УДК 004.451

Ю. П. Кльоц, канд. техн. наук,
С. В. Мостовий

ПРОГНОЗУВАННЯ ВЗАЄМОБЛОКУВАНЬ ПРОЦЕСІВ В КОМП'ЮТЕРНИХ СИСТЕМАХ

Анотація. *Проведено аналіз життєвого циклу процесу, виділено граничний стан, який передуває стану взаємоблокування. Розроблено алгоритм виявлення процесів, які знаходяться в граничному стані і можуть потрапити в стан взаємоблокування і алгоритм прогнозування входження процесів в стан взаємоблокування в комп'ютерних системах. Проведена також оцінка часової складності розробленого алгоритму.*

Ключові слова: *взаємоблокування, стан процесу,алгоритм прогнозування стану взаємоблокування процесу*

Y. Klots, PhD.,
S. Mostovoy

DEADLOCKS PREDICTION IN COMPUTER SYSTEMS

Abstract. *In work the analysis of life cycle of process is conducted, the boundary status is chosen which will precede to a status of deadlock. The algorithm of revealing of processes is developed which are in a boundary status and can hit in a status of deadlock, and algorithm of forecasting of entry of processes in a status of deadlock in the CS. An evaluation of temporary complexity of the developed algorithm is conducted.*

Keywords*: deadlock, state of process, algorithm of deadlock prediction of process*

Ю. П. Клец, канд. техн. наук,
С. В. Мостовой

ПРОГНОЗИРОВАНИЕ ВЗАИМОБЛОКИРОВОК ПРОЦЕССОВ В КОМПЬЮТЕРНЫХ СИСТЕМАХ

Аннотация. *Проведен анализ жизненного цикла процесса, выделено граничное состояние, которое предшествует состоянию взаимоблокировки. Разработан алгоритм выявления процессов, которые находятся в граничном состоянии и могут попасть в состояние взаимоблокировки, и алгоритм прогнозирования вхождения процессов в состояние взаимоблокировки в компьютерных системах. Проведена также оценка временной сложности разработанного алгоритма.*

Ключевые слова: *взаимоблокировка, состояние процесса, алгоритм прогнозирования состояния взаимоблокировки процесса*

Вступ

В процесі експлуатації комп'ютерних систем (КС) досить часто виникає ситуація блокування процесів, що виконуються у них [1].

Окремим випадком блокування процесів є можливість їх взаємного блокування. Взаємне блокування (англ. deadlock) – ситуація в багатозадачному середовищі або системах керування базами даних (СКБД), при якій кілька процесів перебувають у стані нескінченного очікування ресурсів, зайнятих цими процесами.

Виникнення взаємних блокувань процесів призводить до збільшення часу їхнього виконання (може зростати до нескінченості), до не ефективного використання ресурсів КС (порожні цикли очікування).

Причинами даного явища є помилки синхронізації програмного забезпечення (ПЗ), невідповідність ПЗ його специфікації.

Для визначення і усунення взаємоблокування процесів у комп'ютерних системах розроблено ряд методів і засобів [1 – 3].

Проте на практиці їх не завжди доцільно застосовувати для вирішення задачі взаємоблокування, оскільки частина з них носить лише теоретичний характер і не може бути реалізованою в сучасних операційних системах [1 – 5]. Інша частина при реалізації стає достатньо громіздкою і ресурсоємною.

Тому розробники сучасних ОС (операційних систем) сімейств Windows та Unix, а також розробники сучасних СКБД не включають відомі алгоритми уникнення взаємоблокувань процесів.

Постановка задачі

За умови виконання в системі невеликої кількості процесів відсутність таких засобів була допустима.

Проте стрімкий розвиток апаратних засобів, зростання об'єму та складності ПЗ, яке займається вирішенням масштабних та відповідальних задач, не повинно дозволяти виникнення взаємоблокування, що в свою чергу вимагає розробки нових підходів до вирішення цієї задачі.

Для усунення суттєвих недоліків відомих методів та алгоритмів вирішення проблеми взаємоблокування необхідно розробити метод прогнозування взаємоблокування процесів, що враховує їхній життєвий шлях [6 – 7] та відповідні алгоритми і засоби, та провести оцінку часової складності розроблених засобів.

Життєвий цикл процесу

Процес – це система дій, що реалізує певну функцію в обчислювальній системі й оформлена так, що керуюча програма обчислювальної системи може перерозподіляти ресурси цієї системи з метою забезпечення багатозадачності [1].

Позначимо множину виконуваних процесів, як $A = \{a_i\}_{i=1}^{y}$, де y – кількість процесів.

Ресурс комп'ютерної системи – засіб

комп'ютерної системи, що може бути виділений процесу обробки даних на певний інтервал часу.

Позначимо множину наявних ресурсів КС, як $RE = \{re_j\}_{j=1}^{x}$, де x – кількість видів ресурсів.

До ресурсів КС віднесемо наявну пам'ять, процесори, пристрої вводу/виводу, засоби взаємовиключення (м'ютекси, семафори та ін.), а також дані, необхідні для роботи процесів (файли в пам'яті та на зовнішніх носіях, результати обчислень інших процесів).

Кожен процес від моменту створення до моменту завершення проходить через ряд станів (рис. 1).

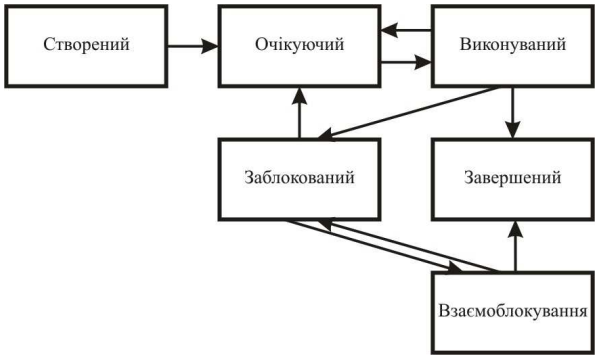

Рис. 1. Діаграма станів процесу, що включає стан взаємоблокування

Під сигнатурою процесу будемо розуміти сукупність його характеристик, яка однозначно ідентифікує стан процесу в КС в певний момент часу t:

$$a_i(t) \to (a_{i_1}^t, a_{i_2}^t, ..., a_{i_z}^t), \qquad (1)$$

де $a_i(t) \in A$ – поточний процес, $a_{i_1}^t$, … , $a_{i_z}^t$ – характеристики процесу в поточний момент часу (параметри та ресурси, які використовує процес в даний момент).

До характеристик процесу, що формують сигнатуру, віднесемо наступні: ідентифікатор процесу, ідентифікатор батьківського процесу, ідентифікатор користувача, якому належить процес, пріоритет процесу, квоти процесу (кількість пам'яті і процесорний час доступні процесу), перелік отриманих ресурсів.

Оскільки до складу сигнатури процесу входять його унікальні характеристики, то в один і той самий момент часу в системі не існує двох абсолютно однакових сигнатур.

Життєвий цикл процесу можна подати у вигляді послідовності станів, через які проходить цей процес. Перехід із стану в стан відбувається через зміну певних параметрів, якими характеризується процес. Зміна параметрів процесу відбувається з ряду причин: дії ОС, дії інших процесів, виконання власного програмного коду. Стан кожного окремого процесу буде впливати на стан КС в цілому.

Позначимо стан процесу через w, причину зміни параметрів через r, а перехід із стану в стан через $w_i \xrightarrow{\text{зміна параметрів з причини } r_j} w_{i+1}$. Тоді життєвий цикл процесу буде мати вигляд (2):

$$a_i : w_0 \xrightarrow{r_j} w_1 \xrightarrow{r_j} ... \xrightarrow{r_j} w_{k-1} \xrightarrow{r_j} w_k, \qquad (2)$$

де $w_0 \in W$ – початковий стан процесу (стан «створений»); $w_k \in W$ – кінцевий стан процесу (стан «завершений»); W – множина програмних станів процесу (рис. 1); $r_j \in R$ - множина можливих причин зміни параметрів процесу (j = 1, 2, 3 …).

Враховуючи визначення сигнатури та (1) і (2), життєвий цикл кожного процесу можна подати у вигляді:

$$\begin{aligned} a_i : (a_{i_1}^{t_0}, a_{i_2}^{t_0}, ..., a_{i_z}^{t_0}) \xrightarrow{r_j} \\ \xrightarrow{r_j} (a_{i_1}^{t_1}, a_{i_2}^{t_1}, ..., a_{i_z}^{t_1}) \xrightarrow{r_j} ..., \\ ... \xrightarrow{r_j} (a_{i_1}^{t_k}, a_{i_2}^{t_k}, ..., a_{i_z}^{t_k}) \end{aligned} \qquad (3)$$

У стан взаємоблокування можуть потрапляти процеси, що взаємодіють між собою у багатозадачних КС в певний момент часу. До потрапляння у стан взаємоблокування процеси протягом свого життєвого циклу знаходяться в інших станах, а саме стан «створений», стан «очікуючий», стан «виконуваний», стан «заблокований» (рис. 1).

У стан взаємоблокування процеси потрапляють, як правило, із стану «заблокований». Отже, у даний момент часу серед множини процесів можна виділити підмножину процесів, які можуть в наступний момент часу потрапити до стану взаємоблокування.

Перед входженням у стан взаємоблокування процес буде знаходитись у певному «граничному» стані [7], після якого ймовірність переходу у стан взаємоблокування буде високою.

Взаємоблокування процесів призводить до часткової або повної втрати функційної здатності КС. Тому вважатимемо стан взаємоблокування процесів неробочим станом КС, а інші стани процесів – робочим станом КС.

Віднесемо до робочого стану такі стани процесу: стан «створений», стан «виконуваний», стан «завершений» та стан «очікуючий». До «граничного» стану віднесемо стан «заблокований».

Представимо шлях, яким процес потрапляє у стан взаємоблокування, у вигляді наступної схеми (рис. 2).

Як видно із схеми, виникнення взаємоблокування процесів можливе лише для частини процесів, що знаходяться у граничному стані.

При переході процесів до граничного стану відбувається зміна їхніх параметрів.

Розглянемо життєвий цикл процесу. В момент створення (стан «створений») процес знаходиться у робочому стані, йому надана частина ресурсів системи. Позначимо цей стан процесу, як $s_{\text{роб}}(t)$. В певний момент часу процесу для подальшого виконання необхідний додатковий ресурс системи, який на даний час є недоступний. Відбувається перехід процесу до стану «заблокований», тобто процес потрапляє у граничний стан.

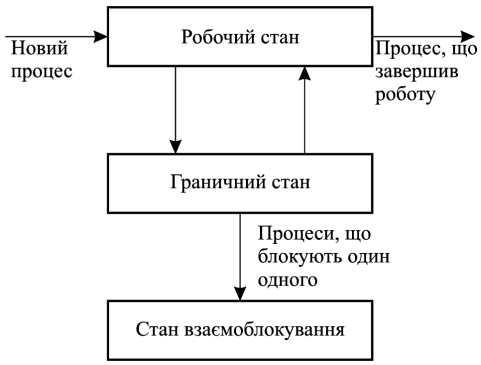

Рис. 2. Схема переходу процесів у стан взаємоблокування

Позначимо цей стан процесу, як $s_{\text{гран}}(t)$, а перехід до даного стану, як $s_{\text{роб}}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{\text{гран}}(t)$. При задоволенні потреб процесу у ресурсах він повертається у робочий стан, тобто здійснює перехід $s_{\text{гран}}(t) \xrightarrow{\text{надання ресурсу } re_i} s_{\text{роб}}(t)$. Коли необхідний ресурс отримати неможливо по причині його використання іншим процесом, то процес залишається у граничному стані.

Якщо ж другий процес в свою чергу очікує ресурс, що зайнятий першим процесом, то вони обидва потрапляють у стан взаємоблокування. Позначимо цей стан процесу, як $s_{\text{взаємоблокування}}(t)$, а перехід до даного стану, як $s_{\text{гран}}(t) \xrightarrow[\text{утримування ресурсу } re_j]{\text{очікування ресурсу } re_i} s_{\text{взаємоблокування}}(t)$.

Таким чином, враховуючи (3), життєвий цикл процесу буде мати один з наступних виглядів.

1) Процес створений, йому надано всі необхідні для завершення роботи ресурси, він виконується і завершується (процес весь час знаходиться в робочому стані $s_{\text{роб}}(t)$), тобто:

$$a_i : s_0 \xrightarrow{r_j} s_1 \xrightarrow{r_j} s_k, \qquad (4)$$

де $s_0 \in s_{\text{роб}}$, $s_1 \in s_{\text{роб}}$.

2) Процес створений, йому надано частину ресурсів, необхідних для завершення роботи, він потребує додаткових ресурсів, через деякий час отримує їх, виконується і завершується (процес спочатку знаходиться в робочому стані $s_{\text{роб}}(t)$, потім переходить .. в граничний стан $s_{\text{гран}}(t)$, потім $s_{\text{гран}}(t) \xrightarrow{\text{надання ресурсу } re_i} s_{\text{роб}}(t)$ знову в робочий стан $s_{\text{роб}}(t)$), тобто:

$$a_i : s_0 \xrightarrow{r_j} \ldots \xrightarrow{r_j} s_l \xrightarrow{r_j} \ldots \xrightarrow{r_j} s_k, \qquad (5)$$

де $s_0 \in s_{\text{роб}}$, $s_l \in s_{\text{гран}}$, $s_k \in s_{\text{роб}}$.

3) Процес створений, йому надано частину ресурсів, необхідних для завершення роботи, він потребує додаткових ресурсів, які утримує інший процес, який в свою чергу потребує ресурсів першого процесу (процес спочатку знаходиться в робочому стані $s_{\text{роб}}(t)$, потім переходить $s_{\text{роб}}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{\text{гран}}(t)$ в граничний стан $s_{\text{гран}}(t)$, із якого відбувається перехід $s_{\text{гран}}(t) \xrightarrow[\text{утримування ресурсу } re_j]{\text{очікування ресурсу } re_i} s_{\text{взаємоблокування}}(t)$ до стану взаємоблокування).

$$\begin{cases} a_i : s_0 \xrightarrow{r_j} \ldots \xrightarrow{r_j} s_l \xrightarrow{r_j} s_x \\ a_m : s_0 \xrightarrow{r_j} \ldots \xrightarrow{r_j} s_n \xrightarrow{r_j} s_x \end{cases}, \qquad (6)$$

де .., $s_l \in s_{\text{гран}}$, $s_n \in s_{\text{гран}}$, $s_x \in s_{\text{взаємоблокування}}$.

Із розглянутих вище можливих варіантів поведінки процесів критичним для КС є останній, при якому процеси потрапляють у стан взаємоблокування

Алгоритм прогнозування потрапляння процесів у стан взаємоблокування

Як видно із рис. 2, прогнозування стану процесу включає два етапи: визначення множини процесів, що можуть потрапити у стан взаємоблокування; виділення із множини потенційних процесів групи процесів, що потраплять у стан взаємоблокування.

Таким чином, алгоритм прогнозування стану процесу буде містити дві частини.

Згідно [6] до моделі прогнозування стану процесів входять наступні величини:

$$M = \langle A, S, D, P, R \rangle, \qquad (7)$$

де A – множина сигнатур процесів, що виконуються в КС у даний момент;

S – впорядкована послідовність характеристик комп'ютерної системи (загальний обсяг оперативної пам'яті, зовнішньої пам'яті, обсяг вільної в даний момент пам'яті, кількість периферійних пристроїв);

D – підмножина сигнатур процесів, що знаходяться у стані, наближеному до стану взаємоблокування (у граничному стані);

P – множина правил, на основі яких визначається група процесів, що потраплять у стан взаємоблокування;

R – вектор ймовірностей переходу у стан взаємоблокування процесів із підмножини D.

Алгоритм 1. Виявлення потенційних процесів, що можуть потрапити у стан взаємоблокування (виявлення процесів, що знаходяться у граничному стані).

1. Якщо $a_k \in A$ потребує ресурс $r_i \in R$, то перехід до 2.

2. Якщо $a_k \in A$ знаходиться в стані $s_{\text{роб}}(t)$, то перехід до 3, інакше перехід до 4.

3. Виконати для $a_k \in A$ перехід із робочого стану до граничного $s_{\text{роб}}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{\text{гран}}(t)$ та включити $a_k \in A$ до $D \subset A$. Перехід до 4.

4. Якщо $a_k \in A$ надано у використання ресурс $re_i \in RE$, то перехід до 5, інакше перехід до 6.

5. Виконати для $a_k \in A$ перехід із граничного стану до робочого $s_{\text{гран}}(t) \xrightarrow{\text{надання ресурсу } re_i} s_{\text{роб}}(t)$ та виключити $a_k \in A$ із $D \subset A$. Перехід до 6.

6. Якщо перевірено всю множину А, то перехід до 7, інакше перехід до 1.

7. Кінець алгоритму.

Для визначення процесів, що потраплять у стан взаємоблокування, використовується система прогнозування стану процесів, в основі якої лежить система нечіткого логічного висновку (СНЛВ).

На рис. 3 показана структура СНЛВ.

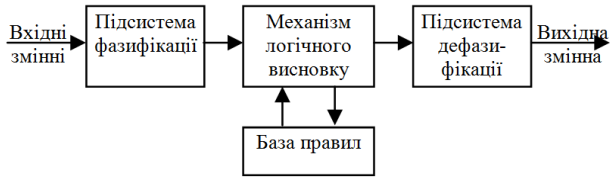

Рис. 3. Загальна схема СНЛВ

Підсистема фазифікації призначена для визначення ступеня приналежності вхідних значень .., $X = D \cup S$, $g = \overline{1,h}$ до нечітких множин входу, що являють собою лінгвістичні змінні з відповідної лінгвістичної шкали $T_{x_g} = \left\{ T_{x_g}^1, T_{x_g}^2, ..., T_{x_g}^{m_{x_g}} \right\}$, де m_{x_g} – кількість лінгвістичних змінних у шкалі для g-го входу. Необхідність у фазифікації зумовлена тим, що у СНЛВ використовуються лінгвістичні правила.

База правил, що містить лінгвістичні правила, є основою механізму логічного висновку. Механізм логічного висновку здійснює відображення вхідних нечітких множин T_{x_g} за допомогою кожного правила у вихідну T_y з набору вихідних лінгвістичних змінних $T_y = \left\{ T_y^1, T_y^2, ..., T_y^{m_y} \right\}$. Правила із множини правил $P = \left\{ P_j \right\}, j = \overline{1,n}$, що міститься в базі правил, подані у наступному форматі:

$$P_j = \text{"якщо } x_1 \in T_{x_1} \text{ і } x_2 \in T_{x_2} \ldots \text{ і } x_h \in T_{x_h}, \text{ то } y_j \in T_y \text{"} \quad (8)$$

Вихідні нечіткі множини y_i кожного правила об'єднуються в одну нечітку множину висновку $\tilde{y}$.

Після цього підсистема дефазифікації здійснює відображення нечіткої множини висновку $\tilde{y}$ у чітке число $\overline{y}$, яке буде результатом СНЛВ для заданих вхідних значень x_g.

Алгоритм 2. Виявлення процесів, що потраплять у стан взаємоблокування:

1. Проводимо нормування характеристик КС за наступною формулою:

$$P_n = 1 - \frac{P_d + P_m}{P_d \cdot P_m}, P_d \neq 0, \quad (9)$$

де P_n – черговий нормований показник;

P_d – поточне значення показника в КС;

P_m – максимальне значення показника в КС.

2. Для кожного $x_g \in X$, $X = D \cup S$, $g = \overline{1,h}$ визначаємо ступінь належності до нечітких множин входу (ступені істинності $\mu_g^j(x_g)$).

3. На основі ступенів істинності передумов $\mu_g^j(x_g)$ для кожного правила P_j, $j = \overline{1,n}$ розраховуємо ступінь його виконання α_j за формулою.

$$\alpha_j = \min\left(\mu_1^j(x_1), \mu_2^j(x_2), ..., \mu_h^j(x_h) \right). \quad (10)$$

4. На основі ступеню виконання α_j для кожного правила P_j, $j = \overline{1,n}$ розраховуємо результат його виконання.

5. На основі результату виконання кожного правила P_j, $j = \overline{1,n}$ визначаємо вихідну нечітку множину з усіченою функцією приналежності $\ddot{\mu}^j(y)$ за формулою:

$$\ddot{\mu}^j(y) = \min\left(\alpha_j, \mu^j(y) \right) \quad (11)$$

6. Вихідні нечіткі множини $\ddot{\mu}^j(y)$ згідно (12) агрегуємо в нечітку множину висновку $\tilde{y}$, що має функцію приналежності (13).

$$\tilde{y} = \max\left(\mu^j(y) \right), j = \overline{1,r} \quad (12)$$

$$\mu_{\tilde{y}} = \max\left(\ddot{\mu}^1(y), \ddot{\mu}^2(y), ..., \ddot{\mu}^r(y) \right) \quad (13)$$

7. Приводимо до чіткості нечітку множину $\tilde{y}$ за допомогою процедури дефазифікації центроїдним методом

$$\overline{y} = \frac{\int_{C_1}^{C_t} x \cdot f_{\tilde{y}}(x) dx}{\int_{C_1}^{C_t} f_{\tilde{y}}(x) dx} \quad (14)$$

8. Встановлюємо для R_k значення .

9. Повторюємо кроки 1-8 k разів.

Кінець алгоритму.

Часова складність алгоритму прогнозування стану процесу

Оцінка ефективності алгоритму включає як якісний так і кількісні показники. Якісним показником є те, чи вирішує алгоритм поставлену задачу, чи ні. Кількісними показниками є час виконання та ємність алгоритму.

В даному випадку критичними показниками є час, за який буде виконуватись прогнозування настання ситуації взаємоблокування, та задіяні при цьому ресурси системи. Отже, критерієм для оцінки ефективності алгоритму буде час.

Згідно загальноприйнятих підходів основним показником часової ефективності є часова складність (ЧС) – порядок складності алгоритму O(f).

Для *алгоритму 1* ЧС становить O(k), де k - кількість наявних в системі процесів, оскільки k разів повторюються однакові лінійні дії.

Для *алгоритму 2* ЧС становить $O\left(j \cdot \log(n) \right)$, де n – кількість процесів, що знаходяться у граничному стані; j – кількість правил, що містяться у базі правил.

На рис. 4 наведена ЧС алгоритму прогнозування стану процесу для різної кількості процесів та правил.

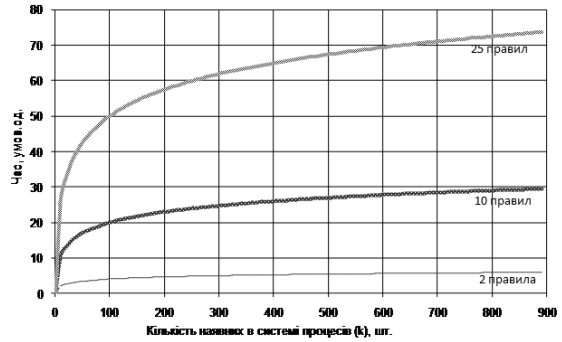

Рис. 4. Часова складність алгоритму прогнозування стану взаємоблокування процесів в КС

Як видно із рис. 4, складність алгоритму зростає нелінійно при збільшенні кількості процесів у системі, оскільки, на відміну від відомих алгоритмів вирішення задачі взаємоблокування, розроблений алгоритм прогнозування стану процесів використовує компоненти нечіткої логіки, що робить його гнучким у використанні.

Висновки

В роботі запропоновано алгоритм прогнозування стану взаємоблокування процесів в КС та проведено оцінку його часової складності.

На відміну від відомих методів та алгоритмів він використовує компоненти нечіткої логіки, що дозволяє виявляти два та більше процесів, які потрапляють у стан взаємоблокування, і при цьому не робить алгоритм громіздким, що дозволяє використовувати його у сучасних операційних системах.

Список використаної літератури

1. Савенко О. С. Дослідження та аналіз блокування процесів в комп'ютерній системі / О. С. Савенко, Ю. П. Кльоц, С. В. Мостовий // Вісник ХНУ. – Хмельницький: – 2007. – № 3.– Т. 1. – С. 248 – 251.

2. Coffman E.G., Elphick M.J., and Shoshani A., (1971), System Deadlocks, *Computing Surveys, June 1971,* Vol. 3, No. 2, pp. 67 – 78.

3. Isloor S.S., and Marsland T.A., (1980), The Deadlock Problem: An Overview, *Computer,* No. 9, Vol. 13, pp. 58 – 78.

4. Nima Kaveh, and Wolfgang Emmerich, (2001), Deadlock Detection in distribution Object Systems, *Software Engineering Notes, September 2001,* Vol. 26, No. 5, pp. 44 – 51.

5. Saddek Bensalem, Jean-Claude Fernandez, Klaus Havelund, and Laurent Mounier, (2006), Confirmation of Deadlock Potentials Detected by Runtime Analysis, *International Symposium on Software Testing and Analysis,* pp. 41 – 50.

6. Савенко О. С. Модель прогнозування стану процесів в комп'ютерній системі / О. С. Савенко, С. В. Мостовий // Радіоелектронні і комп'ютерні системи. – Харків : – 2008. – № 5 (32). – С. 109 – 115.

7. Савенко О. С. Система прогнозування стану процесів в персональному комп'ютері / О. С. Савенко, С. В. Мостовий. // Збірник праць VIII міжнародної конференції «Інтелектуальний аналіз інформації ІАІ-2008». – К. : – 2008. – С. 308 – 314.

8. Savenko O.S., Klots Y.P., and Mostovoy S.V., (2007), Research and Analysis of Lock Processes in a Computer System, *Visnuk KHNU*, Khmelnitsky, Ukraine, No. 3, Vol. 1, pp. 248 – 251 (In English).

Отримано 25.05.2015

References

1. Savenko O.S., Klots Y.P., and Mostovoy S.V. Doslidzhennya ta Analiz Blokuvannya Protsesiv v Komp'yuterniy Systemi [Research and Analysis Processes in a Computer Blocking System], (2007), *Visnuk KHNU Publ.,* Khmelnitsky, Ukraine, No. 3., Vol. 1, pp. 248 – 251 (In Ukrainian).

2. Coffman E.G., Elphick M.J., and Shoshani A., (1971), System Deadlocks, *Computing Surveys, June 1971,* Vol. 3, No. 2, pp. 67 – 78 (In English).

3. Isloor S.S., and Marsland T.A., (1980) The Deadlock Problem: An Overview, *Computer,* No. 9, Vol. 13, pp. 58 – 78 (In English).

4. Nima Kaveh, and Wolfgang Emmerich, (2001), Deadlock Detection in Distribution Object Systems, *Software Engineering Notes, September 2001,* Vol. 26, No. 5, pp. 44 – 51 (In English).

5. Saddek Bensalem, Jean-Claude Fernandez, Klaus Havelund, and Laurent Mounier, (2006), Confirmation of Deadlock Potentials Detected by Runtime Analysis, *International Symposium on Software Testing and Analysis,* pp. 41 – 50 (In English).

6. Savenko O.S., and Mostovoy S.V. Model' Prohnozuvannya Stanu Protsesiv v Komp'yuterniy Systemi [Model of Forecasting of Processes in a Computer System], (2008), *Radio Electronic and Computer Systems*, Kharkiv, Ukraine, Vol. 5 (32), pp. 109 – 115 (In Ukrainian).

7. Savenko O.S., and Mostovyy S.V., Systema Prohnozuvannya Stanu Protsesiv v Personal'nomu Komp'yuteri [The System of Forecasting of Processes in PC], (2008), *Proceedings of VIII International Conference IAI-2008,* Kiev, Ukraine, pp. 308 – 314 (In Ukrainian).

8. Savenko O.S., Klots Y.P., and Mostovoy S.V., (2007), Research and Analysis of Lock Processes in a Computer System, *Visnuk KHNU*, Khmelnitsky, Ukraine, No. 3, Vol. 1, pp. 248 – 251 (In English).

Кльоц
Юрій Павлович,
к.т.н., начальник інформаційно-комп'ютерного центру, доц. каф. системного програмування Хмельницького нац. ун-ту,
т.0677740840.
E-mail: sprklyots@gmail.com

Мостовий
Сергій Володимирович,
ст. викладач каф. системного програмування Хмельницького нац. ун-ту, 29000,
т.0688331573.
E-mail: sprmostovuy@gmail.com