

А

ВТОМАТИКА

И

ЗЧИСЛИТЕЛНА

Т

ЕХНИКА

И

1986

8 / 9

А

ВТОМАТИЗИРАНИ

С

ИСТЕМИ

СЪДЪРЖАНИЕ

АВТОМАТИКА

- И. Попчев, В. Монов, С. Савов — Децентрализирано управление на активната мощност и честотата при взаимосвързани електроенергийни системи 8
- Разглежда се проектирането на оптимални децентрализирани регулатори за управление на активната мощност и честотата в електроенергийна система. Синтезиран е децентрализиран наблюдател на неизмеримите състояния на системата, а за решаване на оптимизационната задача е използван метод за параметрична оптимизация с ограничения в структурата на регулатора. Представени са резултатите от цифровото моделиране на енергийната система, състояща се от две локални подсистеми.
- Н. Ишханян — Моделиране на един клас нелинейни системи за автоматично управление 19
- Разглежда се използването на алгоритмите на Розенброк и Гир за целите на моделирането на един клас нелинейни системи, описвани с диференциални уравнения с прекъсвания. Дадена е програмна реализация. Решен е пример с цел сравняване на двата метода.
- Г. Вачков — Диагностика на отказите в технологични системи при размитост на информацията 25
- Предложен е метод за диагностика на отказите, валиден за общия случай на няколко едновременно действащи отказа. Методът се основава на подходящ диагностичен модел, в който причинно-следствените връзки между отказите и симптомите са представени размито. Алгоритмично и програмно са решени задачите за диагностика и оптимален избор на симптоми от гледна точка на диагностиката. Разработените алгоритми и програми могат да служат за основа при създаване на автоматизирани мониторинжни системи за диагностика с практическо приложение в промишлеността.
- Б. Боянов, О. Аспарухов, Т. Иванов — Метод за изчисляване на периода на основния тон посредством автокорелационна функция 38
- Предложен е метод за изчисляване на периода на основния тон от зашумени и фазово изкривени записи на говорен сигнал посредством автокорелационна функция. Извършено е експериментално изследване на точността на разработения метод.

ИЗЧИСЛИТЕЛНА ТЕХНИКА

- В. Събев, А. Ескенази — Моделиране на елементи от производството на програмно осигуряване чрез мрежи на Петри 44
- Разглеждат се моделите на отделните фази на жизнения цикъл на програмните продукти с използването на мрежи на Петри. Въз основа на тях е формиран общ модел на фазата използване с цел създаване на информационна система, автоматизираща дейности от съпровождането и поддържането на продукта.
- Е. Димитров — М-мрежите — графично ориентирани средства за моделиране и изследване на изчислителни и операционни системи 53
- Разглеждат се основните проблеми при моделирането на сложни комуникационни системи и годността на различните М-мрежови модели. Описани са някои модели и методи за машинната им обработка.
- Б. Янков, Л. Николов — Език за системно и приложно програмиране на микропроцесорни системи 59
- Описва се език за микрокомпютри и микропроцесорни системи, наречен MBPL. Езикът работи с обекти, характерни за 8-битовите микропроцесори — символи, еднобайтови константи и адреси. Конструкциите на езика дават някои възможности за работа, присъщи на асемблерните езици. MBPL използва инструкции, съответстващи на езиците от високо ниво. Езикът е реализиран по два пътя — на основата на асемблерния език на изборния микропроцесор и на основата на виртуален процесор. Вторият вариант позволява преносимост на разработеното програмно осигуряване върху всички типове 8-битови микропроцесорни системи.
- З. Марков, Д. Дочев, Х. Дичев, Г. Агре — Езикът ПРОЛОГ — средство за създаване на интелигентни системи за машинна графика 67
- Обсъдени са съвременните тенденции в развитието на машинната графика и проблемите при използването на технологията за обработка на знания и разработване на интелигентни системи за машинна графика. Разгледани са възможностите на езика за логическо програмиране ПРОЛОГ за

МОДЕЛИРАНЕ НА ЕЛЕМЕНТИ ОТ ПРОИЗВОДСТВОТО НА ПРОГРАМНО ОСИГУРЯВАНЕ ЧРЕЗ МРЕЖИ НА ПЕТРИ

УДК 519.9:681.3

В. Събев, А. Ескенази

УВОД

Във всички модели на жизнения цикъл на програмните продукти под едно или друго име фигурира фаза (етап), отразяваща използването на продукта и присъщите на това използване процеси. Обикновено като най-съществени през този етап се сочат отстраняването на грешки, разширяването на възможностите на продукта, подобряването на някои характеристики. Тези процедури изискват задълбочено познаване на продукта, достъп до пълната документация и висока професионална квалификация. Терминът „съпровождане“ се използва като обединяващ за посочените процедури. Важни са и връзките между потребителя и производителя (респ. разпространителя) при инсталирането на продукта и всяка негова версия, първоначалното и по-нататъшното обучение на потребителя, обмена на информация, свързана с експлоатацията на продукта. Тези пък процедури се обединяват от термина „поддържане“. Очевидно поддържането поставя по-различни изисквания към изпълнителите му в сравнение със съпровождането. Независимо от това функциите съпровождане и поддържане са свързани и фактически осигуряват ефективното използване на програмния продукт.

Ясно е, че за да се повиши тази ефективност, трябва да се създадат и съответни средства. Те могат да имат методологичен и организационен характер или да представляват конкретни програми, автоматизиращи разглежданите функции (последното изглежда съвсем естествено за един производител на програмно осигуряване). В това направление беше и реализацията, отразена в [1] — след построяването на подходящ модел се създава информационна система, автоматизираща дейности от съпровождането и поддържането на програмния продукт.

ОБЩА ПОСТАНОВКА

Естествено колкото по-обхватен и всеобхватен е един модел, толкова по-ясно и задълбочено ще бъде разбирането за моделирания обект. Доколкото в [1] целта беше създаване на автоматизирана информационна система, моделът бе ориентиран преди всичко към отразяване на структурни елементи (програми, изпълнители, за-

дачи, работи, файлове, срокове и др.) и връзките между тях. Обаче процесите, изпълнявани върху тези обекти или пораждащи ги, както и условията за тяхното инициализиране засега не са изведени до явна форма поради липса на необходимост от това.

По тези съображения бе направен опит за изграждане на по-пълен модел на функциите съпровождане и поддържане с подчертаване на съставлящите ги процеси и условия.

При така поставената задача първият проблем е изборът на апарат за моделиране. Съображенията бяха:

на моделиране подлежат процеси и условията за изпълнението им;

част от процесите (напр. откриване на грешки в продукта, поява на желание за усъвършенстване и др.) имат случайна природа;

голяма част от процесите са асинхронни;

някои процеси могат да бъдат паралелни (едновременно или почти едновременно постъпване на заявки за корекции от няколко потребителя, инсталиране на версия у няколко потребителя, паралелно изменение от различни програмисти на няколко модула от един продукт и др.).

При такива предпоставки най-естествен за моделиране се оказва апаратът на мрежите на Петри [2, 3] със следния план на действие:

1. Идентифициране на процесите и условията.

2. Свързването им в елементарни мрежи.

3. Интегриране в обща мрежа.

4. Изследване на свойствата на тази мрежа.

5. Предлагане на конструктивни изменения.

В настоящата работа са дадени резултатите от (1), (2) и (3) и се предлагат съображения по (4).

МОДЕЛ НА ФАЗАТА ИЗПОЛЗУВАНЕ

Обикновено в даден момент от фазата използване от жизнения цикъл програмният продукт може да се разглежда като крайно непразно множество от модули. Може да се приеме, че това множество се изменя динамично (например при добавяне на нови модули с цел разширяване на възможностите на продукта). По тази причина първоначално ще се разгледа случаят на неразширяване (неизменение), а след това въз основа на него ще направим обобщение на възможните случаи на разширяване.

Нека $PRG = \{M_1, M_2, \dots, M_m\}$, $1 \leq m < \infty$, означава програмен продукт, където M_i , $i = 1 + M$, са неговите модули. Непразни подмножества на PRG образуват различните му варианти (конфигурации). Те могат да съществуват относително самостоятелно. Най-разпространеният вариант е този, който се състои от всичките модули. Нека с VAR се означава множеството от варианти, т. е. $VAR = \{V_1, V_2, \dots, V_p\} \in P(PRG) \setminus \emptyset$, където $V_k = \{M_i | i \in I_k\}$, а $I_k \neq \emptyset$ е множество, чиито елементи са индекси на модулите, принадлежащи на k -тия вариант, $k = 1 \div p$. Множеството $USR = \{C_1, C_2, \dots, C_c\}$, $1 \leq c < \infty$, означава множеството на потребителите (клиенти) на програмата. При това всеки клиент разполага само с един екземпляр от даден вариант, който той експлоатира. Множествата $USR_k = \{C_i | i \in J_k\}$, $k = 1 \div p$, показват потребителите на съответните варианти на програмата, където J_k са множества, чиито елементи са индексите на отговарящите потребители. Очевидно е, че при използваем вариант k , $J_k \neq \emptyset$.

МОДЕЛ НА ЕДИН ВАРИАНТ

Съпровождане. За отразяване на действията, изпълнявани при съпровождане на програмен продукт, са въведени елементарните мрежи $C\Gamma_{kj}$ и $RC\Gamma_{kj}$, отнасящи се до

k -тия вариант и j -тия модул, където $j \in I_k$ и $k = 1 \div p$. Елементите от първата група (Cr) означават изменение на модулите с цел подобряване на техните характеристики или пък отстраняване на грешки в тях. Елементите от втората група (RCr) служат за отразяване на случаите на заявяване на изменения. На фиг. 1 графично е представена мрежа на Петри за съпровождане на k -тия вариант, имаща следната алгебрична формула:

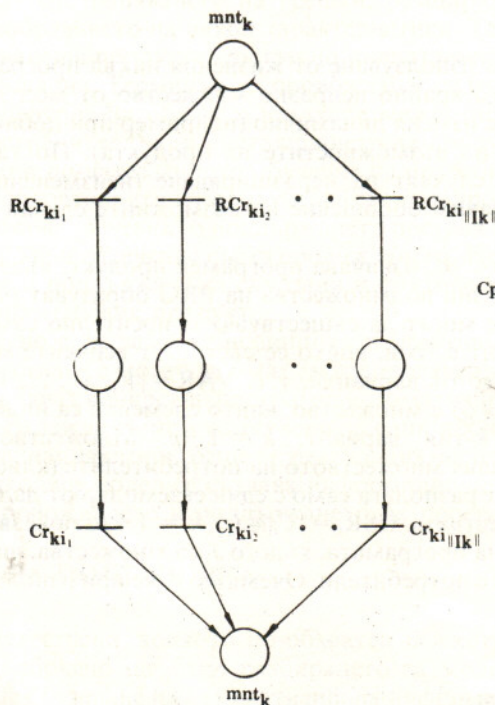
$$MNT_k = ((RCr_{ki_1}; Cr_{ki_1}) \dots (RCr_{ki_{\|I_k\|}}; Cr_{ki_{\|I_k\|}})),$$

където $i_j \in I_k, j = 1 \div \|I_k\|, k = 1 \div p$.

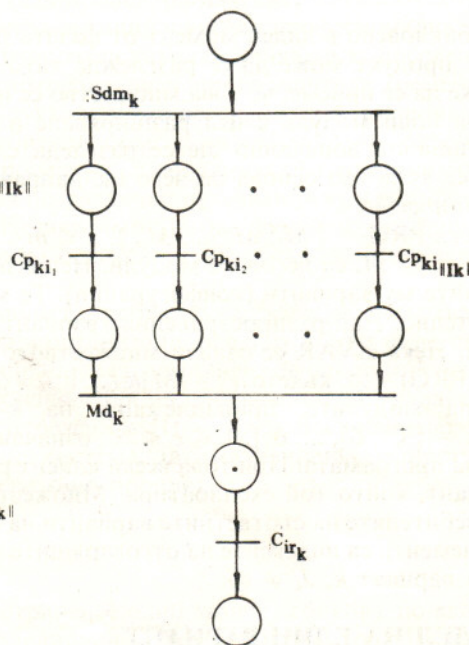
Нека е дадена мрежата MNT_k , която е от този вид, и от нея образуваме $MNT = 1MNT_k$, т. е. в началната позиция има ядро. В практиката това може да бъде заявка за коригиране на вариант. Алтернативният начин на образуване на MNT ще доведе до задействане на само един преход $RCr_{ki_0}, i_0 \in I_k$ и след него на Cr_{ki_0} , което от своя страна означава заявяване на изменение и самото изменение на модула M_{i_0} . В резултат на задействанията на посочените преходи (изпълняване на действията) в крайната позиция на MNT ще се появи ядро, изразяващо условие, че е настъпило изменение в един от модулите на варианта.

Поддържане. Включените елементарни мрежи за означаване на действията, извършвани при поддържане на програмния продукт, условно ще разделим на две групи. Първа група се отнася до създаването и разпространяването на готов за експлоатирание вариант. Към втората група са включени елементарните мрежи, отговарящи на онези действия от съпровождането, които като правило се осъществяват при клиента.

Действията от първата група включват елементарните мрежи Sdm_k, Md_k, Cir_k и Cr_{ki} , валидни за k -тия вариант, $k = 1 \div p$. Първите три означават съответно начало



Фиг. 1



Фиг. 2

на създаване, самото образуване и разпространяване на дистрибутивен носител за k -тия вариант, а последната — копиране на j -тия модул, необходим за създаването на k -тия вариант, т. е. $j \in I_k$.

На фиг. 2 графично е представена мрежа на Петри за създаването и разпространяването на k -тия вариант, имаща следния алгебричен вид:

$$SUP'_k = Sdm_k; (Cp_{ki_1}, \dots, Cp_{ki_{\|I_k\|}}); Md_k; !Cir_k;$$

където $i \in I_k, j = 1 \div \|I_k\|, k = 1 \div p$.

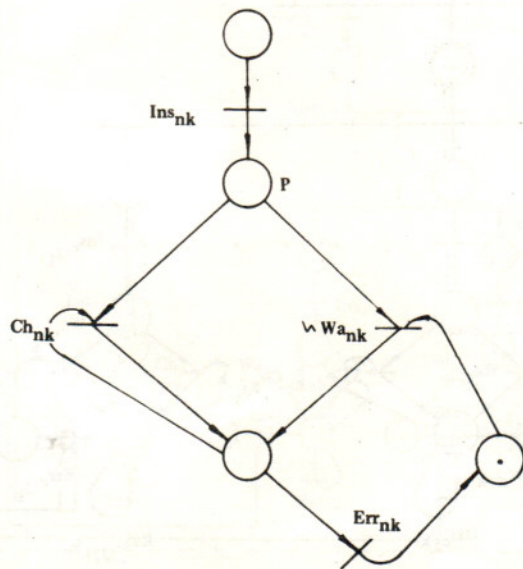
Да разгледаме мрежата $SUP = !SUP'_k$, където SUP'_k е мрежа от описания вид. В този момент разрешените преходи (възможни действия за изпълнение) са начало на създаване на дистрибутивен носител (Sdm) и разпространяване на дистрибутивен носител (Cir). В резултат от задействването на първия преход едновременно ще се разрешат и преходите по копиране ($Cp_{ki}, i = 1 \div \|I_k\|$), чието задействане (на всичките) ще доведе до разрешаване и задействане на прехода по създаване на носителя. Това от своя страна ще доведе до задействането на прехода, изразяващ разпространението на дистрибутивния носител. В резултат на това в крайната позиция на SUP ще има две ядра, показващи наличието на готови за разпространяване дистрибутивни носители за този вариант.

Действията от втората група са Ch_{nk} , Wa_{nk} , Ins_{nk} и Err_{nk} , отнасящи се до n -тия клиент, който използва екземпляр на k -тия вариант, където $n \in J_k$ и $k = 1 \div p$. Тези елементарни мрежи означават съответно замяна, изчакване на получаването на нов екземпляр, инсталиране и откриване на грешки. Мрежа на Петри за съпровождане на даден продукт при клиент е показана на фиг. 3. Тя има следната формула:

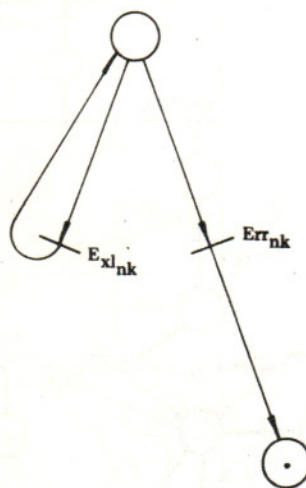
$$SUP''_{nk} = Ins_{nk}; (Ch_{nk} Wa_{nk}); (*Ch_{nk}); (* (Err_{nk}; !Wa_{nk}));$$

където $n = 1 \div \|J_k\|, k = 1 \div p$.

Нека е дадена мрежата $SUP = !SUP''_{nk}$, където SUP''_{nk} е от този вид. Първоначално



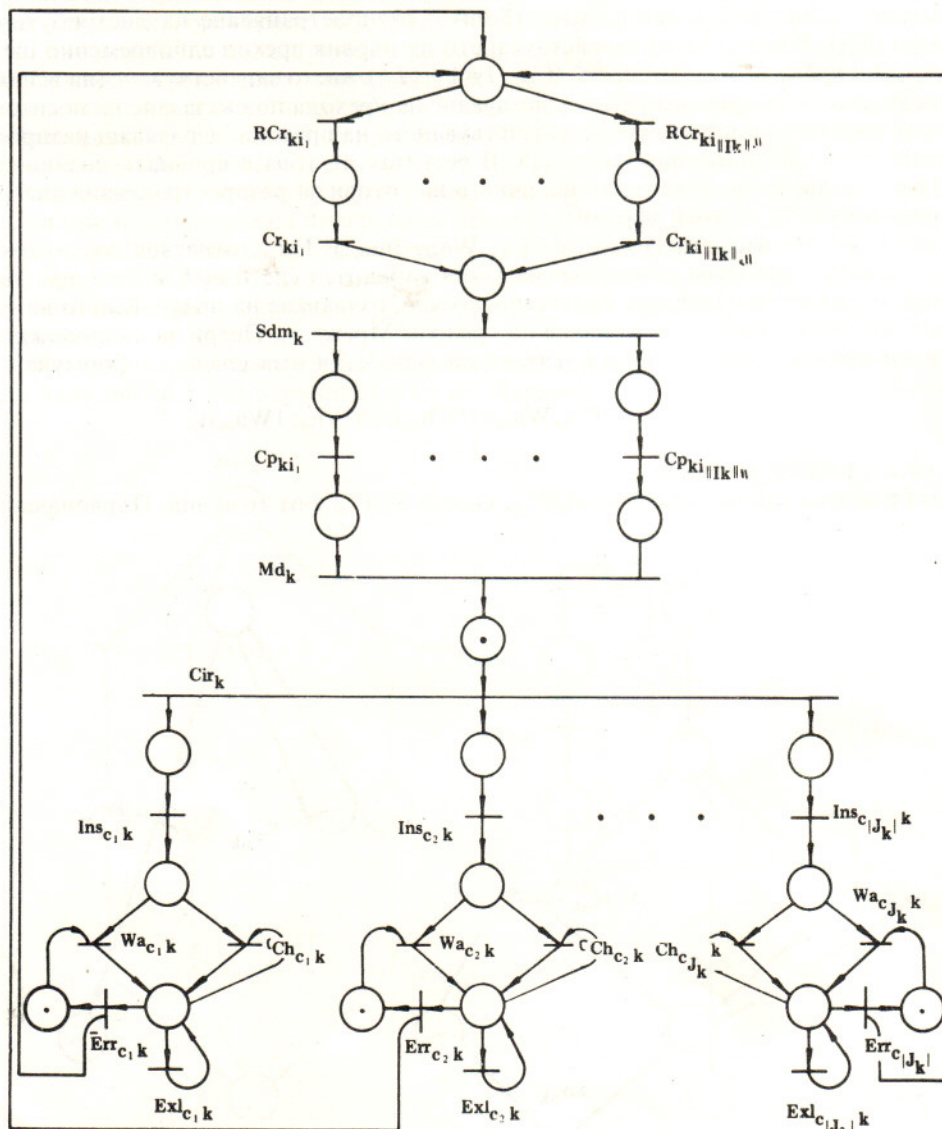
Фиг. 3



Фиг. 4

е разрешен само преходът, изразяващ инсталирането на получен екземпляр на варианта. Неговото задействуване ще доведе до задействуването на Wa_{nk} , чийто резултат показва, че съществува готов за използване екземпляр, т. е. в крайната позиция на SUP" има ядро. В този случай единствен преход, който може да се задействува, е Err_{nk} , като след неговото задействуване всички преходи са мъртви. Това може да се свърже с факта, че е открита грешка и се изчаква коригиране на варианта.

Експлоатиране. Наред със съпровождане и поддържане във фазата използване има още едно действие — експлоатиране на екземпляра на варианта от самия клиент. За целта се въвеждат елементарни мрежи Exe_{nk} , означаващи изпълнение на екземпляр на k -тия вариант от n -тия клиент, където $n \in J_k$ и $k = 1 \div p$.



Фиг. 5

Действията на клиента са представени на фиг. 4. Алгебричната формула на това представяне е:

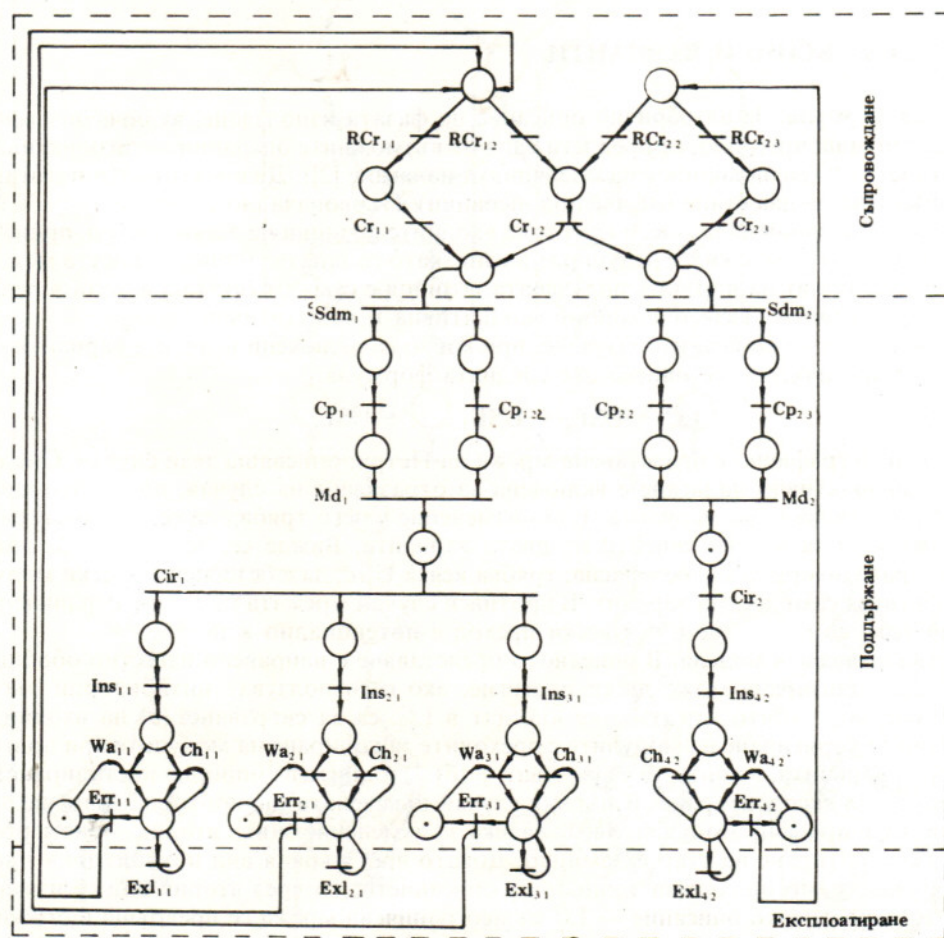
$$EXP_{nk} = *E_{x_{nk}}; E_{g_{nk}},$$

където $n = 1 \div \|J_k\|$, $k = 1 \div p$.

Да разгледаме мрежата $EXP = 1EXP_{nk}$. В нея са разрешени двата ѝ прехода — изпълнение на екземпляр и откриване на грешки. Те са алтернативни. Това положение се запазва, докато $E_{g_{nk}}$ не се е задействувал, т. е. открита е грешка в екземпляр. След неговото задействуване двата прехода са мъртви, а това означава, че при клиента няма готов за експлоатиране вариант.

Формула на вариант. Въз основа на изложеното дотук вече може да се даде алгебрична формула, представляваща общо описание на фазата използване за един вариант. Това описание е следното:

$$USE_k = *(MNT_k; SUP'_k; (CLT_{c_1 k}, \dots, CLT_{c_{\|J_k\|} k})),$$



Фиг. 6

$$CLT_{ij} = \text{SUP}_{ij}''; \text{EXP}_{ij}, c_j \in J_k, j = 1 \div \|J_k\|, k = 1 \div p.$$

Графично това описание е представено на фиг. 5. Ако се проследи задействването на преходите в USE_k , се забелязва, че в резултат на задействване на Cig_k се получава ISUP_{nk}'' за всеки клиент на този вариант. Това от своя страна води до следните алтернативни случаи — готов за експлоатиране вариант и откриване на грешка. От итеративния характер на USE_k следва, че при откриване дори на една грешка (задействване на Erg_{nk}) се получава $r\text{MNT}_k$, $r \geq 1$, като последното води до ISUP_{nk}' , а след това и до ISUP_{nk}'' , $i \in J_k$. За клиенти, които са предизвикали изменения (открили са грешка), ще се задействуват W_a , а за другите — или ще се получи натрупване в P (от фиг. 3), или ще се задействува Ch (което означава замяна на стария екземпляр с нов).

В зависимост от споменатото натрупване в P не е трудно да се покаже, че така получената мрежа е безопасна при $\|J_k\| = 1$ и неограничена при $\|J_k\| > 1$. Всеки преход е потенциално жив, което означава, че за всяко действие съществува поредица от осъществени действия (задействувани преходи), които довеждат до изпълнението му.

МОДЕЛ ЗА ВСИЧКИ ВАРИАНТИ

Обща формула. За алгебрично описание на фазата използване, включваща всички варианти на продукта, се предлага една от възможните операции — дизюнктивно налагане „+“ (разновидност на операцията налагане [2]). Дизюнктивното налагане се изпълнява на два етапа (от две подоперации). Първоначално се осъществява класическата операция налагане, а след това входните позиции на всеки преход, принадлежащ и на двете мрежи, се редуцират в една, като се запазва инцидентността между преходите. Броят на ядрата в получената позиция е сума от броя на ядрата в редуцираните позиции. Тази операция е асоциативна и комутативна.

Общият вид на фазата използване, при който са включени всичките варианти на програмния продукт, се описва със следната формула:

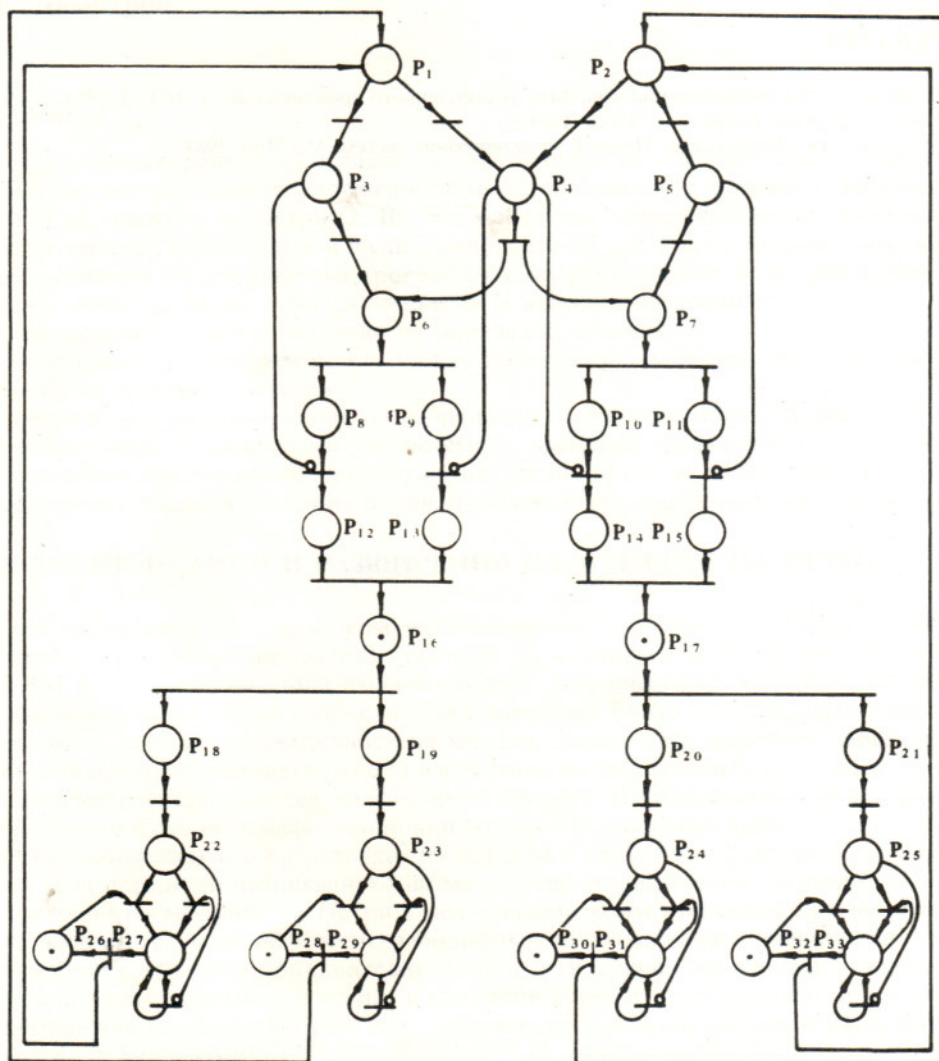
$$LC = \text{USE}_1 + \text{USE}_2 + \dots + \text{USE}_p.$$

На фиг. 6 графично е представена мрежа на Петри, описваща тези случаи. Операцията дизюнктивно налагане е включена за отразяване на случая, при който един модул участва в два варианта. Едно изменение в него трябва да се отразява (разпространява) за всички клиенти на двата варианта. Вижда се, че за да бъде така дефинираната мрежа LC безопасна, трябва всяка USE_k да е безопасна и всеки модул да участва само в един вариант. В противен случай мрежата не е дори ограничена. Освен това се вижда, че в нея всеки преход е потенциално жив.

Детайлизиране в модела. В описаното представяне е направено известно обобщаване. Детайлизиране може да се постигне, ако се използват инхибиторни дъги. Подобни дъги, които могат да се въведат в LC , са за свързване: (а) на входните позиции за коригиране на модулите с преходите за копиране на модулите при подготовка на варианта, в който те участвуват, и (б) изходните позиции на инсталирането с прехода за експлоатиране. В първия случай въведената инхибиторна дъга ще забранява копиране на модул, ако има заявка за изменение, а във втория — ще забранява експлоатиране на стар екземпляр. Докато чрез първия вид инхибиторни дъги се получава само по-голяма точност на описанието, то чрез втория вид се изменя и качествено самото описание — LC от неограничена мрежа се превръща в ограничена. Наличието на маркер в P_i , $i = 22 \div 27$ (от фиг. 7) ще доведе до изпълнение на Ch — ако съответният клиент работи със стар екземпляр, защото в този момент

Ехе не е разрешена (от инхибиторната дъга), или на W_a — ако съответният клиент е активизирал изменение във варианта.

Версии на програмата. Освен разглежданите случаи при съпровождането на един програмен продукт може да се разширяват възможностите му — чрез коригиране на съществуващ модул от програмата или чрез добавяне на нов модул. Първият случай е вече разгледан при модела на един вариант. Вторият случай води до „количествени“ изменения, отнасящи се до MNT_k и SUP'_k , $k = 1 \div p$, които при тези изменения остават от същия вид, т. е. новополучените мрежи са от същия клас и за тях са валидни показаните свойства. В някои източници подготвянето на нов модул се свързва с получаването на нова версия на програмния продукт. Това не противоречи на изложеното дотук, тъй като новата версия представлява също непразно множество



Фиг. 7

во от модули на програмния продукт, образуващо негов вариант със съответните си клиенти. Новополученият случай също може да се опише с мрежа от вида на LC, но вече с разширения на местата и преходите в нея и на връзките между тях (и отново са валидни показаните свойства).

ЗАКЛЮЧЕНИЕ

В предлаганата работа е създаден общ модел на фазата използване от жизнения цикъл на програмата. На основата на мрежите на Петри полученият модел позволи да се определят характерни свойства от фазата. Освен това той е предпоставка за имитационно изследване на протичането на процесите от фазата използване и за създаване на програмни средства (информационни системи и (или) друг инструментариум) за автоматизирането и чрез интерпретация на мрежите на Петри.

ЛИТЕРАТУРА

1. Ескенази. А. Автоматизация на една фаза от софтуерното производство. — АСУ, 3, 1984, 5—9.
2. Котов, В. Е. Сети Петри. М., Наука, 1984.
3. Питерсон, Дж. Теория сетей Петри и моделирование систем. М., Мир, 1984.