УДК 681.03

МОДЕЛЬ ПЕРЕВІРКИ КОМПОНЕНТІВ ТА ЇХ КОМПОЗИЦІЇ У КОМПОНЕНТНУ СИСТЕМУ

А.М. Рожнов

Институт программных систем НАН України
03187, Киев, проспект Академика Глушкова, 40,
тел.: 526 4286

Пропонується модель перевірки простих і складних компонентів, як композицій простих. Розглядається загальна компонента модель, метод верифікації та композиції правильних компонентів у компонентну систему.

Model verification of the simple and the compound components as compositions of the simple are offered. General model components, methods of verification and correct compositions to component systems are considered.

Вступ

З точки зору теорії програмування, компонентне програмування має дуже багато подібного із композиційним, тому перевірку правильності компонентних систем (КС) можна розглядати виходячи з композиційного методу, застосовуючи методи доведення та верифікації композиційних КС [1–5].

Метод перевірки (верифікації) компонентів на основі композиційного підходу можна використовувати як “зверху-вниз” так і “знизу-вверх”. Спочатку доводиться правильність компонентної моделі для предметної області (про) як “зверху—вниз” так і “знизу-вверх”, а потім окремих компонентів та їх сукупностей.

Доведення правильності окремого компонента міститься у доведенні його специфікацій у не залежності від правильності суміжних підкомпонентів, а також у верифікації інтерфейсів цього компонента, як зв’язків з іншими у розподіленому середовищі. В процесі верифікації виникає необхідність у введенні додаткових, або видалених зайвих атрибутів та параметрів інтерфейсів компонента, що передаються іншим компонентам, а також у доведенні правильності вибору типів параметрів компонента шляхом перевірки реалізації операцій типу на множині значень і станів компонента.

Інша точка зору на отримання правильних компонентів є методи тестування компонентів у динаміці їх виконання [6–10]. У процесі тестування виявляються відмови і дефекти, які залишилися у компонента при його розробленні. Вони коригуються і компонент повторно перевіряється на правильність виконання.

У роботі розглядаються модель перевірки компонентів, загальна модель компонентів та підхід до композиції перевірених компонентів у КС. Умовою розгляду є опис зовнішніх і внутрішніх характеристик та тимчасових властивостей компонента, що орієнтовані на перевірку зв'язків різних типів компонентів при їх композиції у більш складні програмні структури [2, 11–13].

Підхід до перевірки компонентів та КС

Для встановлення правильності компонентів виконуються дії за верифікацією:

– простих, примітивних компонентів шляхом перевірки специфікацій, структури, даних та функцій над цими даними;

– компонентів, побудованих з перевірених підкомпонентів шляхом агрегації за такими припущеннями: компоненти верифіковані, їх операції приймаються за теореми; нові компоненти верифікуються шляхом зведення до цих теорем; усі операції над підкомпонентами не виводять їх із множини станів компонентів;

– інтерфейсів шляхом доведення факту, що даний інтерфейс є потрібний і має достатньо параметрів для зв’язків компонентів між собою.

Пропонується модель перевірки компонентів, що охоплює компоненти у просторі станів і сприяє виявленню помилок у зв’язках компонентних композицій. Модель перевірки орієнтована на створення КС із перевірених компонентів і базується на композиційних структурах, інтерфейсах, стандартних правилах композиції звичайних компонентів та компонентів повторного використання – ПВК (reusable) [14–16].

Побудова КС із окремих компонентів на основі моделі перевірки основується на загальній компонентній моделі (ЗКМ), базисом якої є граф, в вершинах якого розміщуються перевірені компоненти, а дуги – це зв’язки між компонентами. Перевірка компонентів ЗКМ включає також перевірку інтерфейсів компонентів та повідомлень.

Модель перевірки потребує вирішення наступних фундаментальних проблем:

– багаторазове використання компонентів із завданням правильної їхній специфікації;

– ідентифікацію правильних компонентів;

– формування загальної специфікації ПС із специфікацій перевірених "правильних" компонентів;

– зменшення проблеми руйнування станів компонентів у просторі середовища їх виконання.

Ці проблеми вирішуються виходячи з наступних положень:

– програмний компонент має характеристики та тимчасові властивості, які допомагають перевірки компонент;

– заданий інтерфейс компонентів використовується як механізм зв'язків компонентів та ПВК;

– компоненти розглядаються як примітивні і складні компоненти;

– перевірка властивостей складного компонента з підкомпонентів виконується за композиційним принципом.

Головне місце у композиції компонентів займає інтерфейс, опис якого виконується за наступною мовою:

< інтерфейс компонента >:: = **Component** < *ім'я_компонента* >:{< множина вихідних інтерфейсів >};
{< множина вхідних інтерфейсів >} **end**
< множина вхідних інтерфейсів > :: = < множина інтерфейсів >
< множина вихідних інтерфейсів > :: = < множина інтерфейсів >
< множина інтерфейсів > :: = ∅ | < інтерфейс; < множина інтерфейсів >;
< інтерфейс > :: = **Interface** < *ім'я_інтерфейсу* >***:*** *{*< множина функцій >*}* **end**
< множина функцій > :: = ∅ | < функція >; < множина функцій >;
< функція > :: = **function** < і*м'я_функції*>***:*** < множина параметрів > **end**
< множина параметрів > :: = < параметр > | < параметр >, < множина параметрів >
< параметр > :: = < тип > (< вид параметра >)
< вид параметра > :: = **in | out | inout**

Множина інтерфейсів компонентів у цій мові забезпечує взаємодію між компонентними процесами. Поняття < вид параметра > це: **in –** вхідний параметр, **out –** вихідний параметр, **inout –** сумісний параметр.

Під перевіркою правильності *інтерфейсів*, що задані у даній мові, розуміється перевірка коректності його параметрів, через які взаємодіють між собою компоненти.

При перевірці КС можуть бути помилки у передачі параметрів інтерфейсів іншим компонентам. Для доведення правильності їх передачі застосовуються методи математичної індукції верифікації, тестування, а також моделювання інтерфейсів без виходу в середовище його виконання.

Властивість складного компонента перевіряється на абстракції компонента, що складена з простих, що відповідають умовам середовища складних компонентів і перевірених властивостей її підкомпонентів. Якщо абстракція компонента занадто складна для перевірки, використовується композиційний підхід для декомпозиції компонентів, перевірки окремих підкомпонентів і включення перевірених компонентів та властивостей в абстракцію.

Загальна компонентна модель (ЗКМ)

ЗКМ визначається виходячи з умов середовища виконання компонента і необхідності перевірки компонента в ньому. Коли ПВК міститься в складі більш складного компонента, його перевірені властивості повинні враховувати можливості середовища. Ця модель виконується динамічно на відповідної моделі обчислень, синтаксис і семантика якої, а також властивості й композиції компонентів визначені. Взаємодія компонентів з іншими компонентами моделі теж перевіряється. Компоненти моделі задаються в мові xUML (діалект UML).

Визначення ЗКМ. Це сукупність компонентів, що перевірені окремо на правильність функціонування. Формально кожен компонент цієї моделі має вигляд:

$$C = (E, I, V, P), \quad (1)$$

де E – кодове представлення компонента, необхідне для виконання;

I – множина інтерфейсу зв'язків з іншими компонентами або повідомлення;

V – множина змінних, визначені у E і зв'язані із властивостями в множині тимчасових властивостей P;

P – множина тимчасових властивостей, що відбивають особливості середовища компонента, кожне з якого перевірене на E і є парою (p, A(p)), де p – тимчасова формула, A(p) – множина формул, визначених на множинах I і V. Властивість включається в P тільки тоді, коли вона перевірена в середовищі компонента C.

Загальний вид композиції компонентів. Композиція компонентів є сукупністю простих компонентів.

$$(E_0, I_0, V_0, P_0) , ..., (E_{n-i}, I_{n-i}, V_{n-i}, P_{n-i}), \quad (2)$$

зазначених на моделі компонента за такими правилами:

E створюється з представлень підкомпонентів E_0, E_1, …, E_{n-1} , які зв'язуються між собою через інтерфейси;

$I = \{I_0, ...,I_{n-i}\}$ – набір інтерфейсів для забезпечення взаємодії компонента C з іншими компонентами;

V – підмножина $\cup_{j=0}^{n-1} V_i$, де V_i – посилання на властивість з множини P компонента;

P – множина тимчасових властивостей, визначених на множинах I і V, і перевірених на E. Множина властивостей P перевіряється на множині кодового представлення E, з використанням $P_0 , ..., P_{n-1}$.

Модель обчислень відображає модель КС у середовище виконання у вигляді системи кінцевої множини взаємодіючих процесів. Кожен процес представляється чотирма кортежами:

$$P = (X, \sum, Q, V), \quad (3)$$

де X – множина перемінних, кожна з якої має обмежений тип;

$\sum$ – розширена модель стану , зв'язане асоціативними діями середовища;

Q – черга повідомлень у порядку надходження;

V – початковий стан значень для перемінних з X, E і порожній стан для Q.

Структурно дія задається сегментом компонента із здійсненних операторів типу: порожній оператор, оператор присвоювання, оператор передачі повідомлень, умовний оператор і складний оператор, як послідовність композицій операторів.

Виконання моделі обчислень у середовищі полягає у чергуванні переходів станів і дій процесів вигляду:

– процес готовий, якщо введений стан і не виконана дія, зв'язана зі станом чи станом, що викликається першим повідомленням з черги повідомлень.

– у будь-який даний момент один процес є запланованим з числа готових процесів.

Процеси в середовищі системи взаємодіють через асинхронну передачу повідомлень. Для заданих двох процесів P_1 і P_2 передача повідомлення від процесу P_1 до P_2 виконується оператором повідомлення, що містить дію в P_1. Цей оператор передачі повідомлення включає тип повідомлення m на множині типів повідомлень M для процесу P_2, і параметри необхідні m. Коли дія виконується, повідомлення m ставиться в чергу повідомлень для процесу P_2.

Підтримка ЗКМ. Модель ЗКМ уточнюється в залежності від умов середовища її виконання. Компонент задається аналогічно вигляду, заданому в (1), але елементи ЗКМ = (E, I, V, P) залежать від особливостей обчислювального середовища, а саме:

– E – здійсненне представлення компонента C із синтаксисом і семантикою, що відповідає обчислювано моделі E має специфікацію із множини взаємодіючих процесів;

– I – інтерфейс передачі повідомлень через інтерфейс, що взаємодіє з іншими компонентами і задається парою (R, S), де R і S відповідно – множина вхідних і вихідних типів повідомлень;

– V і P успадковують їхні визначення у ЗКМ і посилання на семантику реалізаціях E і I;

– композиція компонентів C = (E, I, V, P) складається з множини підкомпонентів вигляду (2), тобто (E_0, I_0, V_0, P_0), ..., (En-i, In-i, Vn-i, Pn-i) і виконується з урахуванням умов обчислювального середовища. Композиція складається з множини Eo, ..., En-i і має відображення вхідних повідомлень R_o ,..., R_{n-1} у вихідні повідомлення S_o,..., S_{n-1}. Якщо відображення визначене типом вихідного повідомлення s у Si у 0 (i <n), і типом вхідного повідомлення r, у Rj, 0 (j <n), то виконуються наступні кроки: перевірка відповідності списку параметрів s і r; заміна всіх s у E перемінної r, крім випадку, де повідомлення типу s, є вихідним параметром компонента C; при відображенні s у більш ніж одне вхідне повідомлення, оператор його передачі з s копіюється в кожному вхідному типі повідомлення;

– I = (R, S) виходить з множини $I_i = (R_i, S_i)$, $0 \le i < n$, R (i S відповідно) підмножини $\cup^{n-1}_{i=o} R_i$ (або $\cup^{n-1}_{i=o} S_i$). Тип повідомлення в $\cup^{n-1}_{i=o} R_i$ (або $\cup^{n-1}_{i=o} S_i$) включається в R/ S за умови, що повідомлення того типу є вихідним C, коли C взаємодіє з іншими компонентами;

– згідно V перевіряється властивість кожного компонента;

– формулювання властивостей у процесі P і їх перевірка.

Взаємодія компонентів. Семантика виконання компонентів визначається рекурсивно. Коли компонент не має ніяких підкомпонентів і виконується, то в будь-який даний момент виконується один процес. Якщо компонент має підкомпоненти, то в будь-який даний момент виконується тільки один підкомпонент.

Компоненти взаємодіють один з одним через механізм передачі повідомлень. Компонент може тільки вводити (чи виводити) повідомлення, із указівкою параметрів у його вхідних (або вихідних) інтерфейсах передачі повідомлень. ЗКМ може перевірятися і виконуватися у середовищі CORBA, DCOM і EJB.

Додаткові поняття композиції компонентів

Оскільки композиція асоціюється з широко використовуваними поняттями як інтеграція, компонування, об'єднання й інше, у якої беруть участь компоненти, каркаси тощо, то вона включає зв’язки (асоціації) між зазначеними елементами, а також твердження про усі ці елементи і їх композиції. Нехай

C1, C2, …, Cn – компоненти;

T – множина їх композицій;

θ та θ’ – теорії, що пов`язані з композиціями компонент K: $C1 \times C2 \times C3 \ldots \times Cn \rightarrow T$.

Для кожної множини параметрів S повинна виконуватися властивість if $S \in \theta$ then $K(S) \in \theta'$ і не виконується відповідно if $S \notin \theta$ then $K(S) \notin \theta'$.

Розширенням поняття компонента є *патерн* – абстракція, що містить опис взаємодії сукупності компонентів у спільній кооперативній діяльності, для якої визначені ролі учасників і їх розподілена відповідальність. У ньому відображається рішення про спілкування КС з замовником, сервіси та ПВК. Мета патерна – визначення певної варіантності при взаємодії компонентів.

Структурно композиції компонентів представляється графом, описом компонентів та зв’язків (асоціацій) між компонентами КС та твердженнями про елементи і їх композиції. З точки зору теорії першого порядку, складний патерн є описом предикатів класів, станів змінних, методів та їх відношень. Клас та об’єкт створюють сорти компонентів першого класу, а також можуть використовуватися сорти bool та int. Наприклад, сигнатура складеного патерну може мати вигляд:

Add: class → bool
Remote: class → bool
GetChild: class × int → bool
Operation: class → bool
Variable: class × object → bool
Inherit: class × class → bool

Композиційний патерн. Складений патерн може використовуватися для представлення частини або усієї ієрархії компонентів, які всі повинні бути уніфіковані. За допомогою патерну виконується доступ до елементів агрегатного об'єкту композиції.

У загальному випадку існують чотири можливі класи формальних композицій, елементами яких є компоненти і каркаси, взаємодіючі між собою.

Композиція *компонент-компонент* забезпечує безпосередню взаємодію компонентів через інтерфейс на рівні застосування.

Композиція *каркас-компонент* забезпечує взаємодію каркаса з компонентами, при якому каркас керує ресурсами компонентів і їхніх інтерфейсів. Така взаємодія виконується на системному рівні.

Композиція *компонент-каркас* забезпечує взаємодію компонента з каркасом типу "чорної скриньки", у видимій частині якого знаходяться специфікації для його розгорнення і виконання визначеної сервісної функції. Така взаємодія виконується на сервісному рівні.

Композиція *каркас-каркас* забезпечує взаємодію між каркасами, кожний з яких розгортається в гетерогенному середовищі і дозволяє компонентам каркаса взаємодіяти через їхній інтерфейс. Дана взаємодія виконується на мережному рівні.

Верифікація компонентів

Перевірка правильності компонентів виконується для примітивних і складних компонентів. Кожному з них відповідає деяка процедура верифікації, що створюються згідно опису системи в мові специфікації xUML і задовольняє наступним вимогам:

– КС проектується як модель, властивості компонента задаються в xUML;
– модель xUML і властивості компонентів транслюються до властивостей середовища;
– якщо властивості не має, генерується траса помилки і повідомлення про це.

Цей підхід дозволяє виконати перевірку компонентів на моделі КС невеликого розміру. Для перевірки проектування великих за розміром КС, модель перевірки властивостей декомпозується на модулі, перевірка властивостей яких проводиться локально для кожного модуля окремо.

Властивості підкомпонентів та ПВК КС перевіряються на підкомпонентах окремо. При цьому вручну проводиться декомпозиція властивостей, але це допускається при перевірці компонентів у великомасштабних КС.

Властивості компонента визначаються з урахуванням умов середовища компонента автоматичним чи ручним шляхом. Генеровані умови – це набір простих і інтуїтивних припущень, сформульованих в інтерфейсі компонента, для перевірки складних компонентів композиції. Кон'юнкція цих припущень є більш сильна, чим автоматично генеровані припущення.

Верифікація примітивів. Примітив має невеликий розмір, зручний для перевірки на моделі. З огляду того, що примітив C = (E, I, V, P) і властивість (p, A(p)) визначені на множинах I і V, верифікуються такими наступними шагами:

1) створення моделі обчислень у середовищі виконання КС, підготовка вхідних типів повідомлень і моделі станів для отримання вихідних повідомлень з множини I;

2) формування системи для виконання процесів кодового представлення компонента E і її транслювання;

3) транслювання елементів середовища для отримання властивості A(p) і зіставлення її з моделлю КС у відповідності зі станом середовища;

4) транслювання умов p з урахуванням властивостей.

Верифікація складних компонентів. Для перевірки властивості (p, A(p)) складеного компонента C = = (E, I, V, P), наприклад, із двох компонентів $C_0 = (E_0, I_0, V_0, P_0)$ і $C_1 = (E_1, I_1, V_1, P_1)$ перевіряються властивості, що були перевірені на окремих підкомпонентах C_0 і C_1.

Створюється абстракція компонента, яка уточнюється поняттям властивості і визначається умовою середовища, а потім перевіряється КС згідно з цими умовами. Коли компонент багаторазово використовується в композиції складного компонента, властивість містить умови середовища, отримані при її перевірці її компонентів. Тобто властивість (p, A(p)), компонента C_i, де i Є {0, 1} і $(p_i, A(p_i))$ Є P_i, допускає можливість функціонувати в композиції C_0 і C_i. Коли властивість $(p, A(p_i))$, компонента C_i, і Є {0, 1}, припустима в композиції C_0 і C_i , передбачається, що A(p) містяться у середовища їхньої композиції. Для кожного припущення q, q Є $A(p_i)$, функція cone $(A(P) \cup P_{i-1}, q)$ спочатку ідентифікує множину P' властивостей.

Якщо q є результатом cone $(A(P) \cup P_{i-1}, q)$, то функція виконується з наступним припущенням для $A(P_i)$, інакше функція повертає false. Значення може бути отримано будь-якою відповідністю q до елемента cone $(A(P) \cup P', q)$ чи моделлю, що перевіряє q на елемент cone $(A(P) \cup P', q)$. Якщо властивість $(p, A(p_i))$ компонента

С не допускається в композиції C_0 і C_i, то це не означає, що р не тримається в С і при композиції (p_i, $A(p_i)$) може стати що допускається, коли всі припущення в $A(p_i)$ допускаються і зажадають перевірки додаткових властивостей C_0 і C_i .

Абстракція складного компонента С перевіряється такими шляхами:

1) реалізація інтерфейсів вихідного повідомлення C_0 (або C_i відповідно) у контексті С шляхом заміни параметрів;

2) створення stub процесів – CP_0 , CP_1 , що відповідають C_0 (чи C_i), перемінні яких мають ті ж самі назви і домени, що і перемінні в множині V_o (або V_i), типи вхідних повідомлень яких ті ж самі, що і типи вхідних повідомлень, що відповідають середовищу С;

3) виконання функції за кожною властивістю P_0 і P_i у середовищі системи, якщо функція повертає true;

4) включення припущення A(p) у середовищі системи;

5) виконання функції cone на процесах середовища системи так, щоб виключалися властивості й припущення, не зв'язані зі значенням р.

Серед властивостей підкомпонентів можуть існувати залежності, що не відносяться до властивостей цих підкомпонентів. Якщо перевірене, що властивість р міститься в компоненті C_0, і відповідає властивості компонента C_1 і навпаки, то робимо висновок, що ці дві властивості містяться в С. Якщо не можна показати, що С не виконується, властивість р є помилковою. Істинність таких залежностей перевіряється за допомогою композиційних правил.

Перевірка абстракції компонента. Замість того, щоб перевіряти властивість р на специфікації компонента С в середовищі, перевіряється воно на абстрактної структурі компонентів КС. Складність моделі, що перевіряє р на обраної абстракції є набагато нижче чим складність прямої перевірки властивості р на специфікації компонента С.

Оцінка якості перевірених компонентів. Після повної перевірки окремих компонентів на правильність функціонування, як правило, виконується оцінка якості компонентів і КС у цілому [18]. Для проведення процесів оцінювання склалося багато методів і підходів [17–27], за допомогою яких розраховується аналітична оцінка деяких показників як експертне оцінювання останніх. Особливим процесом є тестування і збір даних про стан виконання КС і оцінювання надійності КС.

Висновок

Розглянуті питання верифікації простих і складних компонентів на основі опису їх за умов програмування та інтерфейсів у модифікованій мові xUML. Запропоновано модель перевірки компонентів, загальна модель компонентів, що будується з перевірених компонентів. Розглянуті основні підходи до реалізації.

1. *Редько В.Н.* Композиционная структура программологии // Кибенетика и системный анализ. – 1998. – № 4. – С. 47–66.
2. *Doug J., Paulo P Alencar, Covand D.* Correct Composition of design Components. Procceding of Annual Internations Computer Software and Application Conferens (COMPSAC). – 1990.–Oct..– P. 101–108.
3. *Рожнов О.М.* Аналіз методів специфікації та верифікації компонентів розподілених застосувань // Проблемы программирования..– 1998.– № 4. – С. 93–101.
4. *ANSI* / IEEE Std. 10122-1986. Standard for Software Verification and Validation Plans // IEEE. – New York.– 1986.– 61p.
5. *Рожнов А.М.* Модели и средства взаимосвязи компонентов в распределенных средах // Проблемы программирования.– 2000. – № 1–2. – С. 219-225.
6. *Myers G.J.* The art Software Testing.–New York: Wiley.–1979.– 413 P.
7. *Perry D.E. and Kaiser C.E.* Adequate testing and object-oriented programming // J. of Object-Oriented Programming, January.– February. – 1990. – PP. 13–19.
8. *Wang Y., King J., Kourt J., Ross M., Staples S.* On testable object-oriented programming. Software Engineering Notes, Vol. 22, № 4.–1997.– PP. 84–90.
9. *Лаврищева Е.М., Коротун Т.М.* Построение процесса тестирования программных систем // Проблемы программирования.–2002. – № 1–2. – С. 272–281.
10. *Эммерих В*. Конструирование распределенных объектов. Методы и средства программирования интероперабельных объектов в архитектурах OMG/CORBA, Microsoft /COM и Java/ RMI. – М.: Мир, 2002. – 510 с.
11. *Xie Fei, J.C. Browne.* Verified Systems by Composition from Verified Components. – ACM.–2003.– № 1.– P. 13–47.
12. *Haag S., Raja H.K., Sekade L.L.* Quality Function Deployment. Usage in Software Development // Comm. of ACM.– Oct., 2002. – P.35–40.
13. *Crnkovic I., Hnich B., Jonsson T., Kiziltan Z.* Specification, Implementation and Deployment of Components // Comm. of the ACM. – October 2002. – P. 35–40. – 1998. –9, № 1. – P. 71–76.
14. *Batory D., O'Malley S.* The Design and Implementation of Hierarchical Software Systems with Reusable Components // ACM Transactions on Software Engineering and Methodology. – 1992. – № 4, vol. 1, October . – PP. 355–398.
15. *Weide B., Ogden W., Zweben S.* Reusable Software Components // Advances in Computers, Academic Press, 1991. – Vol. 33. – PP. 1–65.
16. *Rada R., Moore J.* Standardizing Reuse // Comm. of the ACM. – 1997. – 40, N 3. – P. 19–23.
17. *Андон Ф.И., Коваль Г.И., Коротун Т.М., В.Ю.Суслов.* Основы инженерии качества программных систем. – Киев: "Академпериодика", 2002.– 502 с.
18. *Лаврищева Е.М., Рожнов А.М.* Концепция аналитической оценки характеристик качества программных компонентов // Проблеми програмування. – Киев: 2004. – № 1–2. – С. 180–187.
19. *Липаев В.В.* Методы обеспечения качества крупномасштабных программных систем. – М.: СИНТЕГ, 2003.–510 с.
20. *Pfleeger S.L.* Software Engineering. Theory and practice.–Printice Hall. Upper Saddenle River, New Jersey 07458. –1998.–576 p.
21. *Цимбал А.А, Аншина М.Л.* Технологии создания распределенных систем. Для профессионалов. – СПб.: Питер, 2003. – 576 с.
22. *Андон Ф.И., Лаврищева Е.М.* Методы инженерии распределенных компьютерных систем.– Киев: Изд. Наук. думка, 1997.–228 с.
23. .*Meyer B.* The role of Object–Oriented Metrics // Computer.– 1998. –№ 11.– P. 23–125.
24. *IEEE* Software. Measurement.– March/April, 1997.
25. *Липаев. В.В.* Выбор и оценивание характеристик качества программных систем. Методы и стандарты. – М.: СИНТЕГ, 2001. – 224 с.
26. *Кулаков А.Ф.* Оценка качества программ ЭВМ.– Киев: Техніка, 1984. – 167 с.
27. *Сигел Дж.* CORBA 3. – М. Малип, 2002. – 412 с.