palety `PAL1` z magazynu `M` na wózek `W`).



Rys. 2. System produkcyjny TOR98
Fig. 2. Production system TOR98

Do każdego ze wzorców przyporządkowany jest *zbiór wzorców następników*, które opisują następstwo czynności elementarnych. *Następnikiem* czynności j jest czynność j_1, jeśli po zakończeniu wykonywania czynności j obiekt jest zdolny do rozpoczęcia czynności j_1.

Jako przykład poniżej przedstawiono szablon dla zmieniacza palet `ZP1` przy pierwszej stacji tokarskiej (↳ oznacza następnik wymienionej czynności).

```
START
                ↳     _pdm_,PAL1;IN1»ZP1
                ↳     _pdm_,PAL2;IN1»ZP1
_pdm_,PAL1;IN1»ZP1          (pobranie załadowanej PAL1 z IN1 na ZP1)
                ↳     _pdm_,PAL1;TKX1,ZP1*
_pdm_,PAL2;IN1»ZP1
                ↳     _pdm_,PAL2;TKX1,ZP1*
_pdm_,PAL1;TKX1,ZP1*        (obróbka przedmiotów z PAL1 na TKX1 z udziałem ZP1)
                ↳     _pdm_,PAL1;ZP1»OUT1
_pdm_,PAL2;TKX1,ZP1*
                ↳     _pdm_,PAL2;ZP1»OUT1
_pdm_,PAL1;ZP1»OUT1         (oddanie załadowanej PAL1 z ZP1 na OUT1)
                ↳     ZP1?
_pdm_,PAL2;ZP1»OUT1
                ↳     ZP1?
ZP1?                        (sprawdzenie gotowości ZP1)
                ↳     _pdm_,PAL1;IN1»ZP1
                ↳     _pdm_,PAL2;IN1»ZP1
```

Na podstawie szablonu działania obiektu algorytmicznie tworzony jest zbiór *wzorcowych cykli elementarnych* realizowanych przez ten obiekt. Poniżej podano wzorcowe cykle elementarne wyznaczone dla zmieniacza palet `ZP1` na podstawie powyższego szablonu.

```
Cykl 1:   _pdm_,PAL1;IN1»ZP1        Cykl 2:   _pdm_,PAL2;IN1»ZP1
          _pdm_,PAL1;TKX1,ZP1*                  _pdm_,PAL2;TKX1,ZP1*
          _pdm_,PAL1;ZP1»OUT1                   _pdm_,PAL2;ZP1»OUT1
          ZP1?                                  ZP1?
```

Szablon opisuje więc wszystkie możliwe cykle pracy obiektu, ale w oderwaniu od konkretnych obiektów przepływających, które mogą się w tych cyklach pojawić. W rzeczywistości, w przypadku stworzenia konkretnego modelu, w którym występuje określony zbiór obiektów przepływających przyporządkowanych do poszczególnych ich typów, każdy obiekt będzie posiadał zbiór *rzeczywistych cykli elementarnych*, złożonych z konkretnych czynności elementarnych (a nie – jak powyżej – z szablonów). Istotny jest jednak fakt, że każdy z tych rzeczywistych cykli odpowiada dokładnie jednemu cyklowi wzorcowemu (tzn. każda z jego czynności *pasuje* do odpowiedniego wzorca w cyklu wzorcowym).

Z każdym obiektem przepływającym związana jest lista czynności elementarnych, opisująca przepływ obiektu przez system. Każda z tych czynności musi pasować do jednego wzorca w szablonie każdego uczestniczącego w niej obiektu stałego. Na przykład, jeśli na liście czynności przedmiotu `PO2` figuruje czynność `PO2[po SZR],PAL1;IN1»ZP1`, to wzorzec `_pdm_,PAL1;IN1»ZP1`, do którego ta czynność pasuje, musi wystąpić w szablonach obiektów `PAL1`, `IN1`, oraz `ZP1`.

3. Unikanie zastojów i przestojów

Zastój jest takim stanem systemu, w którym nic nie da się zrobić. Przykładem ilustrującym tę prostą definicję jest rozpoczęcie czynności pobrania kolejnej palety z magazynu w celu wypełnienia jej na stacji `SZR` półfabrykatami (`PAL1;M»W`) w sytuacji gdy `MOD0` jest zajęty. Wtedy wózek nie może się pozbyć palety ani też - ponieważ jest zajęty - opróżnić miejsca, na które powinien ją złożyć (w systemie jest tylko jeden wózek). *Przestój* z kolei powstaje wtedy, gdy zabraknie półfabrykatów w wymaganym stanie. Rozróżnienie tych dwóch niepożądanych stanów jest uzasadnione tym, że przestój, jeśli wystąpi, jest łatwy do usunięcia: wystarczy w jakikolwiek sposób dostarczyć półfabrykaty w wymaganym stanie do właściwych miejsc i system będzie kontynuował swoje działanie. Odmiennie, zastój wymaga specjalnych działań, np. wycofania się wózka z cyklu, który nie mógłby się zakończyć. Trudny problem zapobiegania zastojom i przestojom jest w SMM jeszcze bardziej złożony, ponieważ trzeba podać metodę, która będzie skuteczna także po przeskalowaniu modelu.

Istota opracowanej metody unikania zastojów i przestojów w SMM polega na tym, że pozwala się na rozpoczynanie tylko tych cykli elementarnych, które (wyłączając sytuacje awaryjne) mogą się zakończyć. Uzyskuje się to poprzez definiowanie, dla cykli elementarnych obiektów przemieszczających, stosownych *warunków dopuszczalności*. Każdy taki warunek

jest interpretowany w następujący sposób: cykl obiektu może być rozpoczęty, jeśli *obiekt warunkujący* jest zdolny do wykonania *czynności warunkującej*. Istotną cechą metody jest fakt, że warunki są zadawane dla cykli wzorcowych (tzn. wyznaczonych na podstawie szablonów). Podczas przetwarzania modelu każdy warunek obowiązuje dla wszystkich cykli rzeczywistych pasujących do cyklu wzorcowego, dla którego został on napisany.

W przypadku warunków zapobiegania zastojom obiektem warunkującym może być dowolny obiekt stały. Zdefiniowanie warunku polega wówczas na wybraniu *wzorca warunkującego* z listy wszystkich wzorców czynności elementarnych obiektu warunkującego. Dla każdego cyklu rzeczywistego, dla którego obowiązuje ten warunek, wzorzec warunkujący jest przekształcany w *czynność warunkującą* w ten sposób, że w miejsce występujących w nim symboli typów obiektów przepływających wstawiane są odpowiednie obiekty przepływające występujące w cyklu (lub – jeśli w cyklu obiekty tego typu nie występują – brany jest pod uwagę każdy pasujący obiekt). W efekcie rozpoczęcie danego cyklu elementarnego jest *dopuszczalne*, jeśli obiekt warunkujący jest zdolny do rozpoczęcia czynności warunkującej.

W przypadku warunków zapobiegania przestojom, gdzie zazwyczaj chodzi o konkretne stany obiektów przepływających, *obiektem warunkującym* może być dowolny obiekt przepływający. Zdefiniowanie warunku polega wówczas na bezpośrednim wybraniu *czynności warunkującej* z listy wszystkich czynności elementarnych obiektu warunkującego. Tutaj warunek jest zatem przyporządkowany na stałe do konkretnego przedmiotu w bazie SMM. Podobnie jak poprzednio, także przy unikaniu przestojów, rozpoczęcie cyklu elementarnego jest *dopuszczalne*, jeśli obiekt warunkujący jest zdolny do rozpoczęcia czynności warunkującej.

Dla wybranego cyklu istnieje możliwość zdefiniowania więcej niż jednego warunku. W przypadku zapobiegania zastojom tworzony jest wówczas iloczyn logiczny (wszystkie warunki muszą być spełnione). Z kolei w przypadku zapobiegania przestojom, spośród warunków wybierany jest ten, którego obiekt warunkujący występuje w rozpatrywanym cyklu. Jeśli obiekt ten w cyklu nie występuje, warunki traktowane są alternatywnie.

A oto przykład warunku zapobiegania zastojowi wózka w wybranym cyklu wzorcowym:

Cykl wzorcowy:	Warunek:
`_pdm_,PAL1;M»W`	Obiekt warunkujący: `MOD0`
`_pdm_,PAL1;W»MOD0`	Wzorzec warunkujący: `_pdm_,PAL1;W»MOD0`
`W?`	

Podczas przetwarzania modelu, dla każdego cyklu rzeczywistego odpowiadającego powyższemu cyklowi wzorcowemu warunek jest interpretowany w pokazany niżej sposób:

Cykl rzeczywisty:	Warunek:
`PO1[po TKX],PAL1;M»W`	Obiekt warunkujący: `MOD0`
`PO1[po TKX],PAL1;W»MOD0`	Czynność warunkująca: `PO1[po TKX],PAL1;W»MOD0`
`W?`	

4. Budowa modelu

Konkretny model macierzowy budowany jest algorytmicznie na podstawie informacji zawartych w bazie SMM. W skład każdego kreowanego modelu wchodzą wszystkie obiekty stałe oraz wybrane przez operatora obiekty przepływające. Lista czynności zestawiana jest z szablonów działania obiektów stałych oraz list czynności obiektów przepływających, według zasady: najpierw do listy dodawane są wszystkie czynności nie będące wzorcami (a więc nie zawierające symboli typów obiektów przepływających), a znajdujące się w szablonach obiektów stałych (np.: `PAL1;M»W, TKX?`). Następnie do tworzonej listy czynności dodawane są kolejno listy czynności wszystkich obiektów przepływających, które mają być wytwarzane.

Końcowym etapem budowania SMM jest algorytmiczne wyznaczenie zbiorów następników, poprzedników i macierzy eliminacji dla wszystkich czynności i obiektów. W efekcie powstaje model, który może być przekształcany według znanych, sprawdzonych zasad opisanych dokładnie np. w pracy [1].

5. Skalowanie modelu

SMM umożliwia dynamiczne uwzględnianie zmian asortymentu aktualnie produkowanych wyrobów. Aby było to możliwe, przyjęto następujące założenia:

1. Baza modelu obejmuje wszystkie przedmioty, które potencjalnie mogą być produkowane w systemie. Ich liczba jest nieograniczona.
2. W przetwarzanym modelu znajdują się tylko te przedmioty, które są aktualnie wytwarzane.
3. *Przeskalowanie* modelu, tzn. dodanie doń nowych przedmiotów, których produkcja ma się rozpocząć, lub usunięcie przedmiotów zakończonych, jest możliwe w każdej chwili.
4. *Skalowanie* modelu realizowane jest algorytmicznie, bez przerywania symulacji lub sterowania, co pozwala zachować ciągłość pracy systemu.

Skalowanie polega na zmianie wymiarów macierzy modelu. W przypadku usuwania przedmiotów z modelu znikają odpowiadające im kolumny, a także wszystkie czynności, w których biorą one udział. Z kolei w przypadku dodawania przedmiotów, w modelu pojawiają

się dodatkowe kolumny reprezentujące nowe przedmioty, a lista czynności jest odpowiednio poszerzana, po czym przeskalowanemu modelowi nadawany jest stan dokładnie odpowiadający aktualnemu stanowi systemu wytwarzania w jego modelu dotychczasowym.

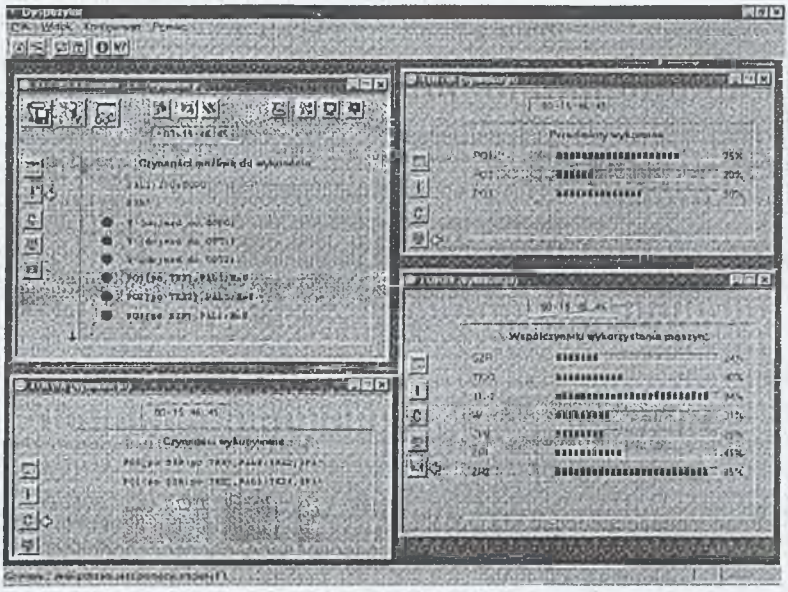
6. Oprogramowanie modelu

Program **Dyspozytor** do symulacji i sterowania przepływem produkcji, oparty na koncepcji skalującego się modelu macierzowego, jest aplikacją 32-bitową, przeznaczoną do pracy w środowiskach *Microsoft® Windows® 95* oraz *Windows NT®*. Wykorzystuje on udostępnianą przez te systemy operacyjne możliwość uruchamiania w ramach jednego programu kilku *wątków*, z których każdy może realizować inne zadanie, niezależnie od np. obsługi interfejsu użytkownika. W przypadku programu **Dyspozytor** istnieje możliwość jednoczesnego przetwarzania kilku modeli przedstawianych w oddzielnych oknach.

Ta funkcjonalność pozwala także na łatwą i wygodną implementację *procesów wirtualnych* uruchamianych w nowych wątkach tej samej kopii programu. Proces wirtualny symuluje działanie systemu wytwarzania począwszy od jego aktualnego stanu. Proces taki należy rozumieć jako mogący zaistnieć przebieg zdarzeń w rzeczywistym systemie. Uruchamiając proces wirtualny w czasie oczekiwania na zakończenie trwających czynności, można np. sprawdzić skuteczność przyjętej heurystyki sterowania.

Jednym z modułów programu **Dyspozytor** jest **Modelarz**, służący do wprowadzenia bazy modelu: obiektów i ich szablonów działania, obiektów przepływających z ich technologią zapisaną w postaci listy czynności elementarnych wraz z technologicznymi następnikami każdej z nich, oraz zbioru warunków zapobiegających zastojom i przestojom.

Okno modelu, w którym podawane są informacje o przebiegu procesu, może się znajdować w kilku trybach wyświetlania: czynności możliwych do wykonania, czynności wykonywanych, stanu zaawansowania produkcji, oraz współczynników wykorzystania maszyn. Dla każdego przetwarzanego modelu (niezależnie od tego, czy jest on przetwarzany w trybie symulacji, sterowania, czy też jako proces wirtualny) istnieje możliwość otwarcia jednego, lub kilku *okien podglądowych* w różnych trybach wyświetlania (rys. 3).



Rys. 3. Program *Dyspozytor* podczas przetwarzania modelu
Fig. 3. Program *Dispatcher* during model processing

7. Zakończenie

Program **Dyspozytor** oparty na SMM, na każdym etapie działania sterowanego systemu wytwarzania, wyznacza możliwie najliczniejszy podzbiór dopuszczalnych czynności elementarnych, pozostawiając swobodę wyboru tej, która zostanie rozpoczęta w pierwszej kolejności. Może pracować jako:

- program symulujący działanie systemu wytwarzania,
- program sterujący systemem wytwarzania niezależnie od uruchamianego w tle *procesu wirtualnego* (nawet w kilku wersjach jednocześnie),
- program sterujący rzeczywistą częścią systemu wytwarzania i symulujący jednocześnie działanie nieistniejących jego modułów.

LITERATURA

1. Cyklis J., Pierzchała W.: Modelowanie procesów dyskretnych w elastycznych systemach produkcyjnych. Zeszyty Naukowe Politechniki Krakowskiej, Mechanika z. 77, Monografia nr 3, Kraków 1995.
2. Cyklis J., Pierzchała W., Zając J. i in.: Opracowanie dyskretnego modelu przepływu informacji w systemie sterowania produkcją zautomatyzowaną. Raport końcowy z

realizacji projektu badawczego Nr 7 T07D 026 08, ITMiAP, Politechnika Krakowska, Kraków 1997.

3. Pierzchała W.: Modernizacja układu sterowania elastyczną linią obróbkową. Materiały Konferencji AUTOMATION'98, PIAP, Warszawa 1998, s. 287-294.
4. Pierzchała W.: Otwarty układ sterowania zautomatyzowanym wytwarzaniem. Materiały Konferencji AUTOMATION'98, PIAP, Warszawa 1998, s. 94-101.

Recenzent: Dr hab.inż. Mirosław Zaborowski, prof.Pol.Śl.

Abstract

The original method of automated production control was developed as a effect of research, which has been realised since 1990. In this method all decisions (concerning the sequence of performed actions) are based on *matrix model* of production system. The matrix model is processed real-time, simultaneously with production process. This method has one significant inconvenience: there is no easy way to change the assortment of simultaneously manufactured products.

In this paper **the scalable matrix model (SMM)** is briefly presented. It enables dynamical assortment changes during the production system control. Scalable model can change its size (by resizing all its matrixes and data structures) automatically when a product is finished or the production of new product begins. Therefore, any assortment change can be taken into account without the necessity of interruption of production process.

The scalable model (on each stage of production) computes set of all activities, which are *possible* to perform. The selection of activity, which will be performed first, is realised by one of available heuristic rules.

The **Dispatcher** is the program for automated production control based on scalable matrix model. It is 32-bit application, which was developed for *Microsoft® Windows® 95* and *Windows NT®* operating systems. It supports *multithreading* so each model processing is realised in separate thread. Therefore, simultaneous simulation of several different models is possible (every model has its own window). **Dispatcher** simulates the work of not existing modules simultaneously with control of real part of the production system.