УДК 621.38(62-52)
О.В. Блажко, О.І. Лісовиченко, Є.С. Пуховський, Л.С. Ямпольський

МОДЕЛЮВАННЯ ПРОЦЕСІВ ВЗАЄМОДІЇ КОМПОНЕНТІВ КОМП'ЮТЕРНО-ІНТЕГРОВАНИХ СИСТЕМ ІЗ ЗАСТОСУВАННЯМ АПАРАТУ ПРОЦЕСНИХ АЛГЕБР

Вступ

При моделюванні КІВС розглядається як складна система, яка уявляє собою композицію окремих компонентів. Саме взаємодія компонентів КІВС і складає основний аспект моделювання системи і має забезпечувати функціонування останньої відповідно до вимог технологій, що в ній реалізуються.

Слід зазначити, що дискретно-подійні системи, до яких, безперечно, належать і КІВС [2], як часові моделі у загальному випадку є аналітично важко отримуваними і комп'ютерно-розрахунково не здійснюваними через багаточисельність раптово-випадкових подій на можливих непередбачуваних інтервалах станів, що розв'язується деякою мірою комбінаторно. Проте, процеси у КІВС, які підпорядковуються часосинхронізуючому впливу, добре моделюються подійно-часовими графами (ПЧГ) – своєрідними модифікаціями сіток Петрі (СП) [6,25], в яких часова затримка асоціюється з кожною позицією, що має тільки по одному вхідному та вихідному переходу [12,16,19]. Іншою модифікацією СП, створеною власне для моделювання динамічних перебігів подій у системі, є гнучкі СП [1,3,19]. Проте, для дослідження динамічної поведінки дискретно-подійних систем з часовим навантаженням можна з успіхом застосувати діоід- та макс-алгебри [13]. В останньому випадку дослідження КІВС може здійснюватися визначенням періодичності системних матричних властивостей і розрахунком їх вагових оцінок. Використовуючи інструментарій макс-алгебри, уявляється можливим, підраховуючи неконтрольовану поведінку ПЧГ-моделі, знаходити подання деякого нового бажаного режиму функціонування і визначити, чи може це подання бути реалізованим за допомогою набору контрольованих подій [4,8,10,21].

При виборі формального апарату для розв'язання поставленої задачі перш за все керуються притаманними особливостями і властивостями композиційних систем із сукупністю взаємодіючих між собою компонентів. Тому формальний апарат, який планується до використання при описі властивостей КІВС, має забезпечувати розвинені набори операцій композиції, компактне синтаксичне подання розрізнених та поєднаних системою компонентів [3]. При цьому виникає питання, якою моделлю користуватись для відбиття особливостей функціонування процесів у КІВС.

Моделі взаємодії

Для розв'язання означеної задачі необхідно вибрати модель, характеристики якої будуть відповідати особливостям реальної КІВС. Розглянемо основні типи існуючих моделей взаємодії, серед яких відзначимо *вкладену, однорівневу* та *мішану моделі* взаємодії [19]. Послідовно розглянемо їх основні характеристики.

Вкладена модель взаємодії. Вкладена модель базується на тому, що взаємодія між вкладеними потоками керування є спеціальним випадком синхронного зв'язку [24], який веде себе подібно процедурному виклику. Тобто мається на увазі, що компоненти, виконуючи запит, залишаються у блокованому стані доти, поки запит не буде задоволений. Ступінь невизначеності у вкладених моделях обмежена тим фактом, що компонент не сприймає зовнішні (чужі) події через очікування відповідної на попередній запит. Вкладені моделі характеризуються гнучкістю. Завдяки цьому спрощується аналіз систем на базі таких моделей.

Однорівнева модель взаємодії. Компоненти таких моделей проектуються відповідно до диспетчерського циклу. На відміну від попередньої моделі, запити можуть оброблятися, поки попередні операції ще не завершені. При цьому послідовність подій в системі стає більш випадковою, ніж у вкладених моделях. В однорівневій моделі можуть використовуватись різноманітні підходи до зв'язку. Однорівневою моделлю забезпечується безтупиковість станів та реагування системи на події, що є необхідною умовою для КІВС.

Однак, при тестуванні однорівневі моделі є більш громіздкими, ніж вкладені моделі. Це пояснюється тим, що задіяний компонент може мати прямі зв'язки із процесами, які ще знаходяться в стадії розвитку. Ще важче прогнозувати поведінку подійно-керованих систем, бо незалежні процеси можуть запускатися, конкуруючи в одному компоненті та непрогнозовано взаємодіючи між собою.

Проблема для семантичного моделювання однорівневих систем полягає в ідентифікації поведінок, що, припустимо, будуть створені взаємодіями компонентів [19].

Мішані моделі. Не зважаючи на недоліки однорівневої моделі, практика іноді потребує її використання в системному проектуванні. Щоб частково уникнути непрогнозованих взаємодій між процесами, що характерно для однорівневої моделі, використовують мішану модель. Вона характеризується застосуванням блокування ресурсів та транзакцій.

Означення 1. *Блокування ресурсів* – це стан системи, в якому два процеси, що використовують ресурси, блокують один одного через те, що перший процес захопив ресурс A і очікує звільнення ресурсу B, тоді як другий – захопив B і очікує звільнення ресурсу A. В такому стані системи обидва процеси виявляються взаємно заблокованими.

Означення 2. *Транзакція* – це перехід процеса з поточного до попереднього стану в точках синхронізації.

Взаємне використання механізмів транзакцій та блокування ресурсів

 ISSN 1562-9945

надає системі більш прогнозованої поведінки. Отже, блокування та транзакції в однорівневій системі може породжувати систему, що інколи виявляє властивості вкладених моделей. З іншого боку, багатопроцесність в системі, що має вкладений потік, може також породжувати поведінку, що характерна як для однієї, так і для другої моделі. Ось чому моделі з використанням зазначених механізмів називаються *мішаними* [19].

В загальному випадку при використанні однорівневого підходу до проектування системи необхідно визначити всі заблоковані стани. Щоб уникнути необхідності точного визначення усіх заблокованих станів, використовують багатопроцесність із застосуванням механізму транзакцій.

До переваг мішаної моделі відноситься підтримка розподіленого проектування, коли різні компоненти в системі проектуються різними людьми, які можуть використовувати як вкладену модель, так і однорівневу модель. Системи розподіленого проектування на сучасному етапі є достатньо поширеними та стрімко розвиваються.

Розгляд зазначених моделей дає можливість вибору, у відповідності до характеристик реальної КІВС, відповідної моделі та підходів у визначенні зв'язків компонентів системи.

Методи моделювання КІВС

Як вже було зазначено, для вирішення задачі моделювання використовуються підходи: методи імітаційного моделювання, теорії автоматів, марковських процесів, теорії черг, макс/мін-плюс алгебри, сітки Петрі, процесні алгебри. Наведемо узагальнену характеристику методів, окресливши основну специфіку задачі моделювання взаємодії компонентів КІВС: композиційність, синтаксичний опис процеса взаємодії, відображення внутрішньої будови КІВС (рівень компонентів та/або їх внутрішньої будови).

Теорія автоматів забезпечує розвинений апарат опису та аналізу переходів між станами системи, але не відображає причинно-наслідкові відношення. Крім того, композиція моделей та їх ієрархічне подання в межах цього апарату ускладнені [3].

Марковські процеси та *теорія черг* враховують стохастичний аспект поведінки системи і дають їх кількісні характеристики відносно певних проміжків часу чи наборів завдань, але вони абстрагуються від внутрішньої будови системи [3].

Алгебри (макс/мін,+) за суттю аналогічні з точністю до операції *max* або *min* і знаходять застосування в задачах розрахунку часу виконання визначених послідовностей дій в системі (якщо спостерігається досить регулярна структура, яка може бути поданою матрицями) і проходження потоку вздовж шляху (маршруту, для якого оцінюється тривалість), що може бути легко подане матричними операціями. Тоді, наприклад, деяка подія відбувається, "як тільки" (або "відразу ж, як") виконується визначена умова (тому "*min*"); аналогічно, якщо треба очікувати завершення усіх передумов (тобто, коли найпізніша завершиться), то виникає "*max*", тощо.

Процесні алгебри (ПА) не призначені для опису чи подання будь-чого на певному рівні ієрархічного моделювання у КІВС. Використання цього апарату забезпечує композиційність у формально-математичному, а не тільки інтуїтивному змістові (тобто надається можливість застосовувати математичні операції на рівні опису системи із застосуванням знаків, символів тощо, перетворювати вирази за визначеними правилами, як це робиться в алгебрі взагалі, але в даному випадку кожний вираз визначає систему, а точніше, процес, що нею реалізується). На відмінність від традиційної теорії автоматів, ПА в більшій мірі спроможні подавати паралелізм, комунікаційність тощо. Щодо *сіток Петрі*, то в порівнянні з ними ПА краще забезпечують зручний синтаксис, проте гірше подають паралелізм. Можна стверджувати, що ПА спрямовані своїм вмістом на моделювання часу та синхронізацію і різняться своїми семантичними та синтаксичними відмінностями.

В цій статті не ставиться за мету розгляд усіх існуючих ПА з їх відмінностями, перевагами та недоліками, а робиться спроба подання деякої стандартної ПА для виявлення основних характеристик цього формального апарату при застосуванні до моделювання взаємодії компонентів КІВС. З вище означеної специфіки задачі, що розглядається, та характеристик апаратів моделювання зупинимося на ПА через те, що вона, по-перше, є найменше дослідженою з позицій застосування до задач моделювання процесів в КІВС, а по-друге, властивості цього апарату є привабливими для подання важливих аспектів функціонування складних систем.

Процесні алгебри. Означення, приклади застосування.

Означення 3. *Процесна алгебра* – формальна мова для специфікації та обгрунтування поведінки процесів при їх взаємодії. Існує багато різноманітних видів процесних алгебр. Процесна алгебра оперує поняттями дій. За допомогою операторів ПА будуються вирази процесів, які і визначають модель системи.

Серед найбільш відомих прикладів відмітимо *CSP (Communicating Sequential Processes)* [20]. Пізніше *CSP* була використана як основа для мов паралельного програмування та специфікації. Серед найбільш поширених результатів є мова програмування *Occam* [16].

Ще одним прикладом є числення Мілнера – *CCS (Calculus of Concurrent Systems)* [22], яке з'явилося десь у той самий час і має близьку до *CSP* область застосування.

І *CSP*, і *CCS* в їх звичайних формах підтверджують, що процеси синхронізації є однаковими при взаємодії, але мають тонкі семантичні відмінності [15].

Визначення понять ПА. Для опису формального апарату ПА визначимо основні його поняття [11].

Означення 4. *Подія* є основною дискретою – станом системи у момент часу. Послідовність або набір таких подій описує поведінку КІВС, що є основною задачею моделювання.

 ISSN 1562-9945

Означення 5. *Процесом* називається чергування (послідовність) станів (подій) системи. Отже, поняття процес означає поведінку об‘єкта, яка описується в термінах обмеженого набору подій. Вважається, що конкретна подія в "житті"об‘єкта відбувається миттєво і немає тривалості у часі. Тривалу дію розглядатимемо як пару подій, де перша означає початок дії, а друга - кінець. Дві тривалих дії можуть перекриватися у часі, якщо початок кожної з них передує завершенню другої.

Означення 6. Множина імен подій, що відібрані для конкретного опису об‘єкта, називається його *алфавітом*. При виборі алфавіту не обов‘язково розрізняти події, що ініціюються самим об‘єктом або деяким зовнішнім до нього фактором. Імена подій позначатимемо малими літерами *-a,b,c,d,e*, ... , а імена процесів позначатимемо великими літерами - *A, B, C, D*, ...

Означення 7. Процес з алфавітом *A*, такий, що в ньому не відбувається жодна подія з *A*, називатиметься *stop* A. Цей процес описує поведінку пасивного об‘єкта: маючи фізичну здатність брати участь в подіях з *A*, він ніколи її не використовує.

Означення 8. *Протоколом поведінки* процеса називається скінчена послідовність символів, що фіксує події, в яких процес брав участь до деякого моменту часу. Протокол позначається послідовністю символів, які розділені комами та охоплені кутовими лапками:

$< x, y >$ (протокол для двох подій – x та наступної y);

$< x >$ для протоколу з однією подією x;

$<>$ для протоколу з пустою послідовністю, що не містить подій.

Означення 9. *Специфікація об‘єкта* - це опис його імовірної поведінки, який являє собою предикат із вільними змінними, кожна з яких відповідає деякому спостереженому аспекту поведінки об’єкта.

Означення 10. *Процесом взаємодії* називається такий процес, що об‘єднує два процеса для спільного виконання. Взаємодії, які полягають у передачі повідомлень і описуються парою $c.v$, де *c* – ім‘я канала, по якому відбувається взаємодія, а v – значення повідомлення, яке передається, являють спеціальний клас подій.

Означення 11. Ситуація, за якою процеси не можуть прийти до узгодженості щодо того, яка дія буде наступною, і при цьому ніякі дії не можуть відбуватися у подальшому, називається *дедлоком* або *тупиковою ситуацією*. Іноді ж процес має деякий спектр можливої поведінки, причому, його оточення не має можливості ані впливати на вибір між різними альтернативами, ані навіть спостерігати його. Іншими словами, вибір здійснюється довільним та недетермінованим способом.

Означення 12. В загальному випадку, нехай X – множина подій, які пропонуються на першому кроці оточенням процеса *P*, їх алфавіти співпадають з алфавітом процеса *P*. Якщо ж при розміщенні *P* в це оточення існує ризик дедлоку на першому кроці, то *X* – це *множина відмов процеса P*.

Означення 13. *Процес називається детермінованим*, якщо він ніколи “не відмовляється” від події, в якій може брати участь. Іншими словами, множина є відмовою детермінованого процеса, тільки якщо ця множина

не містить подій, в яких процес може брати участь на першому кроці.

Отже, у випадку, коли можливе здійснення більш як однієї події, вибір між ними визначатиметься зовнішнім по відношенню до процеса середовищем, тобто оточення процеса може в дійсності здійснювати вибір, або вибір стає відомим оточенню в той самий момент, коли він відбувається.

Означення 14. *Недетермінованим є процес*, який, хоча й міг би взяти участь у визначеній події, але, в результаті деякого внутрішнього недетермінованого вибору, може відмовитися від виконання цієї події, навіть якщо оточення не перешкоджає виконанню події. При цьому вибір здійснюється довільним та недетермінованим способом, і висновок про те, який саме вибір був зроблений, можна зробити тільки пізніше на основі аналізу подальшої поведінки процеса.

Означення 15. Канали використовуються для передачі повідомлень тільки в одному напрямку и тільки між двома процесами. Канал, що використовується процесом тільки для виводу повідомлень, будемо називати *вихідним каналом* цього процеса, а канал для вводу повідомлень – *вхідним каналом*.

Означення 16. У КІВС процес, що має у алфавіті лише вхідний та вихідний канали, називатимемо *транспортером*.

Означення 17. Процес, який виводить з можливою деякою затримкою введену до нього послідовність інформаційних або матеріальних об'єктів, називається *буфером*, і є завжди непорожнім і готовим до виводу об'єктів. Буфери зберігають об'єкти, які очікують на обробку, і є корисними для специфікації поведінки протоколів зв'язків, чиє призначення полягає у передачі об'єктів в тій послідовності, в якій вони надходять до буфера. Такі протоколи складаються з двох процесів – передатчика T та приймача R, які поєднані в ланцюжок $(T >> R)$. Якщо протокол правильний, то $(T >> R)$ – є буфер.

Твердження 1. Якщо Q виступає як основний чи головний процес, то в комбінації $(P||Q)$ процес P виступає *підлеглим* процесом.

Означення 18. *Катастрофічним перериванням* процеса P є переривання, яке викликане не самим P.

Означення 19. *Контрольною точкою процеса* є такий його стан, за яким катастрофа вимагає *перезапуску* процеса шляхом повернення до деякого попереднього стану системи, про який відомо, що він є задовільним. Оскільки може статися, що однієї контрольної точки може виявиться недостатньо, важливою задачею є вибір та збереження в системі набору контрольних точок.

Базові оператори ПА. Оператори ПА змінюють стан процеса та поєднують процеси один з одним. Основні оператори ПА наведені у таблиці 1.

Таблиця 1

Оператор пасивності - stop процеса A визначає процес з алфавітом A, такий що в ньому не відбувається жодної події з A, і позначається *stop* A.

 ISSN 1562-9945

Табл. 1 – Оператори процесної алгебри

Назва оператора	**Умовне позначення оператора**
пасивність	stop
дія-префікс	$a; B$ чи $a\Box B$ чи $a.B$
вибір	$B + C$ чи ΣB_i чи $B\Box C$
композиція	$B\|\|_A C$ чи $B/[A]\|C$
сховування	B/A чи hide Ain$B$
визначення	$p := B$
застосування	p

Цей оператор описує поведінку “пошкодженого” об‘єкта: маючи фізичну здатність брати участь в подіях A, він ніколи її не використовує.

Оператор дія-префікс визначає об‘єкт, який спочатку приймає участь в деякій події x, а потім веде себе точно як процес P. При цьому $(x \to P)$, читається як “P за x”. Взагалі процес $(x \to P)$ має за визначенням той же алфавіт, що і P, тому це позначення можна використовувати тільки за умови, що x належить тому ж алфавіту. Більш формально: $a(x \to P) = aP$, якщо $x \in aP$.

Оператори вибору.

Оператори вибору послідовності описують об‘єкт, який спочатку бере участь в одній з подій. Наприклад, якщо b і c – різні події, то $(b \to B|\ \ c \to C)$ або $(b.B+c.C)$. Наступна поведінка об‘єкта описується процесом C, якщо першою відбулася подія c. Оскільки b і c – різні події, то вибір між B та C визначається тим, яка з подій у дійсності настала першою. Вертикальну риску | слід читати як “чи”: “B за b чи C за c”.

Взагалі, оператор вибору послідовності $(x :\ \ B \to P(x))$ використовується для визначення процеса, який володіє цілим спектром можливої поведінки. Зокрема, оператор || дозволяє деякому іншому процесу робити вибір між альтернативами з множини B – це *оператор детермінованого вибору*.

Оператор недетермінованого вибору описує процес, який здійснюється довільно без відома чи контролю з боку зовнішнього оточення. Так, якщо P та Q – процеси, то оператор $P \prod Q$, або P чи Q визначає процес, який веде себе або як P, або як Q (алфавіти операндів співпадають: $a(P \prod Q) = aP = aQ$. Таким чином, процес, що описаний за допомогою оператора $P \prod Q$, може бути реалізований або як P, або як Q.

Оскільки оточення процеса $(P \prod Q)$ не може не тільки впливати на вибір процеса, але навіть не знає (у випадку недетермінованого вибору) результат вибору, в ПА вводиться операція (з відповідним оператором) *генерального вибору* $(P\Box Q)$, за яким оточення вже може керувати вибором між P та Q за умови, що цей вибір буде зроблений на першому кроці.

Це означає, що якщо перша дія не можлива для P, то вибирається Q; і навпаки, якщо Q не може виконати першу дію, то вибирається P. Але якщо перша дія можлива як для P, так і для Q, то вибір між ними залишається недетермінованим, і має місце: $a(P \prod Q) = aP = aQ$.

Випадок, коли операція генерального вибору аналогічна операції вибору:

$(c \to P \square d \to Q) = (c \to P | d \to Q)$, якщо $c \neq d$.

Випадок для однакових початкових умов: $(c \to P \square c \to Q) = (c \to P \prod c \to Q)$, якщо $c \neq d$.

Відмінність між процесами $(P \prod Q)$ та $(P \square Q)$ є тонкою. Ці процеси можна помістити в таке середовище, в якому процес $(P \prod Q)$ на першому кроці увійде у тупиковий стан (дедлок), а процес $(P \square Q)$ – ні. Нехай, наприклад, $x \neq y$, причому

$$P = (x \to P),\ Q = (y \to Q),\ ap = aQ = \{x, y\}$$

Тоді процес $(P \square Q) || P + (x \to P) = P$, а процес $(P \square Q) || P = (P || P) \prod (Q || P) = P \prod stop$.

Отже, в оточенні P процес $(P \prod Q)$ може прийти в тупиковий стан, а процес $(P \square Q)$ – ні.

Інколи умови функціонування КІВС вимагають об'єднання процесів з однаковими алфавітами для паралельного виконання, в ході якого не відбувається безпосередньої їх взаємодії чи синхронізації. В цьому випадку кожна дія системи – це дія в точності одного з процесів.

Існує ще один оператор вибору – це *оператор чергування*. Дійсно, якщо один процес P не може виконати дію, це повинен зробити інший процес Q (генеральний вибір). Якщо ж одну дію готові виконати обидва процеси P та Q , то вибір між ними є недетермінованим, і для упорядкування недетермінованості вводиться *оператор чергування* $|||$, тобто якщо P чергується з Q, то $P|||Q$.

Оператори взаємодії (композиції).

Ці оператори призначені для об'єднання і подання спільного виконання двох процесів. Взаємодії процесів можна розглядати як події, які вимагають одночасної участі обох процесів. Нехай алфавіти цих процесів співпадають. Якщо P та Q – процеси з однаковими алфавітами, то позначимо через $P||Q$ процес, який веде себе як система, яка складена з процесів P і Q, взаємодія між якими покроково синхронізована.

Оператор паралельної композиції має позначення $||$. Внутрішні події, що відповідають внутрішнім переходам одночасно протікаючих процесів і взаємодії та зв'язкам між паралельно працюючими компонентами, супроводжуються діями, які мають відбуватися автоматично в той момент, коли це можливо, і при цьому залишаються недосяжними для контролю та спостереження з боку оточення процесів. Слід також зазначити, що якщо в комбінації $(P||Q)$ процес Q виступає як основний, то, відповідно до *твердження 1* процес P буде підлеглим. Якщо відношення основного та підлеглого процесів необхідно приховати від їх загального оточення,

 ISSN 1562-9945

застосовується позначення: $P||Q = P||Q)\backslash aP$. Крім того, підлеглість може бути *вкладеною*, наприклад $(n : (m : P||Q)||R)$. В такому випадку усі входження подій, що містять ім'я m, сховуються перш, ніж ім'я n приписується подіям, що залишилися, кожна з яких входить до алфавіту Q та не входить до алфавіту P; процес R при цьому не має можливості ані безпосередньо взаємодіяти з P, ані навіть знати про існування P та його імені m.

Оператор послідовної композиції $(P;Q)$. При проектуванні часто складна задача розподіляється на дві підзадачі, одна з яких успішно завершується до початку іншої. Така складна задача уявляє собою процес, який веде себе спочатку як P, а після завершення Pпродовжує вести себе як Q. Якщо ж успішного завершення P не відбувається, то не завершується й $(P;Q)$. В такій послідовній композиції виконання наступного процеса залежить від успішного виконання попереднього процеса.

Оператор переривання позначається як $(P\wedge Q)$ і застосовується для подання композиції процесів з перериванням, якщо послідовна композиція $(P;Q)$ не залежить від успішного завершення P, і при першій же з подій процеса Q виконання P переривається і вже не поновлюється.

Оператор катастрофічного переривання використовується тоді, коли переривання P викликане не самим процесом P (так званний вихід процеса у нештатну ситуацію), і тоді процес, який веде себе до катастрофи як P, а після неї як Q, подається наступним чином:

$P\wedge_{\leftarrow\!\rfloor} Q = P\wedge(\rightarrow {}_{\leftarrow\!\rfloor} Q)$ причому, тут Q може відігравати роль процеса, який ліквідує наслідки катастрофи.

Оператор перезапуску первинного процеса використовується (відповіднодо *означень* 16 – 17), якщо визначити $P\wedge$ як процес, який веде себе як P до настання переривання ${}_{\leftarrow\!\rfloor}$ а після кожної події ${}_{\leftarrow\!\rfloor}$ знов веде себе як P, але з самого початку (інакше кажучи, цей процес є *таким, що перезапускається*), і визначається рекурсією:

$$aP\wedge = aP \cup \{{}_{\leftarrow\!\rfloor}\},$$

$$P\wedge = \mu X.(P \wedge {}_{\leftarrow\!\rfloor} X) = P \wedge {}_{\leftarrow\!\rfloor}(P \wedge {}_{\leftarrow\!\rfloor}(P \wedge {}_{\leftarrow\!\rfloor}\ldots)),$$

тобто $P\wedge$ – циклічний процес, навіть якщо P таким не є.

Зв'язок подання за допомогою ПА з алгоритмічним поданням. Однією з сфер застосування ПА стали *CSP*-подібні мови паралельного програмування [16]. Так, операторами-прототипами основних алгоритмічних структур стали: *присвоєння, умова, цикл.*

Присвоєння. Нехай x – програмна змінна, e – вираз, а P – процес. Тоді $(x := e;P)$ – це процес, який веде себе як P з початковим значенням x, дорівнюваним початковому значенню виразу e.

Умова. Нехай вираз b обчислює істинність логічної функції (значення *TRUE*, або *FALSE*). Якщо P таQ – процеси, то $P \Leftarrow b \Rightarrow Q$ – це процес, який веде себе як P, якщо початкове значення є *TRUE*, або як Q, якщо початкове значення є *FALSE*, тобто: if b then P else Q.

Цикл. Цикл позначається через $b * Q$, а традиційний цикл: while b do Q.

Операції над протоколами. Найважливішою операцією над протоколами, яким належить основна роль у фіксації, описі та розумінні поведінки процесів, є конкатенація.

Означення 20. *Конкатенація* – це операція побудови нового протоколу з пари операндів – протоколів s та t шляхом з'єднання їх в зазначеному порядку як $s \wedge t$.

Властивості конкатенації:

асоціативність, а також те, що пустий протокол $<>$ слугує для неї одиницею, наприклад: $s\wedge <>=<> \wedge s = s$, а також $s \wedge (t \wedge u) \wedge = (s \wedge t) \wedge u$;

строгість, коли, наприклад, функція f відображає протокол у пустий протокол: $f(<>) =<>$;

дистрибутивність, якщо: $f(s \wedge t) = f(s) \wedge f(t)$. Всі дистрибутивні функції є строгими;

- індукційність по n, де n – натуральне число. Отже, наприклад, t^n є конкатинацією n копій протоколу t. Таким чином, має місце: $t^0 =<>$, а також

$t^{n=1} = t \wedge t^n$.

Іншою важливою операцією над протоколами є їх звуження.

Означення 21. *Звуження протоколу* – це операція $(t\lceil A)$ відкидання з протоколу t символів, які не належать до A (тобто звуження на множину символів A). Звуження є дистрибутивним:

$<> \lceil A =<>$, а також $(s \wedge t)\lceil A = (s\lceil A \wedge (t\lceil A))$.

Для одноелементних послідовностей ефект звуження:

$$< x > \lceil A =< x >, \quad x \in A, \quad < y > \lceil A =<>, \quad y \notin A.$$

Від протоколу, звуженого на пусту множину, нічого не залишається:

$s\lceil\{\} =<>$,

а послідовне звуження на дві множини рівнозначне одному звуженню на перетин цих множин:

$(s\lceil A)\lceil B = s\lceil(A \cap B)$.

Означення 22. *Головою протоколу* є перший елемент s_o непустої послідовності s.

Означення 23. *Залишок протоколу* – це результат, що отримується при видаленні s_oз s та позначається через $s\prime$:

$< x,y,x >_o = x$, причому $< x,y,x > \prime = . < y,x >$.

Означення 24. *Множина скінчених протоколів* A^* - це набір всіх скінчених протоколів (включно $<>$), складених з елементів множини A. Після звуження на A такі протоколи залишаються незмінними, звідки: $A^* = \{s | s\lceil A = s\}$.

Означення 25. *Довжина протоколу* $\#t$ – це кількість подій, що входять до протоколу. Наприклад, $\# < x,y,x >= 3$.

До початку процеса P невідомо, який саме з можливих протоколів буде записаний, бо його вибір залежить від зовнішніх по відношенню до процеса факторів. Проте, повний набір всіх можливих протоколів процеса P може бути відомий раніше, тому вводиться функція *protocols* (P) для позначення цієї множини. Протокол процеса $(c \rightarrow P)$ може бути пустим

 ISSN 1562-9945

через те, що $<>$ є протоколом поведінки будь-якого процеса до моменту початку його першої події. Тому кожний непустий протокол цього процеса починається з c, а його хвіст має бути можливим протоколом P:

Protocols $(c \to P) = \{t | t =<> \vee (t_o = c \& t\prime \quad \in protocols(P))\}$;

$= \{<> \cup \{< c > \wedge t | t \in protocols(P)\}\}$.

Існує тісний зв'язок між протоколами процеса та зображенням поведінки останнього у вигляді дерева. Для кожної вершини дерева протоколом процеса до моменту досягнення цієї вершини буде просто послідовність міток на шляху з кореня дерева в цю вершину. Протоколи процеса уявляють собою множину всіх шляхів, які ведуть з кореня в різні вершини дерева. І навпаки, через те, що всі дуги , які ведуть з кожної вершини, помічені різними подіями, кожний протокол процеса однозначно визначає шлях з кореня дерева у деяку з вершин.

Твердження 2. Якщо $s \in protocols(P)$, то $P/s(P \quad після \quad s)$ – це процес, який веде себе так, як веде себе P з моменту завершення усіх дій, що записані у протоколі s. Якщо ж s не є протоколом P, то (P/s) не визначене.

Крім конкатенації, іншою важливою операцією над протоколами є вибірка.

Твердження 3. Якщо s – послідовність пар, то $s \downarrow x$ визначає *результат вибірки* з s усіх пар, першим елементом яких є x, і заміни такої пари на її другий елемент, причому, перший та другий елементи пари розділяються крапкою: нехай $s =< a.7, b.9, a.8, c.0 >$, тоді $s \downarrow a =< 7, 8 >$, проте $s \downarrow d =<>$. Якщо ж s не є послідовністю пар, то $s \downarrow a$ позначає кількість входження a в s.

Якщо t – протокол, який відображає послідовність подій з того моменту, коли деяка послідовність s успішно завершилася, то їх композиція позначається як $(s\surd; t)$, де символ $\surd$ означає успішне завершення послідовності s. При відсутності в s такого символу протокол t в композиції $(s; t)$ не може розпочатися, тобто $(s; t) = s$, якщо $\neg(< \surd > в s)$.

В разі присутності символу $\surd$ наприкінці s, то він видаляється, а t приєднується до результату, наприклад: $(s \quad < \surd >)$; $t = s \wedge t$, якщо $(< \surd >$ після $s)$.

Протоколи процесів відіграють основну роль в специфікаціях систем.

Означення 26. Опис ймовірної поведінки досліджуваного об'єкта є його *специфікацією*, яка являє собою предикат з вільними змінними, кожна з яких відповідає деякому наглядовому аспекту поведінки об'єкта. Наприклад, специфікація електронного підсилювача з вхідним діапазоном в 1 В та з коефіцієнтом підсилення 30 задається предикатом: $U30 = (0 \le v \le 1 \Rightarrow |v' - 30 \times v| \le 1)$.

В наведеній специфікації v означає вхідну, а v' – вихідну напруги.

Якщо P – об'єкт, що відповідає специфікації S, то P *satisfy* S (тобто задовольняє S). І це означає, що S описує усі можливі результати спостереження за поведінкою P:

$$\forall np.np \in protocols(P) \Rightarrow S.$$

Твердження 4. Якщо з специфікації S логічно випливає інша специфікація T, то будь-яке спостереження, що описується S, описується також і T. Отже, кожний об'єкт, що задовольняє S, в даному разі повинен також задовольняти і більш слабкій специфікації T:

Якщо P *satisfy* S та $S \Rightarrow T$, то P *satisfy* T.

Взаємодія процесів. Взаємодія процесів – це спеціальний клас подій. Взаємодія полягає в передачі повідомлень і власне є подією, яка описується парою $c.v$, де c – це ім'я каналу, по якому відбувається взаємодія, а v– значення повідомлення, яке передається.

Процеси вводу та виводу. Процес P може здійснювати взаємодію по каналу c через множину повідомлень: $ac(P) = \{v \,|c.v \in aP\}$; при цьому функції, що вибирають ім'я каналу c та значення повідомлення v, визначаються, відповідно, як:

$$canal(c.v) = c.$$

$$inform(c.v) = v.$$

Нехай повідомлення v є елементом множини повідомлень $ca(P)$. Тоді процес, який спочатку виводить v по каналу c, а потім веде себе як P, позначається так:

$(c!v \to P) = (c.v \to P)$

З іншого боку, процес, який в початковому стані готовий ввести будь-яке значення x, яке передається через канал c, а потім веде себе як $P(x)$, позначається так:

$$(c?x \to P(x)) = (y : \{y \,|canal(y) = c\} \to P(\text{inform (y)})).$$

Означення 27. *Відтворюваною подією процеса вводу-виводу* є взаємодія $c.v$, де v-значення, яке визначається процесом виводу.

Отже, якщо дотримуватись правил сполучення процесів P (той, що виводить повідомлення) і Q (той, що вводить в той же час це повідомлення), то взаємодія їх по каналу c визначається як $ac(P) = ac(Q)$ із співпаданням алфавітів на обох кінцях каналу c і відбувається за наступними законами:

1. $(c!v \to P) \,\|(c?x \to Q(x)) = c!v \to (P \,\|Q(v)\,)$;
2. $(c!v \to P) \,\|(c?x \to Q(x))\backslash C = (P \,\|Q(v)\,)\backslash C$, де $C = \{c.v \,|v \in ac\}$.

Процеси-транспортери. Такі процеси відіграють важливу роль при композиційному поданні транспортних операцій.

Означення 28. Процес, що має у алфавіті лише два канали: вхідний та вихідний, називається *транспортером*. Результатом послідовного поєднання транспортерів P, Q, R тощо є $(P >> Q) >> R$ і т.д.

Означення 29. *Специфікація транспортера* – відношення S(*in,out*) між послідовністю повідомлень (об'єктів, процесів) з вхідного *in* та вихідного *out* каналів.

Оскільки при поєднанні двох транспортерів у ланцюжок спільна послідовність сховується, то для запобігання ризику замкнення виводи-

 ISSN 1562-9945

ться правило: ЯКЩО *P setisfy S*(*in,out*), а *Q setisfy T*(*in,out*), та *P* попередній зліва, а *Q* попередній справа, ТО *(P»Q) setisfy* $\exists s.S(in,s)$ & *T*(*s,out*).

Процес – буфер. Такий вид процесів є дуже поширеним в КІВС і, як правило, корегує дисбаланс продуктивностей послідовних технологічних процесів в системі.

Означення 30. *Буфер* – це процес, який виводить, після деякої затримки, таку послідовність, яка вводиться. Буфер є непустим і завжди готовий до виводу.

Наступні процеси є буферними: *BUF*, (*COP»COP*).

Буфери використовуються або для зберігання матеріальних об’єктів (в т.ч. інформаційних), або для специфікації поведінки протоколу зв’язків (забезпечуючи передавання об’єктів в тій послідовності, в якій вони надходять). В останньому випадку протокол складається з двох процесів - завантажувача *T* та приймача R, поєднаних у ланцюжок $(T >> R)$. Оскільки на практиці приймач та завантажувач поєднуються доволі довгим з’єднувачем, що може призвести до пошкодження та втрати повідомлень, які надсилаються, то з’єднувач моделюється окремим процесом *CONNECT*, а саме: (*T»CONNECT»R*)

Приклад 1. Простий однопозиційний буфер *Buf*:=*in*; *out*;*Buf*

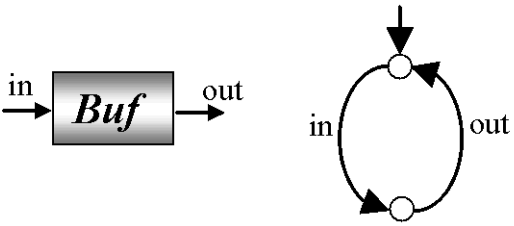

Рис. 1 – Схема та граф однопозиційного буфера

Приклад 2. Складний однопозиційний буфер, утворений послідовним з’єднанням двох простих однопозиційних буферів.

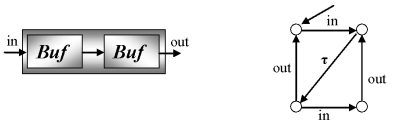

Рис. 2 – Схема та граф складного однопозиційного буфера

Приклад 3. Двопозиційний буфер.
Buf 2:=*in; Half*
Half:=*in; Full*+*out; Buf 2*
Full:=*out; Half*

Алгебраїчні закони та їх узагальнення. Тотожність процесів з однаковими алфавітами можна визначати за допомогою алгебраїчних законів,

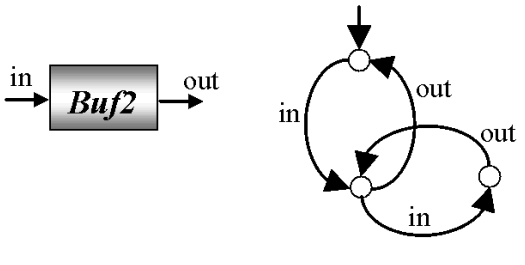

Рис. 3 – Схема та граф двопозиційного буфера

скориставшись поняттям *еквівалента*. Еквіваленти вивчають різноманітні подання понять. Прикладами еквівалентів можуть бути: еквіваленти шляху, тестування і т.і. Різні еквіваленти відрізняються властивостями. Застосування еквівалентностей підпорядковується закономірностям, що спричинило появу алгебраїчних законів, наведених нижче:

A+stop=A; A+A=A;

Комутативність: $B + C = C + B$ *та* $B||_A C = C||_A B$;

Асоціативність: $(B + C) + D = B + (C + D)$ *та* $B||_A(C||_A D) = (B||_A C)||_A D$.

Для ПА є своє тлумачення алгебраїчних законів:

1. *Стосовно оператора вибору.*

Означення 31. Два процеси, визначені за допомогою оператора вибору, *є різними*, якщо на першому кроці вони пропонують різні альтернативи, або після однакового першого кроку ведуть себе по-різному, і *є тотожними*, якщо множини початкового вибору виявляються рівними та для кожної початкової альтернативи подальша поведінка процесів співпадає, тобто: $(x : A \to P(x)) = (y : B \to Q(y)) \equiv (A = B \ \& \ \forall x \in A.P(x) = Q(x))$

2. *Які керують поведінкою процесів з паралельною композицією* $(P||Q)$.

а) закон відображення логічної симетрії "процес - оточення": $P||Q = Q||P$;

б) закон відображення спільності роботи трьох і більше паралельних процесів: $P\ ||(Q||R) = (P||Q)||R$.

Тобто, при спільній роботі декількох процесів неважливо, в якому порядку вони об'єднані оператором паралельної композиції;

в) закон виникнення тупикової ситуації всієї системи: $P||stop\ aP = stop\ aP$, тобто, якщо процес знаходиться в тупиковій ситуації, то це призводить до дедлоку всієї системи;

г) закон одночасності виникнення однакових дій парою процесів $(c \to P)||(c \to Q) = (c \to (P||Q)); (c \to P)||(d \to Q) = stop$, якщо $c \neq d$. Цей закон узагальнюється на випадки, коли у одного чи обох процесів є вибір початкової події. При комбінації процесів залишаються можливими лише ті події, які містяться у множинах вибору обох процесів: $(x : A \to P(x))\,\|(y : B \to Q(y)) = (z : (A \cap B) \to (P(z)\,\|Q(z)))$.

 ISSN 1562-9945

Саме цей закон дозволяє описати систему, визначену у термінах паралельного виконання без використання паралелізму.

3. *Послідовної композиції*:

$(P;Q);\ R=R;\ (Q;R);$

$x:B\to P(x));\ Q=(x:B\to(P(x);\ Q));$

$(a\to P);\ Q=a\to(P;Q);$

Stop; Q=*stop*.

Реалізація процесів. Для охарактеризування загальних умов реалізації процесів скористаємось наступним твердженням:

Твердження 5. Будь-який процес P, який записаний за допомогою введених в даній роботі позначень, може бути поданий у вигляді: $x : B\square F(x)$, де F- функція, що ставить у відповідність множині символів множину процесів.

Власне, кожний процес можна розглянути як функцію F з областю визначення B, яка виділяє множину подій, що слугують початковими подіями процеса. Якщо першою відбулася подія x з B , то $F(x)$ визначає подальшу поведінку процеса. При цьому множина B може бути пустою (випадок *stop*), мати один елемент (випадок префікса), або мати більше, як один, елемент (випадок вибору).

Сфера застосування ПА. Отже, у сучасному стані ПА - це формалізм для визначення поведінки системи у семантичному, модульному та ієрархічному поданні [14]. За допомогою операторів ПА будуються вирази процесів, дій, що забезпечують композиційність системи. Застосування операторів ПА спрощує внутрішню будову системи за рахунок внутрішніх перетворень та безпосереднє подання системи за рахунок того, що оператори в свою чергу можуть складатися з більш дрібних операторів.

Найбільш поширеною сферою застосування ПА є специфікація, а також розробка та реалізація систем, які неперервно діють та взаємодіють зі своїм оточенням. При цьому основна ідея полягає в тому, що ці системи можна розкласти на паралельно працюючі підсистеми, які взаємодіють як одна з одною, та і зі своїм загальним оточенням. Такий підхід є надійною основою для уникнення тупикових ситуацій та зациклювань [11].

Моделі у ПА можна подати у вигляді орієнтованого графа, дуги якого позначаються латинськими літерами та називаються діями, а вершини позначаються прописними літерами і називаються факторами (агентами). Приклади моделей, зображені на рис. 4, представлені множиною дій $\{a,b,c,d,e\}\cup\{t\}$ та факторів $E, E', C1, C2, F, F1, F2, G1, G2$.

Апарат ПА може допомогти розділити поняття сутності інтерактивнокерованої системи від особливості будь-якої визначеної взаємодії. Застосовуючи ПА, інколи можливо сконструювати формальні доведення властивостей поданих систем [18]. У випадках, де шаблон взаємодій між програмними компонентами стає складним (комплексним), ПА використовуються в аналізі чи для однакового визначення специфікації на верхньому рівні абстракції.

Моделювання взаємодії компонентів КІВС на основі ПА

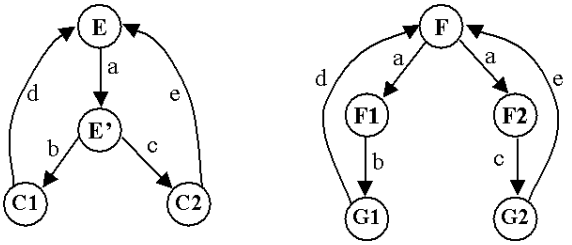

Рис. 4 – Дві відмінні моделі з однаковим набором подання

Розглянемо ПА, яка пропонується для розв’язку задачі моделювання взаємодії компонентів КІВС. При цьому застосовується наступний підхід.

Система КІВС розглядається як сукупність дій її компонентів, що взаємодіють один з одним. Відповідно, кожний компонент системи моделюється послідовним процесом. Взаємодія компонент КІВС моделюється наступним чином: по фіксованим комунікаційним каналам приймаються та відправляються дії компонентів. В моделюванні подання дискретно-подійних систем на основі цього підходу є більш легким, зрозумілим та зручним, ніж моделі, що використовують інші підходи.

Розглянемо деяку абстрактну систему, що складається з двох типових компонентів, які описуються моделями:

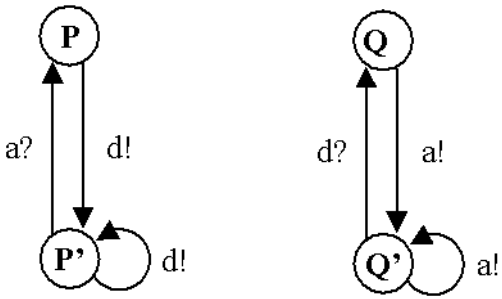

Рис. 5 – Моделі компонентів

Кожна модель компонента також може бути визначена у вигляді системи рівнянь. Так, для першої моделі:

$P = a?.P + d!.P'$;

$P' = d!.P' + a?.P + d!.P'$,

та для другої моделі:

$Q = a!.Q' + d?.Q$;

$Q' = a!.Q' + d?.Q + a!.Q'$.

Тоді модель системи може бути представлена композицією цих двох компонентів. При цьому утворюються взаємні зв’язки та додаткові стани

системи, що відображають взаємодію компонентів та в цілому моделюють поведінку системи. Модель системи, утвореної композицією моделей компонентів (рис.5), наведена на рис. 6.

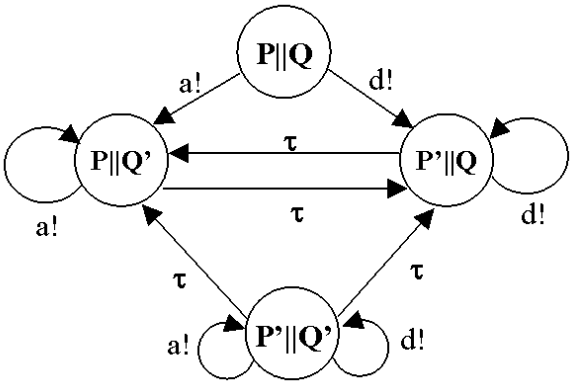

Рис. 6 – Модель взаємодії двох компонентів

Отримана модель відображає паралелізм та невизначеність при взаємодії двох компонентів у системі. У графічному вигляді подання на цій моделі як на прикладі можна розглянути всі вище зазначені моделі подання КІВС.

Більш складні реальні моделі процесів в КІВС із застосуванням апаратів ПА, асинхронних протоколів (АП), алгебр (max/min, +) розглядаються в раніше опублікованих авторами роботах. Зокрема, в роботі [7] запропоновано нові стратегії управління розподіленими робочими потоками КІВС з уникненням тупикових ситуацій. *Стратегія синхронізуючих зон* базується тільки на ресурсних потребах системи, поданих графом потреб ресурсів (ГПР). При цьому інформація про виробничий маршрут зводиться до “розділення” послідовності виготовлення об’єкта виробництва, в результаті чого відокремлені зони розглядаються як “паростки” в системі. *Стратегія графа розподілу заявок* (ГРЗ) базується на розгляді сегментів КІВС, які синзронізуються, і забезпечує локальними правилами керування робочими потоками та вимагає визначення набору таких сегментів в ГРЗ-моделі функціонування КІВС. Така стратегія пов’язана з визначенням усієї ГРЗ-моделі системи, а оскільки проблема дослідження циклів є NP-повною задачею, обчислення виконується в режимі *off-line*, тоді як керування окремим циклом (сегментом або петлею графа), що аналізується, здійснюється в режимі *on-line*. І, нарешті, *стратегія, яка базується на протоколах розподілу потужностей* (ПРП) використовує ідею самосинхронізації робочих потоків в результаті визначення буферних потужностей і формування правил диспетчерування відносно даного набору виробничих маршрутів. В цьому випадку протоколи розглядаються як

процедури, що координують конкуренцію процесів за доступ до необхідним чином розподілених і обмежених ресурсів.

Інше застосування механізмів ПА щодо дослідження проблеми управління обмеженими перепускними спроможностями композиційних складових КІВС з періодичним характером операцій наведено в [5]. Композиційна складова тут розглядається як робоче місце с конкуруючими за доступ до загальних обробляючих ресурсів робочими потоками, які об’єднують технологічне устаткування, вхідний/вихідний буфери обмеженої потужності та обслуговуючі їх промислові роботи. Розв’язання задачі полягає у визначенні достатніх для формування правил розподіленого диспетчерування робочих потоків і умов, які базуються на оціночних критеріях системної циклової продуктивності та коефіцієнтах використання ресурсів КІВС. З цією метою запропоновано *логістичний підхід з використанням поняття асинхронних протоколів*, які і забезпечують локальний механізм для вище означених показників ефективності.

В роботі [9] на основі алгебраїчної моделі з використанням модифікованого апарату min-плюс-алгебри КІВС розглядається як розподілена система, яка складається з автономних повторюваних процесів (транспортних потоків), утворюючих загальний набір і конфігурацію шляхопровідних секцій з визначеними віддалями між їх перетинами. Задача розробки розкладу обслуговування обробляючих ресурсів транспортною системою, за яким стабілізується функціонування КІВС, супроводжується одночасно умовою щодо забезпечення і мінімального циклового часу (обернена величина – перепускна спроможність системи). Для розв’язання цієї задачі запропоновано підхід, який базується на використанні *принципу “віртуальних асинхронних транспортних світлофорів”*, які на маршрутних перетинах забезпечують механізми синхронізації з локальним обмеженням потоку індивідуальних транспортних засобів-робокарів. В результаті, “світлофорне регулювання” завершується усталеним циклічним режимом функціонування системи.

Висновки

Проведений аналітичний перегляд методів моделювання КІВС, який дозволив визначити місце та сферу застосування апарату ПА щодо розв’язання задач синтезу/аналізу складних дискретно-подійних систем.

Досліджено і системно узагальнено основні властивості апарату ПА з інтерпретуванням їх до процесів, які відбуваються у КІВС під час функціонування.

Визначено сферу ефективного використання алгебраїчних підходів в задачах розподіленого управління матеріальними потоками КІВС.

Література

1. Дубина Д.А., Ямпольский Л.С., Пуховский Е.С. Использование модифицированных сетей Петри при моделировании гибких производственных систем //Автом. виробн. проц.-2001р.- 2(13).- С.60-67

 ISSN 1562-9945

2. Интегрированная семантически согласованная среда гиперпространственного моделирования гибких компьютеризированных производственных / О.И. Лисовиченко, В.И. Костюк, А.А. Лавров, Л.С. Ямпольский //Автом. виробн. проц. -2001р.- 2(13).- С.86-100

3. Лавров О.А. Моделювання гнучких дискретно-подійних систем на основі методів з комбінованою семантикою / Дис. на здоб. наук. ступ. докт. техн. наук, Київ, 2000

4. Лісовиченко О.І., Ланкін Ю.М., Ямпольський Л.С. Семантично-узгоджене середовище гіперпросторових моделей складальних комп'ютерно-інтегрованих систем //Міжвід. наук.-техн. зб. "Адаптивні системи автоматичного управління". - Дніпропетровськ: ДНВП Системні технології, 2000.-Вип. 3'(23). - С.137-147.

5. Моделювання розподіленого управління перепускними спроможностями циклічних виробничих систем /З.А. Банашак, О.І.Лісовиченко, Д.І. Гергі, Л.С. Ямпольський //Труды филиала МГТУ им. Н.Э. Баумана в г. Калуге. Спец. выпуск: Материалы междун. науч.-техн. конф. "Приборостроение-2001", - Калуга.-2001г.- С. 158-163.

6. Модифікація апарату сіток петрі і моделювання складних комп'ютерно-інтегрованих систем з ієрархічною семантикою подання процесів / П.В. Кузьмін, О.А. Лавров, О.І. Лісовиченко, К.Б. Остапченко, Л.С. Ямпольський // Вісник ЖІТІ.- Житомир: ЖІТІ, 1998.- 8.-С. 80-92.

7. Нові підходи до моделювання і управління в гнучких комп'ютеризованих системах / П.В.Кузьмін, О.А.Лавров, К.Б.Остапченко, З.А. Банашак, Л.С.Ямпольський // Міжвід. наук.-техн. зб. "Адаптивні системи автоматичного управління".-Дніпропетровськ: ДНВП Системні технології, 1998.-Вип. 1 (21).- С. 47-63.

8. Проблема автоматизации моделирования и управления в гибких гетерогенних распределенных сборочных системах / Л.С. Ямпольский, А.А. Лавров, О.И. Лисовиченко, А.Л. Сигал //Міжвід. наук.-техн. зб. "Адаптивні системи автоматичного управління".-Дніпропетровськ: ДНВП Системні технології, 1999.-Вип. 2 (22).- С. 53-73.

9. Реалізація концепції розподіленого керування з самосинхронізацією потоків транспортних засобів ГВС /З.А. Банашак, О.І. Лісовиченко, Г.М. Ткач, Л.С. Ямпольський // Міжвід. наук.-техн. зб. "Адаптивні системи автоматичного управління". -Дніпропетровськ: ДНВП Системні технології, 2001-Вип. 4'(24).- С.88-108.

10. Семантическое согласование описания процессов гибких производственных систем при композиционном моделировании / Л.С. Ямпольский, А.А. Лавров, Д.А. Дубина, О.И. Лисовиченко, В.В. Швачко //TECHNIKA I TECHNOLOGIA MONTAÏU MASZYN IV: Mikzdyn. konf. nauk.-techn.- Specjalny Dodatec do Kwartalnika Nauk.-Techn.

“TECHNOLOGIA I AUTOMATYZACJA MONTAÏU”.- 2001.-N 2(32).-P. 61-65.

11. Хоар Ч.Е.Р. Взаимодействующие последовательные процессы /Пер. с англ. А.А.Бульонковой, под ред. А.П.Ершова.- М.: Мир, 1989.- 264 с.
12. Ямпольский Л., Банашак З., Хасегава К., Крог Б., Такахаши К., Борусан А. Управление дискретными процессами в ГПС /Под ред. проф. Л.С. Ямпольского.-К.: Тэхника; Вроцлав: Изд-во Вроцлав. полит-го ин-та; Токио: Токосё.- 1992.-251с.
13. A max-algebra approach to the robust distributed control of repetitive AGV system /M.B. Zaremba, A. Obuchowicz, Z.A. Banaszak, and K.J. Jedrzejek // International J. “Production Research”.- 1997.-Vol.35, 10.- P. 2667-2687
14. Brinksma E.A., D’Argenio P., Hermanns H., and Katoen J.-P. Stochastic Process Algebras: linking process descriptions with performance //IFIP WG10.4 Stenungsund, 2001
15. Brookes S.D. On the relationship of CCS and CSP in Lecture Notes in Computer Science 4154: Automata, Languages, and Programming. The10-th Colloquium.- Springer-Verlag, 1983.- P.83-96
16. Burns A. Programming in Occam 2.- Addison-Wesley, 1988.
17. Chen P.P. The Entity-Relationship Model - Toward a Unified View of Data //ACM Transactions on Database Systems, 1976.- Vol.1, . 1

    Craigen D., Gerhart S., and Ralston T. Formal Methods Reality Check: Industrial Usage //IEEE Transactions on Software Engin., 1995.-Vol.21, .2.-P.90-98
18. Flater D. Specification of Integrated Manufacturing Systems.- NISTIR 6484, 2000.
19. Hoare C.A.R. Communicating Sequential Processes //Communicating of the ACM, 1978.-Vol.21, 8.- P.666-677
20. Lavrov A.A., Yampolsky L.S. Configuration Synthesis and Refinement: an Approach to Simulation of Heterogeneous Distributed systems // Межд. научн. журн. “Управляющие системы и машины”.-Ин-т киберн. им. Глушкова, 2000.- 3.-.-С.11-17
21. Milner R. Lecture Notes in Computer Science 492: A Calculus of Communicating Systems.-Springer-Verlag, 1980
22. Morris K., Plate D., Libes D., and Jones A. Testing of Interaction-driven Manufacturing Systems.- NISTIR 6260, 1998
23. Raynel M. and Helary J.-M. Synchronization and Control of Distributed Systems and Programs.- John Wiley & Sons,1990
24. Reisig W., Petri Nets: An Introduction.- Springer-Verlag, 1985