УДК 681.3.06

МЕТОДИ АНАЛІЗУ ПРОГРАМНИХ АРХІТЕКТУР, ПРЕДСТАВЛЕНИХ НЕЧІТКИМИ ГРАФОВИМИ МОДЕЛЯМИ

І.М.Парасюк, С.В.Єршов

Інститут кібернетики ім.В.М.Глушкова НАН України, 03680, Київ-187, просп. Акад. Глушкова, 40,
тел.526 6422, e-mail: ivpar1@i.com.ua

Запропоновано засоби для представлення складних нечітко визначених підсистем та модулів на рівні об'єктно-орієнтованої програмної архітектури. Розглянуто способи введення нечіткості в діаграмах класів UML, що є основою аналізу архітектури. Описані трансформації, які підвищують ефективність представлення графових моделей архітектури з метою поліпшення розуміння програмної структури, діагностування, аналізу архітектури та її реструктуризації. Розроблено узагальнений підхід до трансформації нечіткого графа моделі архітектури, який ґрунтується на понятті нечіткої семантичної збіжності шаблонів правил перетворення.

Facilities for presentation of complex fuzzy subsystems and modules at the level of the object-oriented program architecture are offered. Methods of uncertainty introduction are considered in UML classes diagrams, which are the basis of architecture analysis. Transformations, which promote efficiency of presentation of graph-based models of architecture with the purpose of improvement of program structure understanding, diagnostics, analysis of architecture and its restructuring, are described. Generalized approach is developed to transformation of fuzzy graph of architecture model, which is based on notion of fuzzy semantic matching of transformation rules patterns.

Вступ

Сучасна точка зору, відображена у модель-орієнтованій архітектурі MDA (Model-Driven Architecture) [1] і робить значний акцент на використанні аналізу та моделювання програмних архітектур у процесі розробки та подальшої еволюції програмного забезпечення.

Підхід MDA ґрунтується на поняттях платформно-незалежної моделі (Platform Independent Model, PIM) та платформно-залежної моделі (Platform Specific Model, PSM). PIM – абстрактна модель архітектури програмного забезпечення, яка ігнорує особливості платформи реалізації (певні особливості реалізації в окремому середовищі). PSM, з другого боку, – інша модель, яка включає певні деталі реалізації та середовища. Одночасно декілька моделей PSM можуть бути автоматично породжені від одної моделі програмної архітектури рівня PIM. Проте цей процес породження надзвичайно залежить від ступеня невизначеності архітектурних моделей: різним архітектурам, можливо, потрібні спеціальні методи для здійснення породження цільової моделі залежно від ступеня нечіткості початкової архітектури і невизначеності у моделях самого середовища.

В процесі розробки моделей програмного забезпечення системи можуть бути сформульовані у нечіткому вимірі вимоги до системи або оцінок ефективності її майбутнього функціонування. При виборі або проектуванні базової архітектури програмної системи існує певна кількість альтернативних рішень щодо її структури. У традиційних методах аналізу програмних архітектур, таких, як SAAM [2,3], вибір адекватної архітектури здійснюється серед можливих альтернатив на основі програмних метрик або числових показників. Таким чином, нечіткі показники повністю ігноруються, а там, де вони природно виникають, зводяться до чітких значень, що призводить до втрати важливої інформації щодо намірів учасників розробки. З урахуванням еволюції програмного забезпечення зазначений факт вимагає частих "модифікацій" PIM-моделей, означає втрати її гнучкості, а також вимагає додаткових витрат при генерації та тестуванні реалізацій PSM-моделей.

Програмна архітектура – це виразник найважливіших знань про програмну систему, що не залежать від способів фіксації, якими є певні мови і нотації. Разом з тим загальноприйнятий спосіб архітектурної нотації, який сьогодні існує, – це Уніфікована мова моделювання (Unified Modeling Language, UML) [4-6].

Архітектурні структури підрозділяються на три загальні групи [2], в кожну з яких включаються елементи певного характеру. Елементами модульних структур є модулі – елементи реалізації. У разі структури компонент елементами є компоненти (основні одиниці обчислень) і з'єднувачі (інструменти взаємодії між компонентами) періоду обчислень. Структури розподілу демонструють зв'язок між програмними елементами і елементами зовнішнього середовища, в якому програмне забезпечення виконується.

Набір конструкцій, призначений в UML для відображення модульної структури, вельми обширний [2]. Конструкція класу в UML відповідає об'єктно-орієнтованій спеціалізації модуля. У випадках, коли значну роль грає угрупування функціональності (наприклад, при відображенні рівнів і класів), зручніше звертатися до пакетів. На рис.1 наводиться приклад позначення засобами UML відношень, які властиві для модульних представлень.

Несумісність, неточність, невизначеність, невпевненість і неоднозначність – це п'ять основних видів недосконалої інформації щодо програмної архітектури.

Несумісність є свого роду семантичним конфліктом, коли один аспект системи несумісно представлений більше одного разу в моделі системи або окремих різних моделях. Інтуїтивно, неточність і невизначеність

доречні для вмісту значення атрибуту, і це означає, що вибір має робитися із заданого ряду значень, але не відомо точно, яке з них вибрати в даний час.

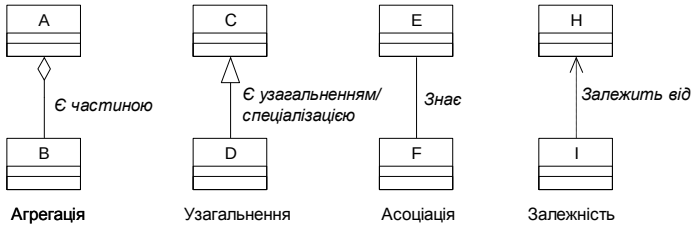


Рис. 1. Основні відношення між модулями (класами або пакетами) в моделях модульної структури

Невпевненість пов'язана зі ступенем істинності значення класів, об’єктів та атрибутів, і це означає, що хтось може розділити деяку, але не всю, нашу довіру до даного значення або групи значень. Неоднозначність означає, що деяким елементам моделі не вистачає повної семантики, що приводить до окремих можливих інтерпретацій.

Невизначеність і невпевненість загалом моделюються нечіткими множинами і теорією можливостей. Багато існуючих підходів мають справу з неточністю і невпевненістю, представленими засобами теорії нечітких множин [7].

Дана стаття є подальшим розвитком ідей та досліджень авторів у напрямку становлення трансформаційного підходу на основі модель-орієнтованих архітектур у нечіткому поданні [8-10].

Зокрема, розглядається задача формалізації об'єктно-орієнтованої модульної структури, що є основою програмної архітектури та враховує нечіткість визначення як самих модулів, так і можливих відношень між ними. Для цього розроблено нечітке подання класів та класів діаграм UML, а також змістовних відношень між ними (агрегація, узагальнення, асоціація та залежність). Зазначений підхід узагальнено до системи правил трансформацій нечіткого графа моделі архітектури, які ґрунтуються на понятті нечіткої збіжності нечіткого графа, що описує модель архітектури, та нечіткого правила трансформації. Крім того, прокласифіковано основні види трансформацій архітектур, представлених нечіткими графами, та наводяться приклади їх використання у контексті покращення модульної структури програмної системи.

Зважаючи на те, що основу моделювання програмної архітектури складають об'єктно-орієнтовані модульні структури, пропонуються інші структури (компонент і розподіл) зводити до специфікації діаграм класів, оскільки всі типи елементів, які в них використовуються, – класифікатори UML [4]. Зазначена специфікація може здійснюватися засобами метамоделювання UML, розширених механізмами для представлення нечітких об’єктно-орієнтованих структур.

Формалізація нечітких об’єктно-орієнтованих структур

UML є набором об'єктно-орієнтованих позначень для моделювання і є стандартом Об'єктної Групи Управління Даними (ODMG). Вона інтенсивно застосовується до моделювання об’єктно-орієнтованих структур, що описують програмну архітектуру. Проте інформація щодо елементів та їх відношень в реальних системах часто невизначена або неоднозначна. Введемо в класи UML та у відповідні графічні представлення різні рівні нечіткості та розглянемо запропонований спосіб моделювання нечітко визначеної архітектури діаграмами класів UML.

UML забезпечує колекцію моделей, щоб охопити різні аспекти системи програмного забезпечення. З точку зору моделювання програмної архітектури найдоречніша модель – це модель класів. Будівельні блоки в цій моделі– це класи і відношення між ними.

Класи є дескриптором для набору об'єктів з подібною структурою, поведінкою і відношеннями, відображають поняття в межах модельованої системи, містять структуру даних, поведінку та взаємовідношення з іншими елементами.

Іншим важливим структурним компонентом в діаграмі класів UML є представлення взаємовідношення між класами або екземплярами класів. UML підтримує різні відношення: агрегат і композиція, узагальнення, асоціація та залежність [4-6]. Агрегат охоплює відношення ціле-частина між агрегатом, класом, який зображає ціле, і складовою частиною. Узагальнення використовується, щоб визначити відношення між класами для побудови таксономії класів: один клас – це більш загальний опис набору інших класів. Асоціація – це відношення, які описують зв'язки між екземплярами класів. Для кожного класу, що бере участь в асоціації, може бути призначена роль, що робить асоціацію направленим зв'язком. Залежність указує семантичне відношення між двома класами. Це указує ситуацію, в якій зміна в цільовому класу може вимагати зміни у початковому класу в залежності.

Нечіткі об'єкти та класи. Об'єкти є нечіткими через відсутність інформації щодо вимог стосовно їх функціонування. Формально об'єкти, які мають як мінімум один атрибут, чиє значення – нечітка множина, – це нечіткі об'єкти.

Теоретично клас може бути розглянутий з двох різних точок зору:

а) клас-екстенсіонал, де клас визначений списком об'єктів-екземплярів, і

б) клас-інтенсіонал, де клас визначений набором атрибутів і їх допустимих значень.

Крім того, підклас, визначений від суперкласу за допомогою механізму спадкоємства, може розглядатися як особливий варіант вищезазначеного випадку б).

Таким чином, клас є нечітким через наступні окремі причини. По-перше, існують певні об'єкти, які є нечіткими та мають подібні властивості. Клас, визначений цими об'єктами, може бути нечітким. Ці об'єкти належать класу зі ступенем належності [0, 1]. По-друге, коли клас є інтенсіонально визначеним, область атрибуту може бути нечіткою і нечіткий клас сформований. По-третє, підклас, що виробляється нечітким класом за допомогою спеціалізації, та суперклас, що виробляється певними класами (серед яких є як мінімум один нечіткий клас) за допомогою узагальнення, також є нечітким.

Нечіткі відношення об'єкт-клас. Існують наступні чотири ситуації для відношень об'єкт-клас:

а) чіткий клас і чіткий об'єкт: об'єкт впевнено належить або не належить до класу;

б) чіткий клас і нечіткий об'єкт: об'єкт може бути пов'язаний з класом зі спеціальним ступенем впевненості в діапазоні [0, 1];

в) нечіткий клас і чіткий об'єкт: об'єкт може належати класу зі ступенем належності в діапазоні [0, 1];

г) нечіткий клас і нечіткий об'єкт: об'єкт належить класу зі ступенем належності в діапазоні [0, 1].

Вищезазначені в б) – д) відношення об'єкт-клас називають нечіткими. Фактично ситуація а) може розглядатися як особливий випадок нечітких відношень об'єкт-клас, де суб'єктивна ступінь належності об'єкта до класу є одиницею.

Для того щоб обчислити суб'єктивну ступінь належності об'єкта до класу в нечіткому відношенні об'єкт-клас, необхідно оцінити ступені того, що області атрибутів класу включають значення атрибутів об'єкта. Атрибути грають іншу роль у визначенні та ідентифікації класу, тому вага *w* призначається проектувальником для кожного атрибута класу згідно його важливості. Тоді ступінь належності об'єкта до класу в нечіткому відношенні об'єкт-клас має бути обчислений з використанням ступеня включення значень об'єктів щодо областей класу і ваги атрибутів.

Позначимо *U* універсальну множину. Нечітке значення на *U* може характеризуватися нечіткою множиною *F* в *U*. Функція належності $\mu_F : U \to [0,1]$ визначена для нечіткої множини *F*, де $\mu_F(u)$ для кожного *u* означає ступінь членства *u* в нечіткій множині *F*. Тому нечітка множина *F* описується як вказано нижче [7]:

$$F = \{\mu(u_1)/u_1, \mu(u_2)/u_2, ..., \mu(u_n)/u_n\}, \tag{1}$$

де пара $\mu(u_i)/u_i$ представляє значення u_i і ступінь його членства $\mu(u_i)$. Функція належності $\mu_F(u)$ може інтерпретуватися як міра можливості, що значення змінної *X* – це *u*.

Нехай *C* – клас з атрибутами $\{A_1, A_2, ..., A_n\}$; *o* – об'єкт визначений на множині атрибутів $\{A_1, A_2, ..., A_n\}$; $o(A_i)$ означає значення атрибута об'єкта *o* на $A_i, 1 \le i \le n$. В класі *C* кожний атрибут A_i позначають $ID(dom(A_i), o(A_i))$. Дослідимо оцінку $ID(dom(A_i), o(A_i))$.

Припустимо $fdom(A_i) = \{f_1, f_2, ..., f_m\}$, де $f_j, 1 \le j \le m$, – нечітке значення, і $cdom(A_i) = \{c_1, c_2, ..., c_k\}$, де $c_l, 1 \le l \le k$ – чітке значення. Тоді

$$\begin{aligned} ID(dom(A_i), o(A_i)) &= \max\big(ID(cdom(A_i), o(A_i)), ID(fdom(A_i), o(A_i))\big) = \\ &= \max(\mathrm{SID}(\{1{,}0/c_1,\ 1{,}0/c_2,\ \ldots,\ 1{,}0/c_k\}, o(A_i)), \\ &\quad \max_j(SID(f_j, o(A_i)))) . \end{aligned} \tag{2}$$

Тепер визначимо формулу для обчислення ступеня належності об'єкта *o* до класу *C*, як вказано нижче, де $w(A_i(C))$ означає вагу атрибута A_i в класі *C*:

$$\mu_C(o) = \frac{\sum_{i=1}^{n} ID(dom(A_i), o(A_i)) \cdot w(A_i(C))}{\sum_{i=1}^{n} w(A_i(C))} . \tag{3}$$

Визначимо три рівні нечіткості.

1. Нечіткість, до якої міри клас належить моделі даних також, як нечіткість вмісту (в термінах атрибутів) класу.

2. Нечіткість, пов'язана з визначенням того, чи певні об'єкти є екземплярами класу.

3. Нечіткість на значеннях атрибутів екземплярів класу; атрибут в класі визначає область значення, і, коли ця область нечітка підмножина або набір нечітких підмножин, з'являється нечіткість значення атрибута.

Для того щоб моделювати перший рівень нечіткості, тобто атрибут або клас із ступенем належності, атрибут або ім'я класу мають бути завершені парою слів WITH *mem* DEGREE, де $0 \le mem \le 1$. Для того щоб промоделювати третій рівень нечіткості, ключове слово FUZZY розміщене перед атрибутом. При другому рівні

нечіткості додатковий атрибут (μ) вводиться в клас для представлення ступеня належності екземпляра до класу з областю значень атрибута [0, 1]. Для диференціювання класу з другим рівнем нечіткості можна використовувати прямокутник з пунктирним контуром, щоб позначати такий клас.

Нечітке узагальнення. Новий клас, що називається підклас, – породжувана форма іншого класу, що називається суперклас. Розглянемо нечітке узагальнення, яке ґрунтується на точці зору на клас як на ектенсіонал.

Можна виробити нечітке відношення підклас-суперклас, використовуючи ступінь включення об'єктів до класу. Оскільки підклас – спеціалізація суперкласу, будь-який об'єкт, що належить до підкласу, має належати суперкласу. Зазначена характеристика може бути використана, щоб визначити, чи мають два класи взаємовідношення підклас-суперклас. Підклас, утворений від нечіткого суперкласу, повинен бути нечітким, відношення підклас-суперклас є нечітким так само. Іншими словами, клас – підклас іншого класу зі ступенем належності [0, 1] в певний момент часу. Відповідно є наступний метод для визначення відношення підклас-суперклас.

Якщо класи мають тільки другий рівень нечіткості, тоді для нечіткого об'єкта, якщо ступінь належності з яким він є екземпляром підкласу – менша або дорівнює ступеню належності суперкласу, то він належить суперкласу. Ступінь належності, що він є екземпляром підкласу більший або дорівнює до заданого порогу.

Формально позначимо *A* і *B* – нечіткі класи і β – заданий поріг. Тоді *B* – підклас *A*, якщо

$$(\forall e)(\beta \le \mu_B(e) \le \mu_A(e)). \tag{4}$$

Ступінь належності, з яким *B* є підкласом *A*, має бути $\min_{\mu_B(e)\ge\beta}(\mu_B(e))$. Тут *e* – об'єкт-екземпляр класів *B* в універсальній множині, $\mu_A(e)$ і $\mu_B(e)$ – ступені належності *e* до *A* і *B* відповідно.

Розглянемо ситуацію, у якій *A* або *B* – класи з певним ступенем належності *deg(A)* та *deg(B)* відповідно, а саме з першим рівнем нечіткості:

```
A WITH deg(A) DEGREE
B WITH deg(B) DEGREE
```

Тоді *B* – підклас *A*, якщо

$$(\forall e)(\beta \le \mu_B(e) \le \mu_A(e)) \wedge (\beta \le \deg(B) \le \deg(A)). \tag{5}$$

Ступені належності *A* і *B* мають бути більшими або дорівнювати заданому порогу, а ступінь належності класу *A* має бути більшою або дорівнювати ступеню належності *B*.

Розглянемо нечіткий суперклас *A* і його нечіткі підкласи $B_1, B_2, ..., B_n$ із ступенями належності $\mu_A, \mu_{B1}, \mu_{B2}, ..., \mu_{Bn}$ відповідно, які можуть мати ступені належності *deg(A), deg(B), deg(B2)* . . . та *deg(Bn)* відповідно. Тоді наступне взаємовідношення є вірним:

$$(\forall e)(\max(\mu_{B1}(e), \mu_{B2}(e), ..., \mu_{Bn}(e)) \le \mu_A(e)) \wedge (\max(\deg(\mathrm{B1}), \deg(\mathrm{B2}), \ldots, \deg(\mathrm{Bn})) \le \deg(\mathrm{A})). \tag{6}$$

Якщо прийняти за основу інтенсіональну точку зору на клас, можна використовувати ступінь включення класу щодо іншого класу для визначення відношень між нечітким підкласом і суперкласом. Припустимо, що класи *A* та *B* можуть мати тільки другий рівень нечіткості. Якщо *A* і *B* – нечіткі класи, то ступінь того, що *B* є підкласом *A*, позначимо $\mu(A, B)$. Для заданого порогу β *B* – підклас *A*, якщо $\mu(A, B) \ge \beta$.

Розглянемо ситуацію, у якій *A* або *B* – класи з певним ступенем належності, а саме з першим рівнем нечіткості. Тоді *B* – підклас *A*, якщо

$$(\mu(A, B) \ge \beta) \wedge (\beta \le \deg(\mathrm{B}) \le \deg(\mathrm{A})). \tag{7}$$

Ступінь включення нечіткого підкласу до нечіткого суперкласу може бути визначена з урахуванням ступеня включення областей атрибута підкласу до областей атрибутів суперкласу, а також ваги певних атрибутів.

Нечітка агрегація. Агрегація охоплює відношення ціле-частина між агрегатом і його складовою частиною.

При екстенсіональній точці зору на клас кожен екземпляр агрегата може проектуватися на множину екземплярів його складових частин. Нехай *A* – агрегат складових частин $B_1, B_2, ...$ та B_n. Для $e \in A$ проекцію *e* на B_i позначимо $e \downarrow B_i$. Тоді $(e \downarrow B_1) \in B_1, (e \downarrow B_2) \in B_2, \ldots, (e \downarrow B_n) \in B_n$.

Клас, який агрегований із нечітких складових частин, має бути нечітким, тобто агрегат є нечітким так само. У цей момент клас – агрегат складових частин з ступенем належності у діапазоні [0, 1].

Припустимо, що класи можуть мати тільки другий рівень нечіткості. По-перше, для будь-якого нечіткого об'єкта, якщо ступінь його належності до агрегату є меншою або дорівнює ступеню належності, з яким його проекція на кожну складову частину належить до відповідної складової частини. По-друге, ступінь його

належності до агрегату є більшою або дорівнює заданому порогу. Тоді агрегат – агрегація складових частин зі ступенем належності, який є мінімумом серед ступеней належності, з якими проекції цих об'єктів на ці складові частини належать до відповідних складових частин.

Припустимо що A – нечіткий агрегат нечітких наборів класів $B_1, B_2, ...$ і B_n зі ступенями належності екземплярів $\mu_A, \mu_{B1}, \mu_{B2}, ..., \mu_{Bn}$ відповідно, β – заданий поріг. Тоді

$$(\forall e)(e \in A) \wedge (\beta \le \mu_A(e) \le \min(\mu_{B1}(e \downarrow B_1), \mu_{B2}(e \downarrow B_2), ..., \mu_{Bn}(e \downarrow B_n))). \quad (8)$$

Ступінь належності того, що A є агрегатом набору класів $B_1, B_2, ...$ і B_n має бути $\min_{\mu_{Bi}(e \downarrow B_i) \ge \beta}(\mu_{Bi}(e \downarrow B_i)), 1 \le i \le n$. Тут e – об'єкт-екземпляр A.

Розглянемо перший рівень нечіткості в класах $A, B_1, B_2, ...$ і B_n, а саме коли вони є невизначеними класами з певним ступенем належності:

```
 A WITH deg(A) DEGREE,
B1 WITH deg(B1) DEGREE,
B2 WITH deg(B2) DEGREE,
     . . . . . .
Bn WITH deg(Bn) DEGREE.
```

Тоді A – агрегат $B_1, B_2, ...$ і B_n, якщо

$$(\forall e)(e \in A) \wedge (\beta \le \mu_A(e) \le \min(\mu_{B1}(e \downarrow B_1), \mu_{B2}(e \downarrow B_2), ..., \mu_{Bn}(e \downarrow B_n))) \wedge \\ \wedge (\deg(A) \le \min(\deg(B1), \deg(B2), \ldots, \deg(Bn))). \quad (9)$$

Опишемо нечітку агрегацію при застосуванні інтенсіональної точки зору на клас. Припустимо що A – нечіткий агрегат набору нечітких класів $B_1, B_2, ...$ і B_n, а β – заданий поріг. Також позначимо проекцію A на B_i як $A \downarrow B_i$. Припустимо, що класи $A, B_1, B_2, ...$ і B_n можуть мати тільки другий рівень нечіткості. Тоді для нечіткої агрегації має виконуватися наступне:

$$\min(\mu(B_1, A \downarrow B_1), \mu(B_2, A \downarrow B_2), ..., \mu(B_n, A \downarrow B_n)) \ge \beta. \quad (10)$$

Тут $\mu(B_i, A \downarrow B_i), 1 \le i \le n$ – ступінь, з яким B_i семантично включає $A \downarrow B_n$. Ступінь належності, з якою A – агрегація $B_1, B_2, ...$ та B_n визначається як $\min(\mu(B_1, A \downarrow B_1), \mu(B_2, A \downarrow B_2), ..., \mu(B_n, A \downarrow B_n))$.

Розглянемо перший рівень нечіткості в класах $A, B_1, B_2, ...$ і B_n, а саме, коли вони – нечіткі класи із певними ступенями належності. Тоді A – агрегат $B_1, B_2, ...,$ і B_n, якщо

$$\min(\mu(B_1, A \downarrow B_1), \mu(B_2, A \downarrow B_2), ..., \mu(B_n, A \downarrow B_n)) \ge \beta \wedge \\ \wedge \deg(A) \le \min(\deg(B1), \deg(B2), \ldots, \deg(Bn)). \quad (11)$$

Нечітка асоціація. Два рівні нечіткості можуть бути ідентифіковані для відношенні асоціації. Перший рівень означає, що відношення асоціації нечітко існує для двох зв'язаних класів, а саме це відношення асоціації відбувається зі ступенем можливості. Також, можливо, що є невідомим, чи два екземпляри класу, які відповідно належать асоційованим класам мають задане відношення асоціації, хоча це відношення асоціації має відбуватися для двох класів. Це є другий рівень нечіткості у відношенні асоціації і він викликаний тим, що екземпляр належить заданому класу з певним ступенем належності. Можливо, що два вищезазначені рівні нечіткості відбуваються у відношенні асоціації одночасно.

Розмістимо вираз `WITH mem DEGREE`, де $0 \le mem \le 1$, після імені ролі відношення асоціації, щоб представити перший рівень нечіткості відношення асоціації. Як було зазначено вище, у класах можуть відбуватися три рівні нечіткості.

Класи з другим рівнем нечіткості, загалом, приводять до другого рівня нечіткості в асоціації, якщо ця асоціація певним чином існує. Нехай A і B – два класи з другим рівнем нечіткості. Тоді екземпляр класу e має ступінь належності $\mu_A(e)$, а екземпляр f класу B – $\mu_B(f)$. Припустимо, що відношення асоціації між A та B, позначене $ass(A, B)$, – без першого рівня нечіткості. Ясно, що відношення асоціації між e і f, що позначене $ass(e, f)$, - з другим рівнем нечіткості, ступінь належності якого може бути розрахована наступним чином:

$$\mu(ass(e, f)) = \min(\mu_A(e), \mu_B(f)). \quad (12)$$

Класи з першим рівнем нечіткості приводять до першого рівня нечіткості асоціації, якщо ця асоціація не вказана явно. Позначимо A і B два класи тільки з першим рівнем нечіткості, тобто `A WITH deg(A) DEGREE` та `B WITH deg(B) DEGREE` відповідно. Тоді відношення асоціації між A і B, що позначене $ass(A, B)$, – з першим рівнем нечіткості, а саме `ass(A,B) WITH deg(ass) DEGREE`. Для екземпляра e класу A і екземпляра f класу B, де $\mu_A(e) = 1$ і $\mu_B(f) = 1$, виконується:

$$\mu(ass(e, f)) = \deg(\text{ass}) = \min(\deg(\text{A}), \deg(\text{B})). \quad (13)$$

Зосередимося на ситуації, в якій класи першого та другого рівня нечіткості. Припустимо A і B – два класи з першим рівнем нечіткості, позначені `A WITH deg(A) DEGREE` та `B WITH deg(B) DEGREE` відповідно. Нехай $ass(A, B)$ – відношення асоціації першого рівня нечіткості між A та B, яке явно позначено `WITH deg(ass) DEGREE`. Також нехай екземпляр e класу A – зі ступенем належності $\mu_A(e)$ та екземпляр f класу B – зі ступенем належності $\mu_B(f)$. Тоді маємо:

$$\mu(ass(e, f)) = \min(\mu_A(e), \mu_B(f), \deg(\text{A}), \deg(\text{B}), \deg(\text{ass})). \quad (14)$$

Нечітка залежність. Відношення залежності пов'язане тільки з класами і не вимагає набору екземплярів для його визначення. Тому нечіткість другого і третього рівнів в класі не зачіпає відношення залежності.

Нечітке відношення залежності – відношення залежності зі ступенем можливості. Припустимо, що початковий клас відношення нечіткий, з першим рівнем нечіткості. Цільовий клас має бути нечітким, з першим рівнем нечіткості. Ступінь можливості, що цільовий клас вирішується початковим класом, такий же, як ступінь належності початкових класів.

На рис. 2 показано простий приклад визначення нечіткого класу.

```
CLASS Модуль_коду_програми WITH DEGREE OF 1,0
INHERITS Модуль WITH DEGREE OF 1,0
ATTRIBUTES
     Номер: TYPE OF string WITH DEGREE OF 1,0
     Назва: TYPE OF string WITH DEGREE OF 1,0
     Час існування: FUZZY DOMAIN {новий, бета-версія, старий, застарілий}: TYPE OF integer
          WITH DEGREE OF 1,0
     Програмна метрика: DOMAIN [0.600, 2.100]: TYPE OF real WITH DEGREE OF 1,0
WEIGHT // вага атрибутів
     w(Номер) = 0,1
     w(Ім'я) = 0,1
     w(Вік) = 0,9
     w(Програмна метрика) = 0,2
METHODS
...
END
```

Рис.2. Визначення нечіткого класу

Таким чином, формалізація нечітких об'єктно-орієнтованих структур дозволяє ввести та аналізувати різні способи представлення нечіткості в діаграмах класів UML, що представляють основу аналізу архітектури. Однак введення загального методу для опису нечітких графових моделей та різних гнучких стратегій трансформації програмних архітектур у нечіткому поданні дозволяє спростити реалізацію єдиного інструментарію для аналізу властивостей створюваних систем.

Формалізація нечітких графів та їх трансформації

Основна функція нечіткого графа – служити у якості представлення неточно визначених залежностей. За допомогою поняття нечітких графів можуть задаватися наближені представлення функцій і відношень (рис.3). Використовуватимемо нечіткі графи для наближеного представлення функцій, щоб апроксимувати одну вихідну змінну y залежно від n вхідних змінних x_i $(1 \le i \le n)$.

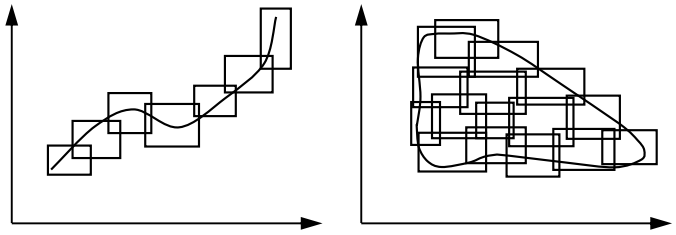


Рис. 3. Наближене представлення функцій і відношень

Будь-який нечіткий граф може бути побудований на нечітких точках типу

$$\text{якщо } x_1 = A_1 \text{ та } \ldots x_n = A_n \text{ то } y = B, \quad (15)$$

де деякі з вхідних функцій належностей A_i можуть бути постійними, тобто $\mu_{Ai} = 1$. Тому різні нечіткі точки можуть залежати тільки від індивідуального набору вхідних змінних. Спрощення цього рівняння з використанням $A = A_1 \times \ldots \times A_n$ приводить до

$$\text{якщо } \vec{x} = A \text{ то } y = B, \tag{16}$$

яке також може бути представлене як нечітке обмеження на об'єднаній змінній $(\vec{x}, y)$:

$$(\vec{x}, y) = A \times B. \tag{17}$$

Функція належності $A \times B$ задана з використанням $\wedge$ як оператора логічного множення, який звичайно визначений як мінімум

$$\mu_{A\times B}(\vec{x}, y) = \mu_A(\vec{x}) \wedge \mu_B(y) = \min\{\mu_A(\vec{x}), \mu_B(y)\} \tag{18}$$

або точніше

$$\mu_{A\times B}(\vec{x}, y) = \mu_{A1}(x_1) \wedge ... \wedge \mu_{An}(x_n) \wedge \mu_B(y) = \\ = \min\{\mu_A(x_1),..., \mu_A(x_n), \mu_B(y)\}. \tag{19}$$

Для позначення нечіткої множини $A \times B$ використовується термін "нечітка точка". Сукупність таких правил може розглядатися як суперпозиція *n* нечітких точок:

$$(\vec{x}, y) = (A_1 \times B_1 + ... + A_m \times B_m), \tag{20}$$

де + означає оператор логічного складання (звичайно визначений як максимум). Заде називає зазначену характеристику залежності нечітким графом, тому що такий набір правил може розглядатися як грубе представлення функціональної залежності f^* над $\vec{x}$. Тому нечіткий граф f^*:

$$f^* = \sum_{j=1}^{m} (A_j \times B_j). \tag{21}$$

Відзначимо, що індивідуальні правила не залежать від якого-небудь загального набору функцій приналежності над вхідними змінними. Натомість кожна нечітка точка описана через індивідуальний набір функцій приналежності A_j. Це робить нечіткі графи дуже компактною формою специфікації програмних моделей, особливо в порівнянні з підходами, які використовують "глобальну решітку", визначену за допомогою тільки одного набору функцій належності для кожної вхідної змінної.

Задача інтерполяції, тобто отримання лінгвістичного значення B для y при заданому довільному лінгвістичному значенні A для $\vec{x}$ і нечіткого графа f^*

$$\frac{\vec{x} = A, \quad f^* = \sum_{j=1}^{m} (A_j \times B_j)}{y = B}, \tag{22}$$

приводить до перетину нечіткого графа f^* з циліндровим розширенням вхідної нечіткої множини A. Ця функціональна залежність може обчислюватися таким чином:

$$\mu_{B=f^*(A)}(y) = \sup_{\vec{x}}\{\mu_{f^*}(\vec{x}, y) \wedge \mu_A(\vec{x})\} = \\ = \sup_{\vec{x}}\{(\mu_{A1\times B1}(\vec{x}, y) \vee ... \vee \mu_{Am\times Bm}(\vec{x}, y)) \wedge \mu_A(\vec{x})\}. \tag{23}$$

На рис. 4 наведена інтерполяція нечіткого значення B: “об’єкт майже належить до класу” як перетин нечіткого графа з циліндровим розширенням A, де $A = A_1 \times ... \times A_n$ – лінгвістичні значення, що характеризують атрибути класу.

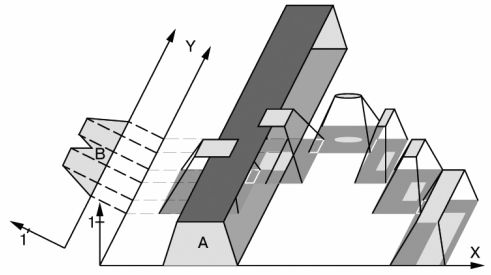


Рис. 4. Інтерполяція нечіткого значення належності об’єкта до класу

В загальному випадку покладемо, що граф є бінарним і включає будь-яку кількість нечітких точок, тобто

$A = A_1$, X=Y – універсальна множина лінгвістичних значень. Тоді $f^* =< V_F, E_F >$, де $V_F = \sum_{i=1}^{m} A_j \cup \sum_{j=1}^{m} B_j$ – множина нечітких вершин, які задані лінгвістичними перемінними; $E_F = \sum_{i=1}^{m} A_i \times B_i$ – множина нечітких відношень.

Трансформація нечітких графів включає два окремих поняття. Одне із них – це правило заміни, що встановлює відповідність між вершинами та може бути формалізоване як часткова функція на множині вершин.

Нехай f^* та g^* – нечіткі графи. Правило трансформації – це четвірка $p = \left(f^*, g^*, t_V : V_{f^*} \to V_{g^*}, t_E : E_{f^*} \to E_{g^*}\right)$, де t_V, t_E – часткові функції відображення вершин та ребер нечітких графів відповідно.

Друге поняття – це "збіжність", за її наявності можуть відбуватися відповідні правила заміни. Збіжність має виділяти певні підграфи у графах, що підлягають трансформації. Опишемо спосіб визначення семантичної збіжності на нечітких графах. Нехай $U = \{u_1, u_2, ..., u_n\}$ – універсальна множина. Нехай π_A та π_B – два значення нечітких даних на *U*, які ґрунтуються на поширенні можливості, а $\pi_A(u_i), u_i \in U$ означає можливість, що u_i відповідає дійсності. Нехай *Res* – відношення схожості на області *U*, α для $0 \le \alpha \le 1$ – поріг, відповідний *Res*. Тоді $SID(\pi_A, \pi_B)$ визначається як

$$SID(\pi_A, \pi_B) = \sum_{i=1}^{n} \min_{u_i \in D} (\pi_A(u_i), \pi_B(u_i)) \Big/ \sum_{i=1}^{n} \pi_B(u_i) . \quad (24)$$

Поняття семантичного ступеня еквівалентності значень атрибута може бути розширено до семантичного ступеня еквівалентності графів. Нехай $(f^* \downarrow) =< V_F, E_F >$ – підграф нечіткого графа, з яким порівнюється зразок правила трансформації, та $g^* =< V_G, E_G >$ – ліва частина правила трансформації (зразок). Семантичний ступінь еквівалентності нечітких графів $(f^* \downarrow)$ та g^* означає

$$SE(t_i, t_j) = \min \left\{ \begin{matrix} SID(f^* \downarrow (A_1 \times B_1), g^*(A_1 \times B_1)),\ SID(f^* \downarrow (A_2 \times B_2), g^*(A_2 \times B_2)), \\ ...,\ f^* \downarrow (A_n \times B_n), g^*(A_n \times B_n)) \end{matrix} \right\}. \quad (25)$$

Застосуванням трансформації нечіткого графа моделі архітектури *M* програмної системи (тобто моделі UML користувача) є перетворення нечіткого графа моделі заміною підграфа, що виділяє збіжність (семантичну збіжність *Res*) у лівій частині правила на підграф, визначаючого правило трансформації. Застосування включає наступні кроки:

- пошук у графі моделі *M* семантичної збіжності, встановленої у нечіткому правилі;
- перевірку можливості застосування правила трансформації (виконується, якщо певна збіжність знайдена);
- вилучення підграфу моделі *M*, який відповідає знайденій ε-збіжності (отримання контекстного нечіткого графу);
- вставка у контекстний граф нечіткого графа g^* і отримання результуючого графу моделі *M'*.

Процес трансформації складається із застосування трансформацій (мікрокроків) відповідно до семантики (макрокроків) графу потоку керування трансформаціями. Система трансформацій нечіткого графу – це трійка $FTS = \langle FG_{INIT}, P, CONF \rangle$, де FG_{INIT} – початковий нечіткий граф моделі; P – множина нечітких правил трансформації, а CONF – множина графів потоку керування.

Трансформації нечіткої програмної архітектури

В процесі еволюції програмного забезпечення часто виникає необхідність обробляти нечітку інформацію для кращого розуміння, аналізу та можливої модифікації моделей програмної архітектури [11]. Трансформації, які часто застосовуються до графових моделей архітектури, можна розділити на три види:

1. Трансформації для розуміння використовуються у разі необхідності дослідження та розуміння програмної структури. Наприклад, створюються різні рівні абстракції структури.

2. Трансформації для аналізу використовують для здобуття різних типів інформації стосовно програмної системи, такої, як наявність модулів, що вірогідно утворюють циклічну взаємодію. Зазначена інформація у подальшому використовується для визначення способів модифікації системи.

3. Трансформації для модифікації здійснюються з метою зміни у програмній структурі. Наприклад, структура змінюється з урахуванням нових вимог до системи або її коректність відновлюється у випадку, коли знайдено недоречну взаємодію між підсистемами.

Графові моделі, до яких застосовуються трансформації, представлені як нечіткі графи, що зберігають семантику та графічне представлення сутностей (пакети, класи) та відношень (агрегації, асоціації,

узагальнення). На рис. 5 наведено приклад нечіткого графа програмної архітектури, яка містить 4 підсистеми та 5 модулів, заданих нечіткими класами та пакетами відповідно.

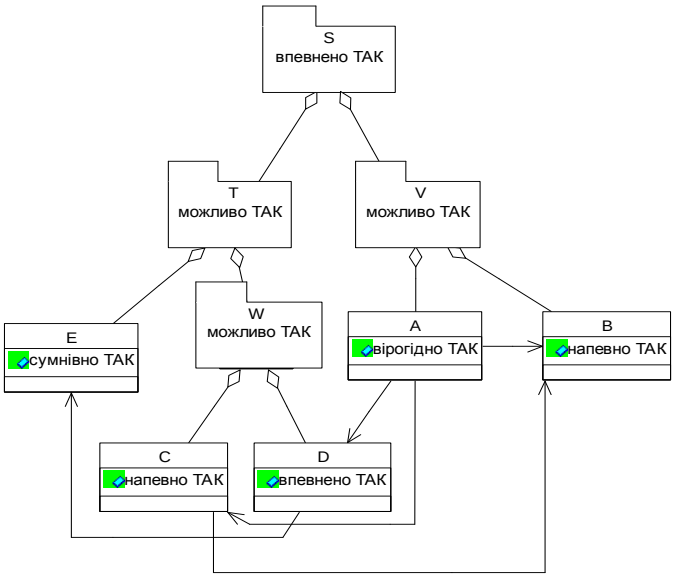


Рис. 5. Графічне представлення нечіткої програмної архітектури

У таблиці перелічені основні види трансформації нечіткої архітектури, яка ґрунтується на нечітких графах, реалізована в експериментальному середовищі і протестована на серії архітектур-зразків. Середовище надає інструментальні засоби для визначення лінгвістичних змінних, значень і нечітких графів з використанням засобів моделювання мовою UML [4-6], розширених засобами нечіткої логіки. Системний аналітик взаємодіє з інструментальними засобами, що містять правила трансформації, та з моделлю архітектури, представленій у вигляді нечіткого графу.

Основні види нечітких трансформацій програмної архітектури Таблиця

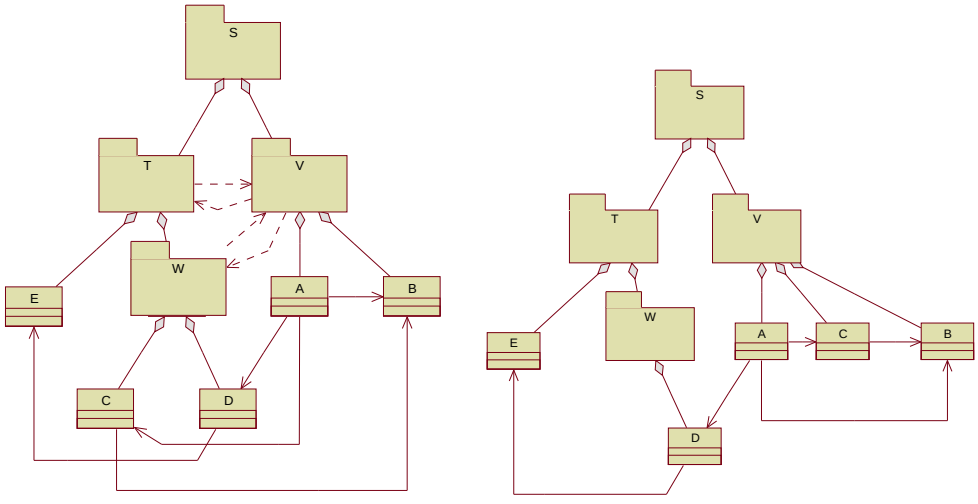
КЛАС	ТИП	ХАРАКТЕРИСТИКА
РОЗУМІННЯ АРХІТЕКТУРИ	**ПІДЙОМ**	**ПІДНЯТТЯ ВВЕРХ ЗА ІЄРАРХІЄЮ СИСТЕМИ НИЗЬКОРІВНЕВИХ ВІДНОШЕНЬ АСОЦІАЦІЇ ДЛЯ ТОГО, ЩОБ ДОСЛІДИТИ СТРУКТУРУ НА РІЗНИХ РІВНЯХ АБСТРАКЦІЇ**
	УКРИТТЯ ВНУТРІШНОСТІ/ЗОВНІШНОСТІ	**ВИКЛЮЧЕННЯ ІНФОРМАЦІЇ, ЩОБ ЗРОБИТИ СТРУКТУРУ ЗРОЗУМІЛІШОЮ ШЛЯХОМ ДЕТАЛІЗАЦІЇ АБО АБСТРАГУВАННЯ, ЩОБ ЗОСЕРЕДИТИСЯ НА ІНТЕРЕСНИХ ПРЕДСТАВЛЕННЯХ**
АНАЛІЗ АРХІТЕКТУРИ	**ДІАГНОСТИКА**	**ДЛЯ ВИСОКОРІВНЕВИХ НЕДОРЕЧНИХ РЕБЕР ГРАФУ, ПОМІСТИТИ ЇХ ВНИЗ ІЄРАРХІЇ СИСТЕМИ, ЩОБ ВИДІЛИТИ НИЗЬКОРІВНЕВІ НЕБАЖАНІ РЕБРА**

	ПРОСІЮВАННЯ	**ПОМІТИТИ КОМПОНЕНТИ, ЯКІ ГРАЮТЬ ПЕВНУ РОЛЬ В ЦІЛЬОВІЙ ЗМІНІ СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**
МОДИФІКАЦІЯ АРХІТЕКТУРИ	**ПРЯМЕ ОНОВЛЕННЯ**	**ЗМІНІТИ ОТРИМАНУ АРХІТЕКТУРУ (АБО КОНКРЕТНУ АРХІТЕКТУРУ), ЩОБ ЗРОБИТИ ЇЇ БІЛЬШ УЗГОДЖЕНОЮ З УЯВНОЮ МОДЕЛЛЮ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ (ТОБТО КОНЦЕПТУАЛЬНОЮ АРХІТЕКТУРОЮ)**
	ЗВОРОТНЕ ОНОВЛЕННЯ	**ЗМІНІТИ КОНЦЕПТУАЛЬНУ АРХІТЕКТУРУ, ЩОБ ВОНА СТАЛА БІЛЬШ УЗГОДЖЕНОЮ З КОНКРЕТНОЮ АРХІТЕКТУРОЮ**

Як приклад, наведемо застосування трансформацій підйому та прямого оновлення архітектури. При виконанні першого виду трансформації асоціації між модулями, що належать різним підсистемам дублюються відповідними асоціаціями між підсистемами, які містять зазначені модулі (рис. 6,а). Це сприяє кращому розумінню прихованих та побічних зв'язків між підсистемами вищого рівня. Пунктиром показані неявні відношення залежності, які утворились в результаті використання правила трансформації. Так, оскільки *C* пов'язаний із *B*, вірогідно підсистема *W* також може взаємодіяти із підсистемою *V*.

Пряме оновлення існуючої архітектури робиться, щоб вона стала більш сумісною із концептуальною архітектурою. Одним із способів досягнення цього є правило "перехоплення". Зазначене правило переміщає модуль або підсистему з її оригінальної підсистеми в іншу. Це безпечне перетворення, яке не зачіпає тексту програми.

Перш за все необхідно визначити найкращих кандидатів для перехоплення. Одна з можливостей полягає у прийнятті як кандидати модулів, які не пов'язані відношеннями асоціації (або використання) з іншими модулями в середині підсистеми, до якої вони належать. Для цього застосовується трансформація



а б

Рис. 6. Архітектура програмної системи після виконання трансформації "діагностика"(а) та "пряме оновлення" (б)

"Просіювання". Якщо недоречний модуль *x* асоційований нечіткими відношеннями з модулями, які усі належать підсистемі *Y*, це означає, що підсистема *Y* має "перехопити" модуль *X*. Після цього використовується правило трансформації, зміст якого наведено на рис. 7. На рис. 6,б показана оновлена архітектура після застосування зазначеного правила. При цьому модуль C переміщено з підсистеми *W* у підсистему *V*, оскільки *T*

та *W* не використовують *C*. Зазначимо, що модуль C зберігає усі пов'язані з ним відношення асоціації, що дозволяють компонентам користуватися сервісами один одного.

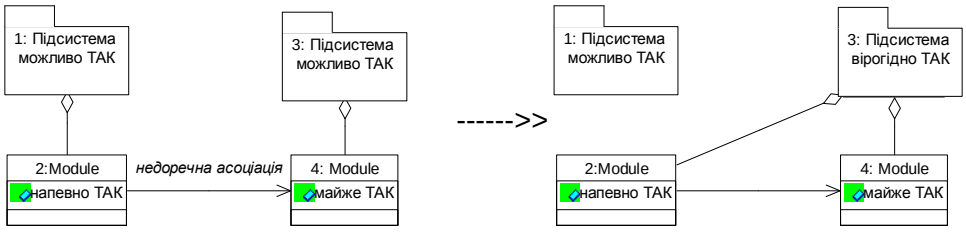


Рис. 7. Правило трансформації нечіткого графа для реалізації перехоплення модуля

Висновки

У статті запропоновано засоби для представлення складних нечітко визначених підсистем та модулів на рівні об'єктно-орієнтованої програмної архітектури. Розглянуто різні способи введення нечіткості в діаграмах класів UML, що представляють основу аналізу архітектури. Описані трансформації, які підвищують ефективність представлення графових моделей архітектури з метою поліпшення розуміння програмної структури, діагностування та аналізу архітектури та реструктуризації. Запропоновано загальний підхід до трансформації нечіткого графа моделі архітектури, який ґрунтується на понятті нечіткої семантичної збіжності шаблону правила перетворення та його входження у графову модель. Акцент робиться на конкурентних трансформаціях архітектур, що представлені нечіткими графами, – правилі підйому та прямого оновлення.

1. *Object* Management Group. Model Driven Architecture – A Technical Perspective. – 2001. – http://www.omg.org/mda.
2. *Басс Л., Клементс П., Кацман Р.* Архитектура программного обеспечения на практике – СПб: "Питер", 2005. – 576 с.
3. *Kazman, R., Bass, L., Webb, M., Abowd, G.* SAAM: a method for analyzing the properties of software architectures // Proc. of the 16th international conference on Software engineering, Sorrento, Italy, 1994. – P. 81-90.
4. *Буч Г., Якобсон А., Рамбо Дж.* UML: 2-е изд. – СПб: "Питер", 2006. – 736 с.
5. *Pilone D., Pitman N.* UML 2.0 in a Nutshell. – O'Reilly, 2005. – 234 с.
6. *O'Docherty M.* Object-Oriented Analysis and Design. Understanding System Development with UML 2.0. – John Wiley & Sons Inc., 2005. – 580 с.
7. *Прикладные* нечеткие системы: Пер. с япон. / Под ред. Т.Тэрано, К.Асан, М.Сугэно. – М.: Мир,1993. – 366 с.
8. *Сергієнко І.В., Парасюк І.М., Єршов С.В.* Нечіткий трансформаційний підхід до розробки програмних систем // Пробл. программирования. – 2004. – № 2-3. – С. 122-132.
9. *Ершов С.В.* Нечеткие графы функциональных зависимостей как основа метамоделирования программных систем // Компьютерная математика. – 2005. – № 3. – С. 139-149.
10. *Ершов С.В.* К проблеме формализации объектно-ориентированных методов разработки программного обеспечения на основе нечеткой логики // Компьютерная математика. – 2003. – № 2. – С. 62-77.
11. *Unhelkar B.* Verification and validation for quality of UML 2.0 models. – John Wiley & Sons Inc., 2005. – 313 с.