**УДК 681.3.06**

**В.В. Кулібаба**
Харківський національний університет радіоелектроніки,
кафедра “Соціальна інформатика”

# ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ СКЛАДНИХ ПРОГРАМНИХ СИСТЕМ НА ОСНОВІ АГРЕГАТІВ З ВИКОРИСТАННЯМ СИСТЕМНО-ОБ’ЄКТНОГО ПІДХОДУ



***Use of the combined model and system-object approach will allow the researcher to combine process of development of object-oriented and imitating models, to examine process of construction of models as a single unit. Use of imitating modeling for development of program systems allows: to increase quality of developed system, to reduce time of development, to increase efficiency of work. In clause are given corrected determining rules of construction and functioning of imitating model, and also the criteria of an estimation of the created model are given.***

***Використання комбінованої моделі і системно-об’єктного підходу дозволить досліднику сполучити процес розробки об’єктно-орієнтованої та імітаційної моделей, розглядати процес побудови моделей як єдине ціле. Використання імітаційного моделювання для розробки складних програмних систем дозволяє: підвищити якість розроблювальної системи, скоротити час розробки, підвищити ефективність праці. Наведено правила побудови і функціонування імітаційної моделі складних програмних систем.***

## 1. ВСТУП

Стрімкий ріст обсягів інформації [1], який спостерігається сьогодні, змінив відношення до використання обчислювальних ресурсів, що, у свою чергу, змінило роль програмного забезпечення в суспільстві. Інформація перетворилася в один з основних ресурсів сучасного суспільства, програмне забезпечення (системи обробки інформації, автоматизовані системи керування, інтелектуальні автоматизовані системи керування, експертні системи тощо) – в основний засіб обробки інформації. Тобто, з ростом цінності інформації актуальними задачами стають задачі підвищення якості й ефективності створюваного програмного забезпечення [2–5].

Перебудова ведення бізнесу з використанням інформаційних ресурсів визначила великі (складні) програмні системи (новий клас програмних систем), в яких багато складних задач неможливо вирішити простими методами через причини природного і штучного характеру [3, 4, 6–7]. Зниження впливу фактора “складність” на процес розробки є актуальною задачею, яку розв’язують різними способами, один із яких – поетапна (ітеративна) розробка (каскадний процес з поверненням, спіральна модель, RUP) [4, 8]. В ітеративній розробці кожний з циклів містить етапи класичного технологічного процесу (аналіз, проектування, кодування, тестування, експлуатація) [4, 5, 8], розділені так, що проведення наступного циклу дозволяє уточнювати результати попереднього. Така організація процесу розробки дозволяє знизити вплив на систему зміни вимог замовника,

чіткіше формулювати і точніше дотримуватися поставленої задачі, врахувати особливості реалізації тощо. Серед недоліків ітераційного процесу розробки можна виділити такі:

- складно контролювати розробку (неможливо оцінити час, який знадобиться для всієї розробки і для проведення одного циклу розробки);
- зберігається ймовірність розробки програмного продукту, що не відповідає вимогам замовника (використовувані методи аналізу і проектування впливають на розробку);
- зберігається імовірність морального старіння програмного продукту тощо [2, 3, 7].

Використання автоматизованих інформаційних засобів (CASE-засобів) [4, 5, 8], дозволило підвищити продуктивність праці, скоротити час розробки продукту. Однак, зважаючи на те, що більшість CASE-засобів розраховані на підтримку рішення, прийнятого розроблювачем, вони не допомагають йому "знаходити" рішення, тому їхнє використання не вирішує вищеописаних задач [9]; вирішення цих задач вимагає внесення змін у технологію і процес виробництва.

## 2. ПОСТАНОВКА ЗАДАЧІ

Розв'язання вищеописаних задач можливе, якщо:

1) розглядати процес розробки програмної системи комплексно;

2) сполучити розгляд статичних і динамічних аспектів моделювання системи;

3) будувати розробку системи на основі отриманих знань і досвіду з попередніх розробок;

4) скоротити час, необхідний на аналіз, проектування, кодування і тестування системи, тим самим прискоривши кожний з етапів ітераційного процесу;

5) скоротити час між етапами.

Цього можна домогтися, використовуючи на етапах аналізу і проектування системно-об'єктний (системологічний) підхід [1] та імітаційне моделювання.

## 3. ІМІТАЦІЙНА МОДЕЛЬ

### 3.1. Математичний апарат

При розробці імітаційної моделі важливим етапом є вибір математичного апарата (безперервно-детерміновані моделі – D-схеми, дискретно-детерміновані моделі – F-схеми, дискретно-стохастичні моделі – P-схеми, безперервно-стохастичні моделі – Q-схеми, мережні моделі – N-схеми, комбіновані моделі – A-схеми, або агрегатні схеми), що враховує специфіку розглянутої предметної області і поводження системи. Сучасні реалізації існуючих математичних моделей характеризуються універсальністю, що дозволяє стверджувати, що «термины "смешанная система", "событийно-управляемая система", "агрегативная система", "непрерывно-дискретная система", "система переменной структуры", "гибридная система" являются синонимами» [10]. Наприклад, у [11] виділяють комбіновані моделі як універсальний підхід в імітаційному моделюванні – підхід, "позволяющий описывать поведение непрерывных и дискретных, детерминированных и стохастических систем" [11, с. 75], який у принципі не відрізняється від існуючих математичних апаратів (D-, F-, P-, N-, Q-схем). Отже, використання A-схем дозволить уніфікувати алгоритми імітації і використовувати стандартні методи обробки й аналізу результатів моделювання.

Базовим елементом у комбінованих моделях є агрегат, який за певних умов може розглядатися як A-схема, що складається з агрегатів нижнього рівня. Агрегат представляється у вигляді:

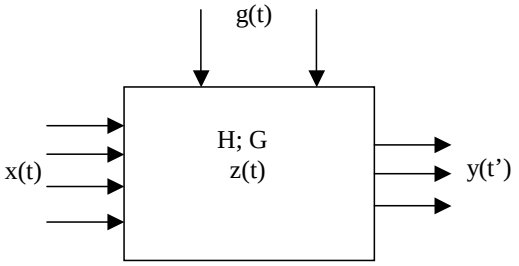


*Рис. 1. Схематичне зображення агрегату*

$A = \{T, X, Y, Z, \Gamma, G, H\}$ (на рис. 1 схематично зображено агрегат), де $A$ – агрегат (чи А-схема); $T$ – інтервал моделювання $T = [0, T_i]$, що задає множину станів $z \in Z$; $X$ – множина вхідних сигналів $x = [x^1, x^2, x^3, x^4, ..., x^n]$, де n – число контактів.

Моменти надходження вхідних сигналів визначаються через тимчасову послідовність $\{t_j\}$, $t_j \in T$. Послідовність прийому вхідних сигналів задається через послідовність подій $\langle t_j, x_j \rangle$, $x_j \in X$;

$Y$ – множина вихідних сигналів, обумовлена аналогічною множиною вхідних сигналів $X$. Послідовність видачі вихідних сигналів $\langle \xi_i, y_i \rangle, y_i \in Y, \xi_i \in T$;

$Z$ – множина станів $z = \{z_1, z_2, z_3, z_4, ..., z_j\}$, де $\{z(t)\}$ – множина поточних станів, $\{z(0)\}$ – множина початкових станів, які характеризуються вхідними і вихідними сигналами;

$\Gamma$ – множина керуючих сигналів $g = \{g^1, g^2, g^3, g^4, ..., g^n\}$, де n – число вхідних керуючих контактів. Послідовність прийому керуючих сигналів задається через множину $\langle \tau_i, g_i \rangle, g_i \in \Gamma$, $\tau_i \in T$;

$G$ – множина операторів перевірки умови $y = G[z(t'), g]$;

$H$ – оператор переходу, що змінює значення стану агрегату $z(t) = H[z(0), t]$, має випадковий характер. Залежно від вхідних (даних і керуючих сигналів) і вихідних сигналів розрізняють оператори виду:

$U$ – $z(t) = U[z(\tau'), g', t]$ (поточний стан агрегату);

$V'$ - $z(t) = V'[z(t'), x, g]$;

$V''$ $z(\tau) = V''[z(\tau'), g]$;

$V$ $z(t) = V[z(t'), x]$;

$W$ $z(t) = W[z(t'), g]$.

Отже, будь-якому $z(0)$ відповідає множина значень $z(t)$ з деяким законом розподілу.

Виділяють також кусково-лінійний агрегат, агрегати, для яких спрощене рішення задачі моделювання. Кусково-лінійний агрегат – агрегат з лінійним оператором $U$, зі структурованою множиною Z, що розглядає сигнал g як вид вхідних сигналів [11]. Кусково-лінійний агрегат підходить для моделювання об'єктно-орієнтованих систем через те, що у цих системах складно відокремити керуючий сигнал від даних, тоді як агрегат більше підходить для моделювання систем, побудованих на структурній методології [12].

Використанню комбінованих моделей при моделюванні об'єктно-орієнтованих систем заважає орієнтованість даного математичного апарата на функціональну декомпозицію. Тобто, при використанні неадаптованої комбінованої моделі користувач не отримає явних переваг: розроблювані системи будуть мати недоліки, яких не мають об'єктно-орієнтовані системи, використовуваний алгоритмічний аналіз поступається при розробці інформаційних систем об'єктно-орієнтованому аналізу тощо [1, 3, 9,12].

Тому задача розвитку комбінованих моделей полягає у використанні при формалізованій розробці моделі об'єктно-орієнтованої декомпозиції. Існуючі відмінності між алгоритмічним і об'єктно-орієнтованим моделюванням відображені в [1]. У цій статті пропонується спільне використання при моделюванні А-схем [11] і системно-об'єктного підходу [1].

### 3.2. Підвищення ефективності математичного апарата за рахунок використання системно-об'єктного підходу

Системно-об'єктний підхід містить останні досягнення сучасної науки в галузі системології, системологічного класифікаційного аналізу. Даний підхід є цілком сумісним з об'єктно-орієнтованою методологією. Використання системно-об'єктного підходу в об'єктно-орієнтованій методології дозволить підвищити якість аналізу, формалізувавши дану процедуру. Приклад практичного застосування системно-об'єктного підходу при проектуванні організаційних систем наведено у [13]. Одним з інструментів розробленого системно-об'єктного підходу є УФО-аналіз, що дозволяє представити систему (кожен елемент системи) з погляду УФО-елементів (елемент типу "вузол – функція – об'єкт"), які дослідник може відокремити за допомогою розробленої моделі подання знань і класифікації УФО-елементів. УФО-аналіз дозволяє формалізувати опис системи (елементів системи), надавши повну інформацію про структуру, функціонування і поводження системи (елемента системи).

**Твердження 1**

Елемент (явище, процес і т. д.), що є частиною програмної системи (чи програмною системою), розглянутий з позиції системології як система з урахуванням його міри системності, є УФО-елементом і може бути перетворений до подання у вигляді "вузол – функція – об'єкт".

**Наслідок твердження 1**

Будь-який програмний компонент може бути перетворений на УФО-елемент.

**Порівняння агрегату й УФО-елемента**

| | Агрегат | УФО-елемент |
|---|---|---|
| 1 | 2 | 3 |
| Розуміння системи | Система розглядається як агрегат із входами і виходами. Для того щоб підкреслити складність розглянутого агрегату, використовують термін А-схема | Система розглядається як УФО-елемент із проточними вузлами (входами і виходами). УФО-елемент дозволяє відображати складні взаємодії елементів системи (зв'язок, відносини, функції) |
| Розуміння об'єкта | Агрегати виділяються за допомогою системного аналізу і на підставі результатів імітаційного моделювання. Описується вхідними і вихідними сигналами, особливими станами, операторами перевірки і переходів | УФО-елемент виділяється за допомогою системно-об'єктного підходу. Описується в термінах "вузол – функція – об'єкт", де "вузол" визначає інформацію, що трансформується через об'єкт, якою можуть бути дані чи задачі (сигнал керування) залежно від розглянутої абстракції. "Функція" визначає характер "трансформування" інформації. "Об'єкт" – система, реалізуюча "функції" і "вузли" |

**Порівняння агрегату й УФО-елемента** (*продовження*)

| 1 | 2 | 3 |
|---|---|---|
| Вхід<br>Вихід | Описується безліччю сигналів, змістовно вхідні і вихідні дані ніде не описуються. Контроль одержуваних і переданих даних здійснюється оператором G, перетворення даних контролюється оператором H | Описується в розумінні “вузлів”. Ставиться акцент на значеннєвий зміст вузла (рис. 2). Визначено правила з’єднання УФО-елементів |
| Керуючий сигнал | Задається на безлічі станів, вхідних даних і оператора перетворення H. Керуючий сигнал – дія до зміни стану системи | Керуючий сигнал не відокремлюється. Можна припустити, що ним може бути функціональний запит до виконання УФО-елементом своєї задачі в системі і становити один з варіантів вузлів |
| Оператор перетворення стану (оператор переходу) | Необхідно визначати для кожного типу вхідних і вихідних даних, що вимагає виявлення необхідних даних та облік можливого діапазону значень | Для УФО-елемента оператор перетворення подано у вигляді конкретної функції, що поєднує безліч вхідних і вихідних вузлів, типи і діапазон значень – характеристика вузла (зазначається у моделі подання знань) |
| Стан | Визначається на безлічі Z, z(t) залежить від оператора H, вхідних і вихідних сигналів | В УФО-елементі стан не виділяється як окрема характеристика системи |
| Час | Задається безліччю T | Не розглядається |
| Об’єднання елементів у систему | Визначається існуванням для кожного об’єднання чи агрегатів вхідних і вихідних сигналів в агрегаті оператора сполучення | Спосіб об’єднання УФО-елементів визначається “правилом композиції”. “Правило балансу” – контролює коректність утвореного вузла. “Правило реалізації” – можливість існування даного вузла |

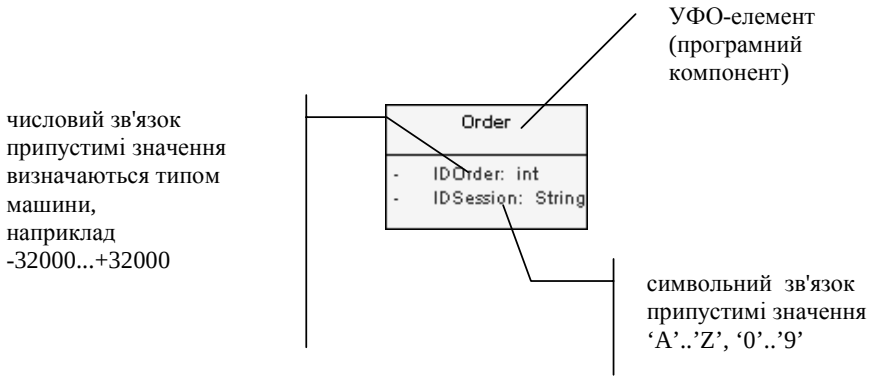


*Рис. 2. Приклад елемента з виділеними зв’язками, безліч припустимих значень для зв’язку задається в моделі знань*

З таблиці видно, що для підвищення ефективності використання як А-схем, так і УФО-підходу в імітаційному моделюванні складних програмних систем необхідно їх синтезувати. Використання комбінованих моделей у системно-об’єктному підході дозволило формально описати динаміку процесів, що протікають у системі, вказало на

необхідність розгляду для УФО-елементу стану і тимчасової характеристики. Використання системно-об'єктного підходу для А-схем вирішує проблеми розробки моделі (статичної) системи [12]: модернізація оператора перетворення й операторів сполучення, завдання безлічі вхідних, вихідних і керуючих сигналів (описати "ідеальні канали"), рішення задачі об'єктно-орієнтованого аналізу системи.

### 3.3. Розробка імітаційної моделі

Процес імітаційного моделювання умовно можна розбити на кілька етапів:

- розробка схеми сполучення;
- приведення схеми-сполучення до стандартної форми (до форми, яка описується за допомогою модифікованих агрегатів, А-схеми);
- верифікація моделі;
- ініціалізація вихідних даних для моделювання;
- імітація функціонування елемента і взаємодії елементів;
- обробка й аналіз результатів моделювання.

Схема сполучення – модель програмної системи. Як моделі, що підходять для одержання схеми сполучення, керуючись ствердженням 1 і наслідком з твердження 1, можна виділити:

- УФО-моделі;
- моделі (проектні моделі), розроблені за допомогою існуючих графічних мов моделювання UML, IDEF і ін.;
- моделі (програма), розроблені за допомогою мов програмування.

УФО-модель може бути створена за допомогою інструментарію UFO-toolkit[13], або розробляємого інструментарію імітаційного моделювання, що створюється з використанням досвіду попередніх розробок [14].

Процес моделювання з використанням системно-об'єктного підходу описаний у [1]. При переході від моделі до стандартної форми необхідно верифікувати модель, перевірити повноту моделі, відсутність незв'язаних елементів, правильне з'єднання вузлів у моделі, ввести обмеження на безліч припустимих значень для кожного зв'язку і т. д.

Використання моделей, відмінних від УФО-моделі, вимагає приведення їхнього опису до термінів УФО-моделі, а потім до стандартної форми. Процес адаптації здійснюється на основі розробленої моделі подання знань, що містить набір шаблонів, які визначають елемент моделі, характеристики вузлів та функцій елемента.

Стандартна форма представлення моделі – модель, в якій кожен агрегат (об'єкт) представлений у вигляді УФО-елемента. При представленні моделі в стандартній формі необхідно вирішити ряд практичних задач:

- спосіб опису зв'язку (спосіб опису вихідних сигналів, спосіб опису адресації сигналів, спосіб задання безлічі значень сигналу);
- спосіб комбінування агрегатів (з'єднання агрегатів, порядок функціонування зв'язків агрегату, безліч припустимих станів агрегату).

Для розв'язання вищеописаних задач пропонується скористатися правилами приєднання, балансу, замкнутості [1], увести ряд стверджень, що задають загальні правила роботи з агрегатами і регулюють взаємодію між агрегатами.

**Твердження 2**

Сигнал визначається на підставі інформації з моделі подання знань (рис. 2). Задається $x = \langle g', g, F \rangle$. Форма сигналу $g'$ – тип сигналу, що визначає безліч припустимих значень (вхідних чи вихідних), значення $g$ – поточне значення даного зв'язку. Функція $F$ – логічний опис сигналу (призначення).

**Правило об'єднання сигналів**

Вихідний сигнал $y_j$, де $y_j \in Y \quad Y = \bigcup_{j=1}^{m} [g_l^j]_1^n$ може бути об'єднаний із вхідним сигналом $x_i$, де $x_i \in X \quad X = \bigcup_{j=1}^{m} [g_k^j]_1^n$, якщо:

- зберігається форма сигналу (виконується правило приєднання), тобто при об'єднанні сигналів не порушується істотна властивість зв'язку типу KIND-OF форми сигналу, $x_i = \{x', ...g'_n\}$, $y_i = \{x', ...g'_m\}$ , $x' = \{g'_i\}_1^i$;
- виконується правило балансу, значення вихідного сигналу входить у діапазон припустимих значень вхідного сигналу $x_j = [g_k^j]_1^n \cap [g_l^j]_1^n$;
- функція $F(x_j)$, виконувана вихідним сигналом і функція вхідного сигналу $F'(x_j)$, тотожні $F \equiv F'$.

**Твердження 3**

Елемент системи вважається заданим, якщо в нього є хоча б по одному вхідному і вихідному контакту і між ними можна встановити взаємну відповідність.

**Правило композиції агрегатів**

Кожному вхідному контакту агрегату $A_k$ відповідає вихідний контакт агрегату $A_i$, якщо між вихідними $[Y_l^{(i)}]_1^r$ і вхідними контактами $[X_r^{(k)}]_1^m$ існує оператор сполучення $Y_l^{(i)} = R(X_r^{(k)})$, що приведе сигнали у відповідність, тобто будуть виконані правила приєднання (збереження форми сигналу) і правило балансу (збереження змісту сигналу), а також функції, виконувані сигналами, збережуть тотожність одна до однієї.

**Наслідок із правила композиції агрегатів**

Для кожного оператора сполучення $R \exists R^{-1}$ (для однозначного з'єднання агрегатів), для якого виконується $X_r^{(k)} = R^{-1}(Y_l^{(i)})$. Існуюче обмеження (кожному вихідному сигналу агрегату відповідає один вхідний сигнал) у комбінованих моделях знімається за допомогою операторів $R^h$ і $R^{h^{-1}}$, що зберігають історію викликів. Оператори $R_i^{\,h}$ і $R_i^{\,h^{-1}}$, приналежні до однієї історії викликів, варто розглядати як звичайні оператори $R$, $R^{-1}$.

**Приклад:**

Дано агрегати $A_1$ і $A_2$, перший є генератором унікального числа, другий агрегат заносить інформацію в базу на підставі унікального ключа. Кожний з агрегатів має безліч вхідних і вихідних контактів. Для агрегату $A_1$ визначений контакт $y_j = \langle g', g, F \rangle$, де $g'$ – задає форму сигналу на підставі моделі представлення знань (ціле число, діапазон значень

0..32000), $g$ – поточне значення сигналу (можливо завданням випадкового числа з діапазону припустимих значень), $F$ – функція сигналу "унікальне число". Для $A_2$ – $x_j = \langle g', g, F' \rangle$, $g'$ – форма сигналу (ціле число, діапазон значень 0..65000), $g$ – поточне значення сигналу, $F$ – унікальне число, індекс. При розв'язанні цієї задачі необхідно розглянути такі питання:

- відповідність форми сигналу;
- можливий діапазон значень;
- значення сигналу;
- виконувана функція.

Для агрегатів $A_1$, $A_2$ виконується «правило об'єднання сигналів»: у сигналів зберігається форма; заданий діапазон значень для вихідного сигналу агрегату $A_1$ входить у діапазон значень агрегату $A_2$, функції, виконувані сигналами, – тотожні. В описаному прикладі агрегати об'єднує простий оператор сполучення, що буде виконуватися завжди на зазначеній безлічі вхідних і вихідних сигналів, якщо вихідний сигнал є підмножиною вхідних сигналів. Для випадків, коли вхідний сигнал є підмножиною вихідних сигналів, їхнє з'єднання розглядається як "критичний вузол", і для цього з'єднання оператор сполучення задається з обмеженнями (ідеальний випадок – зв'язок через агрегат "перетворення").

Цей приклад ілюструє роботу правил «об'єднання сигналів» і «композиції агрегатів» на елементарних сигналах. Для простих сигналів (вони позначені у моделі подання знань як простий об'єкт даних) – оператор сполучення (зворотний оператор сполучення) простий і може бути реалізований засобами імітаційного середовища. Для більш складних інформаційних потоків оператор сполучення (і зворотний оператор сполучення) буде більш складним, враховуючим додаткові параметри, що впливають на коректність з'єднання агрегатів.

Важливим у моделюванні є процес верифікації – визначення коректності побудованої моделі, адекватність даної моделі досліджуваній. Будемо вважати модель заданою (коректною) у стандартній формі, якщо:

- для кожного сигналу виконується "правило з'єднання сигналів";
- для всіх елементів виконується "правило композиції агрегатів";
- у моделі встановлені зв'язки із зовнішнім середовищем (надсистемою).

Адекватність системи визначається на підставі проведеного імітаційного експерименту. Будемо вважати систему адекватною вимогам замовника, якщо середовище на вихідних контактах, з урахуванням посланих сигналів з вхідних контактів, одержує очікувані сигнали (аналогічно для елемента, залежно від цілей імітаційного експерименту).

Функціонування елементів і системи імітується на підставі збереження в стійкому стані з'єднань у схемі сполучення, функціонуванні операторів G і H.

З використанням УФО-елемента [1] стає можливим автоматизувати використання існуючих мір оцінки [4], тобто одержувати повну інформацію про модель: вартість, витрати на розробку, результати тестування тощо.

## 4. ВИСНОВКИ

Наведено приклад використання модифікованих комбінованих моделей для моделювання складних програмних систем. Використання системно-об'єктного підходу і модифі-

кованого математичного апарата дозволяє створювати модель з можливим подальшим її використанням як імітаційну.

Отже, вперше було розроблено імітаційну модель на основі комбінованої моделі з використанням системно-об'єктного підходу та теорії патернов [1, 15, 16], було удосконалено процес моделювання за допомогою системно-об'єктного підходу та теорії патернов. Наведено опис процесу імітаційного моделювання, правила, що описують функціонування імітаційної моделі.

*1. Маторин С.И. Анализ и моделирование бизнес-систем: системологическая объектно-ориентированная технология / Под ред. проф. М.Ф. Бондаренко. – Харьков: ХНУРЭ, 2002. – 322 с. 2. Charles C. Mann Why Software Is So Bad July/August 2002. http://www.technologyreview.com/customerservice/reprints.asp?title=Why+Software+Is+So+Bad&date=July/August%202002. 3. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. – М., СПб., 1998. 4. Орлов С. Технологии разработки программного обеспечения: Учебник. – СПб.: Питер, 2002. – 446 с. 5. Петров В.Н. Информационные системы. – СПб.: Питер, 2002. – 688 с. 6. Фредерик Брукс. Мифический человеко-месяц или как создаются программные комплексы. — Пер. с англ. — СПб.: Символ-Плюс, 1999, — 304 с. 7. Эдвард Йордон ПУТЬ КАМИКАДЗЕ. Как разработчику программного обеспечения выжить в безнадежном проекте. – М.: Издательство "Лори". 2000. – 255 с. 8. www.rational.com. 9. Кулибаба В.В. Обоснование использования методологии OMSAD при проектировании case-средств моделирования информационных систем // Проблемы бионики. – 2002. – Вып. 57. – С. 30–34. 10. Парийская Е.Ю. Сравнительный анализ математических моделей и подходов к моделированию и анализу непрерывно-дискретных систем. (DIFFERENTIAL EQUATIONS AND CONTROL PROCESSES Electronic journal, reg. N P23275 on 07.03.97) http://www.neva.ru/journal/ 11. Советов Б.Я., Яковлев С.А. Моделирование систем: Учеб. для вузов. – 3-е изд., перераб и доп. – М.: Высш. шк., 2001. – 343 с. 12. Сахаров П. Rational Rose, BPwin и другие – аспект анализа бизнес-процессов // Журн. "Директору информационной службы". #11/2000. 13. Маторин С.И, Попов А.С., Маторин В.С. Проектирование UFO-tolkit // Проблемы программирования. – 2002. – № 3–4. С. 22–25. 14. Бондаренко М.Ф., Кулибаба В.В., Соловьева Е.А. Базовый программный инструментарий для построения обучающих систем основам дисциплин "TeachConcept" // Всеукраинский межведомственный научно-технический сборник "Проблемы бионики". – 2001. – № 53. – С. 3–7. 15. Гренандер У. Лекции по теории образов. 1. Синтез образов / Пер с англ. – М.: Мир, 1979. – 384 с. 16. Гренандер У. Лекции по теории образов. 2. Анализ образов / Пер. с англ. – М.: Мир, 1981. – 448 с.*