УДК 681.3.06

МОДЕЛЕ-ОРІЄНТОВАНІ МЕТОДИ ПОБУДОВИ ТА ОЦІНЮВАННЯ ПРОГРАМНИХ АРХІТЕКТУР НА ОСНОВІ НЕЧІТКИХ ГРАФІВ

І.М. Парасюк, С.В. Єршов

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

Розглядається підхід до створення програмних систем згідно моделе-орієнтованої парадигми, який відкриває шлях до здійснення трансформаційної еволюції програмних систем. Основна увага приділена формалізації нечітких графів, нечітких графових граматик та правил їх трансформації. Розроблено теоретико-категорне представлення нечітких програмних архітектур, яке надає можливість управляти процесом їх еволюційної зміни на основі прийняття рішень в нечіткому просторі моделювання стосовно характеристик функціонування цільової платформно-залежної системи.

Approach is examined to creation of the software architectures in accordance with the model-oriented paradigm, which opens a way to implementation of transformation evolution of software systems. Basic attention is spared to formalization of fuzzy graphs, fuzzy graph grammars and rules of their transformation. Theoretical categorical representation of fuzzy software architectures is developed, which gives possibility to manage the process of their evolutional change on the basis of decision-making in fuzzy space of design in relation to descriptions of functionality of target platform-specific system.

Вступ

Напрямок створення програмних систем згідно парадигми всесторонньої орієнтації на формальні моделі, що має відкрити шлях до трансформаційної еволюції програмних систем, в Інституті кібернетики ім. В.М. Глушкова розвивається з 70-х років минулого століття [1–4]. Інтерес до цього напрямку суттєво зріс з появою специфікацій архітектури програмних систем керованої моделями – MDA (Model Driven Architecture), прийнятою Групою Управління Об'єктами (OMG) [5, 6].

Трансформації моделей можуть бути представлені як правила відповідної граматики. Зокрема, якщо в основу покладені графові моделі, то правила представляються в рамках графових граматик [7] – деякого узагальнення теорії формальних граматик Хомського [8–11].

Як відомо, в теорії послідовних графових граматик замість ланцюжків символів як об'єкти розглядаються графи. Продукція графової граматики описує заміну підграфа в графі. Для визначення продукції недостатньо задати два графи, додатково слід описати перетворення включення замінюваного (другого) графа в граф, що залишився після видалення першого графа правила. Отож, продукція в послідовній графовій граматиці визначається як трійка (g_1, g_2, E), де g_1, g_2 – графи, E – перетворення включення. Відомі різні способи завдання перетворень включення [8, 9], зокрема цікавим є графове представлення правил граматики (рис. 1).

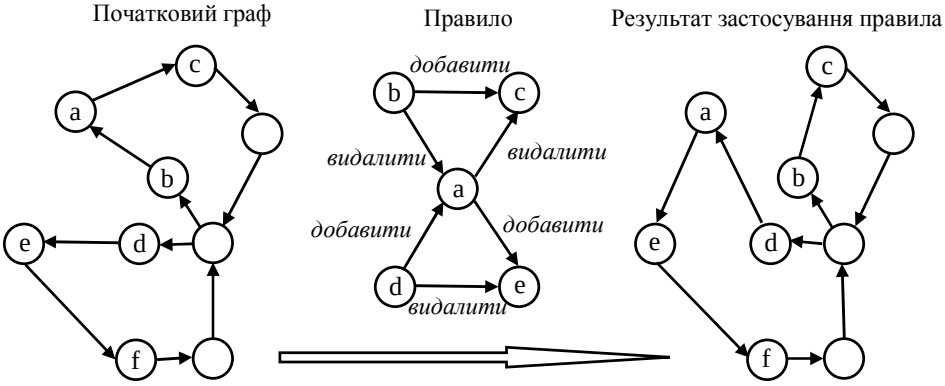


Рис. 1. Приклад застосування правила графової граматики

Графові граматики є окремим випадком Web-граматик запропонованих Пфальцем і Розенфельдом. Будь-яка графова граматика – це кортеж (A, P), де A – непорожній початковий граф (аксіома) і P – множина продукцій графової граматики. Кожна продукція має два графи, які називаються лівим і правим

графом. *A* – спеціальний випадок продукції з порожньою лівою частиною. Найбільш відомими представниками даного класу граматик є багаторівнева [10] і зберігаюча графова граматика [11].

Проте, не зважаючи на те, що графи та графові граматики застосовуються для представлення логічної структури програми та її архітектури, поки що не досліджені повною мірою методи оцінювання та аналізу програмних архітектур у нечіткому просторі моделювання, який утворюється в процесі розробки графових граматик.

Нечіткі графові моделі можуть використовуватися безпосередньо при описі програмної архітектури, яка складає основу трансформаційного походу MDA. Одним із загальноприйнятих підходів до моделювання об'єктно-орієнтованих структур, що описують програмну архітектуру, є використання мови UML, яка є набором об'єктно-орієнтованих позначень для моделювання на основі графів. Оскільки інформація щодо елементів і їх відносин в реальних системах часто невизначена або неоднозначна, виникає необхідність використання моделей на основі нечітких графів.

Мета даної роботи полягає у розробці теоретичних принципів, методів і засобів створення програмних систем на основі трансформаційного підходу MDA з використанням нечітких графових моделей програмної архітектури.

Моделе-орієнтована архітектура програмних систем

Мета MDA – забезпечити методологічне вирішення проблеми перепроектування системи програмного забезпечення кожного разу, коли з'являється нова “актуальна” технологія реалізації. Наприклад, на сучасному етапі Web-сервіси замінюють загальновідому технологію CORBA. Багато існуючих систем, можуть бути перепроектовані з допомогою технології Web-сервісів із збереженням своєї функціональності. Засобами MDA вирішуються проблеми мобільності програмних систем, що обумовлені постійними змінами в технологіях реалізації. У числі інших проблем, які можуть бути вирішені завдяки MDA – проблема продуктивності проектування на всіх рівнях розробки та проблема сумісності взаємодії складових компонент [6].

Методологічно підхід MDA розділяє моделі на два класи: платформно-незалежні моделі (ПНМ) та платформно-залежні моделі (ПЗМ). ПНМ визначені на вищому рівні абстракції, ніж ПЗМ. Незалежна від платформи модель – представлення системи з незалежної від платформи точки зору [5, 6], водночас як ПЗМ визначена як вид системи з точки зору певної платформи. Платформа – сукупність підсистем і технологій, які забезпечують послідовний набір функціональностей через інтерфейси та зразки використання, які будь-яка підтримувана цією платформою прикладна програма може використовувати без деталізації здійснення функціональності, забезпеченої платформою.

Незалежність від платформи – властивість моделі, яка надає можливість абстрагуватися від характеристик платформ специфічних технологій. Термін платформа використовується, щоб позбутися технологічних та технічних деталей, які несуттєві для фундаментальної функціональності системи (або її частини).

Поняття, які використовуються для розробки незалежних від платформи моделей, відрізняються від понять, доступних у цільових платформах, оскільки перші поняття мають бути досить узагальненими, щоб дозволити відображення на різні набори інших понять. Зазначена відмінність важлива при виборі певного способу реалізації. Для кожного поняття, представленого в незалежній від платформи моделі, має бути відповідне поняття або відповідна комбінація понять в цільовій платформі.

Модель – опис системи, що відображає структурні і динамічні аспекти цієї системи та складається з набору елементів і асоціацій, які зв'язують існуючі елементи. Метамодель – це модель, яка визначає конструкції та правила, потрібні для створення моделей.

Згідно підходу MDA розробка системи починається з побудови ПНМ цієї системи, яка згодом перетворюється в одну або більше ПЗМ.

Основна операція, що застосовується на моделях в MDA, – трансформація моделей. Це відображення одного набору моделей на інший згідно специфікації трансформації, де відображення визначає кореспонденції між елементами в початкових та цільових моделях. Процес трансформації приймає ПНМ як вхід і генерує ПЗМ у якості вихідних даних.

Специфікація трансформації MDA визначає, як і при яких умовах моделі, що відповідають початковим метамоделям, перетворені або переписані до моделей що відповідають цільовим метамоделям (рис. 2). Для того, щоб специфікація трансформацій була застосована неодноразово (незалежно від початкової моделі, до якої вона застосована), її визначають в термінах початкової і цільової метамоделі. Структурно специфікація трансформації складається із наборів правил трансформації, що конкретизують спосіб перетворення частини однієї моделі в іншу. Власне, специфікація виконується на конкретних моделях засобами трансформації.

Існує ряд різних підходів, придатних для трансформації моделей, як наприклад, реляційний, пряме маніпулювання та трансформації графів [12].

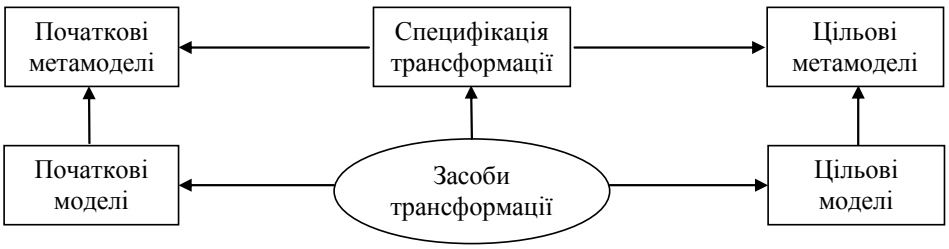


Рис. 2. Основні поняття трансформації моделей

Зокрема, дворівнева моделе-орієнтована архітектура прототипів програмних систем розроблена в рамках створеного в Інституті кібернетики ім. В.М. Глушкова методу Сигма-Дельта побудови прикладних систем. В цьому методі знання про різні предметні області інтегровані в так звану базову платформно-незалежну модель ($\sum$-систему) з подальшою трансформацією у спеціалізовані, в тому числі платформно-залежні, системи (Δ-системи), що відповідають конкретним вимогам користувача [1]. У загальних рисах процес генерації таких платформно-залежних моделей систем полягає в наступному.

Якщо $\sum(U,V,W)$ – базова інтегрована програмна система з множиною функціональних можливостей U, реалізованих множиною компонент V, між якими встановлено множину зв'язків W, то для довільної підмножини $U_i \subseteq U$ можна встановити відповідну їй конфігурацію Δ системи $\sum$, тобто

$$\sum(U,V,W) \Rightarrow \Delta(U_i,V_i,W_i), \quad V_i \subseteq V, W_i \subseteq W,$$

яка є однією з моделей даної сім'ї програмних систем. Такий підхід передбачає можливість еволюції та удосконалення сім'ї програмних систем, що задається множиною моделей $\Delta = \{\Delta_1, \Delta_2, ...\}$ шляхом удосконалення базової моделі $\sum$. Відповідні моделе-орієнтовані методи і засоби проектування, генерації і удосконалення $\sum$-систем відображені в [1].

Програмна архітектура – це виразник найважливіших знань про програмну систему, які покликані донести до практики найбільш ефективний спосіб проектування системи з урахуванням певних обмежень [13]. Сучасна точка зору, відображена у моделе-орієнтованій архітектурі MDA, робить значний акцент на використанні композиційних методі та засобів аналізу програмних архітектур у процесі розробки та подальшої еволюції програмного забезпечення.

Архітектурні структури підрозділяються на три загальні групи [13], 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 об'єктно-орієнтованих моделях для композиції модульної структури, вельми обширний [13]. 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.

Особливо важливу роль відіграють композиційні структурно-модульні методи при розробці програмних систем на основі сервіс-орієнтованої архітектури (SOA) – розподіленої мережної архітектури, в якій чітко відокремлюються забезпечені сервіси від об'єктів, які споживають ці сервіси. Сервіси взаємодіють між собою, є самодостатніми і не залежать від стану інших сервісів, що приводить до гнучкої архітектури оскільки в результаті з'єднання послуг систему можна легко синтезувати. Загалом, підхід SOA визначений як архітектура інформаційних технологій, яка забезпечує слабке зчеплення, багатократне використання, і сумісність між системами, або на платформнозалежному рівні як архітектура, що використовує технології Web-сервісів.

Базовою для метода Сигма-Дельта є функціонально- та сервіс-орієнтована програмна архітектура прототипу платформнозалежної системи $\Delta = (L,P,M,G,S,R)$, де L – інтерфейсний компонент (вхідна мова системи); P – прикладні програмні модулі; M – модель предметної області; S – набір сервісів, що реалізують системні функції; G – модель керування, яка описує інформаційну та логічну взаємодію сервісів при реалізації функцій системи; R – системний компонент специфікації взаємодії сервісів. Сервіси та компоненти цієї моделі взаємодіють за схемою дворівневої семантичної керованої системи.

Питання побудови та теоретичного обґрунтування алгоритмів і обчислюваних об'єктів теоретико-категорними засобами є одним з сучасних напрямів computer science [2]. Як розвиток метода Сигма-Дельта з метою використання ефективних композиційних моделей програмної архітектури, що складаються з програмних об'єктів-модулів за допомогою спеціальних операцій над ними, запропонований метод структурно-модульного композиційного програмування [3, 4]. Теоретико-категорне узагальнення структурно-модульного проектування засноване на тому факті, що категорним аналогом програмного модуля є поняття морфізму. Як категорні узагальнення початкових даних (результатів) програмної системи розглядаються об'єкти категорії і їх добутку, причому останні використовуються для категорного опису модуля над складними структурами даних.

Моделі та методи специфікації і оцінювання програмних архітектур

Основна мета сучасних засобів специфікації та оцінювання програмних архітектур – вміти враховувати їх адаптивність та еволюцію під час розробки та супроводження. Поточні методи трансформації в MDA не забезпечують у повній мірі ідентифікацію альтернативних перетворень, які виникають під час еволюції, і їх порівняння на основі певних якісних характеристик результуючих моделей. Моделювати набір альтернативних перетворень для початкової моделі архітектури можливо на основі побудови простору моделювання. Врахування основних джерел нечіткості інформації (несумісність, неточність, невизначеність, невпевненість і неоднозначність) в процесі трансформації моделей приводить до необхідності побудови нечіткого простору моделювання та розроблення підходу до специфікації і оцінювання архітектур на його основі [14, 15].

Еволюційні процеси можуть відбуватися на трьох основних рівнях: моделей, трансформацій та метамоделей. В еволюції моделей, зміни до початкових моделей відображаються на зміни до цільових моделей через правила трансформації. В процесі еволюції моделей описують зміни $\Delta M1$ у моделі $M1$, а потім здійснюються необхідні зміни до інших моделей, які отримують з $M1$ через деяку трансформацію. Наприклад, якщо $M2$ отримана з $M1$ через трансформацію T, (тобто $M2 = T(M1)$), мета – згенерувати $\Delta M2$ так, що, застосовуючи $\Delta M2$ до $M2$ отримують той самий наслідок, що дає повторна трансформація зміненого $M1$ (тобто $M2 \otimes \Delta M2 = T(M1 \otimes \Delta M1)$).

$$\begin{array}{ccc} M1 & \xrightarrow{T} & M2 \\ \Delta M1 \downarrow & & \downarrow \Delta M2 \\ M1' & \xrightarrow{T} & M2'. \end{array}$$

При еволюції трансформацій, зміни до визначення трансформацій, які використовуються, щоб генерувати інші моделі, мають бути відображені на зміни до цільових моделей. Основна властивість еволюції трансформацій полягає у тому, що вона може бути зведена до еволюції моделей. Оскільки правила перетворення написані на мові, визначеній засобами метамоделей, кожна трансформація представляє собою окрему модель. Така метатрансформація має достатньо інформації, щоб описати, як зміни правил перетворення впливають на моделі, які вироблені на основі цих правил.

MDA визначає тільки один вимір класифікації для моделей, заснований на відокремленні ПНМ і ПЗМ. Цей вимір указує рівень абстракції моделей. Інший вимір походить з відмінності моделей на основі функціональних аспектів. Приклади аспектів – управління паралелізмом, безпека, мережна розподіленість та обробка помилок. Очевидно, запропонований простір моделювання придатний до оцінювання моделей на основі більшого набору критеріїв, ніж класифікація ПНМ/ПЗМ.

Простір моделювання – багатовимірний простір над рядом незалежних вимірів, кожен з яких формує окрему множину координат, а кожна точка в такому просторі представляє трансформацію, яка застосовується до моделі, що є екземпляром цільової метамоделі.

Припустимо, що ми маємо початкову модель (рис. 3), яка містить елементи моделі $a1$, $a2$, $a2$,..., an.

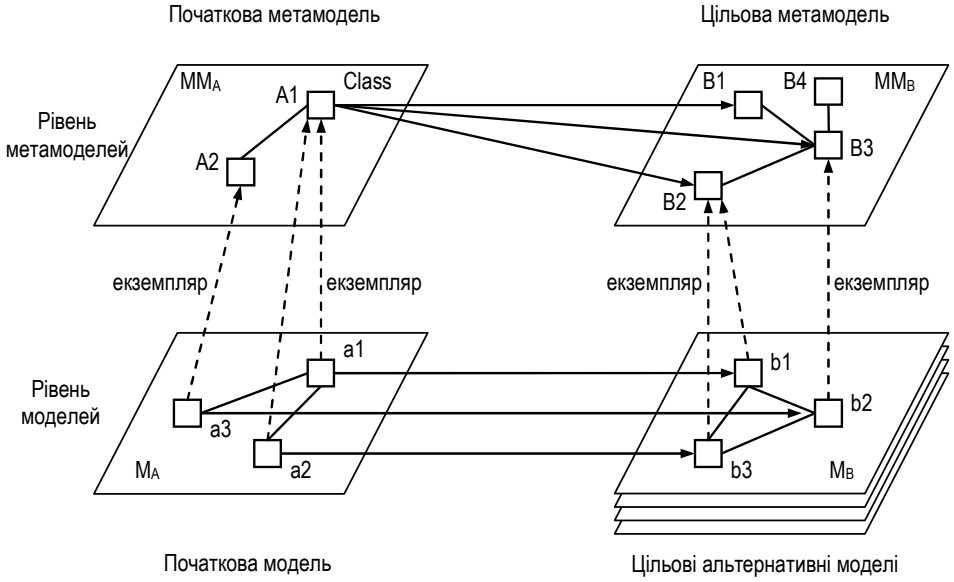


Рис. 3. Приклад альтернативних трансформацій для наданої початкової моделі

Кожен елемент моделі в початковій моделі визначає один вимір у просторі моделювання. Елемент моделі, який визначає вимір, – екземпляр конструкції з початкової метамоделі. Для цієї конструкції визначений набір

елементів з цільової метамоделі, який використовується, щоб сформувати множину координат для виміру. Конструкція з початкової метамоделі, може відображатися на кожну конструкцію у цьому наборі.

Розглянемо вимір для елемента $a1$ в початковій моделі на рис. 3. Елемент $a1$ – екземпляр конструкції $A1$ початкової метамоделі. $A1$ може бути відображена на елементи $B1$, $B2$ або $B3$ в цільовій мета моделі, що визначає множину координат $\{B1, B2, B3\}$ для виміру $a1$. Оскільки $a2$ – також екземпляр $A1$, така ж множина координат задана для виміру $a2$. Припустимо, що конструкція $A2$ в початковій метамоделі може бути відображена на одну з конструкцій $B2$, $B3$ або $B4$. Тому, множина координат для виміру $a3$ – $\{B2, B3, B4\}$.

Точки в просторі моделювання інтерпретовані як альтернативні перетворення початкової моделі. Для кожної початкової моделі цільова модель – точка над вимірами, відповідними цій початковій моделі. Точка в просторі моделювання S представлена як кортеж з компонентами для кожного виміру і координатою в цьому вимірі:

$$(d_1.cd_1, d_2.cd_2,\ldots, d_n.cd_n),$$

де d_i – ім'я виміру, cd_i – координата точки в цьому вимірі. На рис. 4 показано точку з координатами ($a1.B2$, $a2.B2$, $a3.B3$), яка відповідає цільовій моделі на рис. 3.

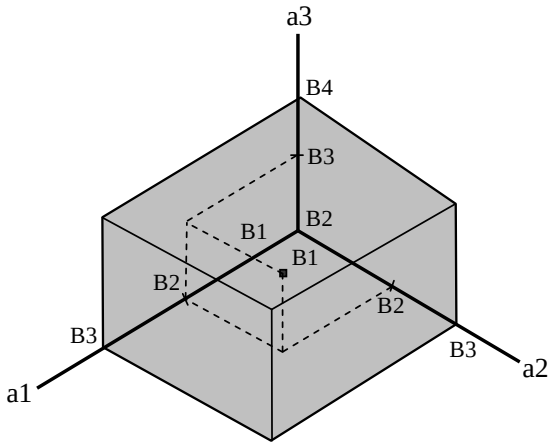


Рис. 4. Простір моделювання для початкової моделі

У процесі моделе-орієнтованої розробки на основі простору моделювання мають бути вибрані альтернативні трансформації для наданої моделі. Різні вимоги щодо результуючих цільових моделей приводять до функціонально еквівалентних реалізацій, які відрізняються за атрибутами якості [16].

Властивості модулів набуті з моделі властивостей, роль якої у нашому прикладі відіграє модель розширюваності. Ця модель використовується, щоб сформувати множину з двома координатами: {Розширювана, Нерозширювана}. Заснована на цьому рішенні операція вибору використовується, щоб скоротити простір S. Наприклад, якщо модулі $\{b_1, b_3, b_3\}$ – елементи цільової моделі, тоді:

$$S_R = \{b \in M_B \mid b1.\text{Розширювана} \wedge b2.\text{Розширювана} \wedge b3.\text{Нерозширювана}\}.$$

Властивості розширюваності об'єднані в просторі моделювання S_R^2, що використовує операцію об'єднання, визначену понад двома просторами моделювання:

$$S_U = S_1 \cup S_2,$$

де $dim(S_U) = dim(S_1) = dim(S_2)$.

Нехай функція $dim(S)$ повертає набір вимірів для наданого простору S, а функція $cSet(dimension, S)$ повертає множину координат для наданого виміру в наданому просторі S. Множина координат виміру d в S_U – декартовий добуток, визначений в таким способом:

$$cSet(d, S_U) = cSet(d, S_1) \times cSet(d, S_2).$$

Точки в просторі S_U визначені на основі точок з просторів S_1 і S_2 наступним способом:

$$S_U = \{(d_1.(p_1', p_1'')),\ldots(d_n.(p_n', p_n''))\ \Big|\ \begin{array}{l} d_i \in dim(S_U), p_i' \in cSet(d_i, S1), p_i'' \in cSet(d_i, S2), \\ (d_1.p_1', \ldots, d_n.p_n') \in S_1, \\ (d_1.p_1'', \ldots, d_n.p_n'') \in S_2, i = \overline{1,n}. \end{array}$$

Операція об'єднання може бути застосована на просторах S і S_R, щоб створити простір моделювання для розширюваних схем XML:

$$S_R^2 = S \cup S_R .$$

Даний підхід забезпечує явне представлення властивостей, і може використовувати критерії визначення підпростору, засновані на їх виборі. Вищерозглянутим характеристикам функціонування притаманна чітка природа, тобто використовуються двозначні оцінки: модуль, або володіє певною властивістю, або не володіє. Очевидно, що засоби представлення моделей у нечіткому просторі моделювання дозволяють більш адекватно представити та оцінювати програмні архітектури, ніж двозначна характеристика показників функціонування.

Специфікація метамоделі нечіткого програмного модуля – це набір $[T,(\mu_T, D_M),(\mu_1, D_1),(\mu_2, D_2),...,(\mu_n, D_n)]$, де T – ім'я типу модуля, μ_T – ступінь його належності до простору моделювання, D_M – область визначення функції належності модуля, μ_i – функція належності атрибута модуля T і D_i – область визначення μ_i. Присутність ступеня належності μ_T, зазвичай визначена на реальному інтервалі [0,1], де крайні значення 0 і 1 означають, відповідно, що даний модуль (не) належить даному типу модуля. У разі нечітких модулів, належність до типу нечіткого модуля характеризується значенням відповідної функції належності.

Нечіткий модуль може бути представлений у нечіткому просторі цільової моделі як $[T,(\mu_T, V_M),(P_1, V_1),(P_2, V_2),...(P_n, V_n)]$, де значення функції належності V_M залежить від окремих трансформацій, які виробляють зазначений модуль на основі початкової моделі. Властивості $P_1, P_2,..., P_n$ задають характеристики модуля за кожним окремим виміром. Наприклад, значення функції належності модуля μ_T залежить від значень належності властивостей "Розширюваність" та "Автономність" модуля. Нечіткі лінгвістичні терми $V_1, V_2,..., V_n$ такі як "слабка", "незначна", "середня", "істотна", "значна" можуть використовуватися для уточнення координат модуля у просторі моделювання.

У разі врахування кількох показників використовуються нечіткі правила, які на основі значень показників $V_1, V_2,..., V_n$ виробляють нечітку оцінку V_M, яку можна інтерпретувати як інтегральний показник застосовності трансформації у нечіткому просторі. За основу приймаються такі трансформації, які виробляють модулі із найвищим значенням показників функціонування у нечіткому просторі.

Нечіткі моделі [17] можуть використовуватися безпосередньо при описі програмної архітектури. В цьому випадку системи можуть бути визначені нечіткими продукційними моделями, у вигляді нечітких функціональних моделей, нечіткими реляційними (у вигляді набору відношень) моделями, нечіткими графами, які використовуються для представлення неточно визначених залежностей [18]. Зокрема, нечіткі графи використовуються для наближеного представлення функцій і відношень, щоб апроксимувати одну вихідну змінну, залежну від n вхідних змінних x_i $(1 \le i \le n)$.

Один із загальноприйнятих підходів до моделювання об'єктно-орієнтованих структур, що описують програмну архітектуру є використання мови UML яка є набором об'єктно-орієнтованих позначень для моделювання на основі графів [14]. Оскільки інформація щодо елементів та їх відношень в реальних системах часто невизначена або неоднозначна, виникає необхідність використання моделей на основі нечітких графів.

Для формалізації нечіткості основних понять, що входять до складу нечітких графів [12, 18, 19], використаний категорний підхід як розділ математики, що вивчає найбільш загальні властивості відношень між математичними об'єктами, які не залежні від внутрішньої структури об'єктів.

Будь-який нечіткий граф побудований на нечітких точках типу

$$\text{якщо } x_1 = A_1 \text{ та } ... \; x_n = A_n, \text{ то } y = B ,$$

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

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

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

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

Функція належності $A \times B$ задається використанням $\wedge$ як оператора логічного множення, який, зазвичай, визначений як мінімум

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

Нечіткий граф f^* може бути представлений як суперпозиція m нечітких точок співвідношенням

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

Розглянемо нечіткий граф, побудований на нечітких точках. У випадку $n=1$ отримуємо бінарний нечіткий граф $\langle A, \alpha \rangle$ – це пара, що складається з нечіткої множини *A*, елементи якої є об'єктами категорії **Fuzz**([0,1]), і нечіткого відношення $\alpha : A \to A$. При цьому множина істинності є інтервал [0,1], а множина носія – вершина графа.

Морфізмом нечітких графів $f : A \to B$ називається пара $f = \langle f_E : E_A \to E_B,\ f_N : V_A \to V_B \rangle$ морфізмів (визначених над дугами E і вершинами N нечіткого графа відповідно), таких, що $f_N \circ s_A = s_B \circ f_E \wedge f_N \circ t_A = t_B \circ f_E$. Нечіткі графи разом з множиною їх морфізмів і визначеною покомпонентно композицією морфізмів $g \circ f = \langle g_E \circ f_E,\ g_N \circ f_N \rangle$ утворюють категорію **FGraph**.

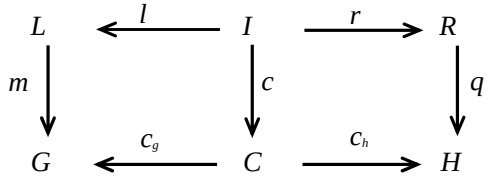
Оцінювання програмних архітектур на основі нечітких графів вимагає представлення моделі, в якій вершини є набором значень нечітких атрибутів. З цією метою поняття нечіткого графа розширене до (L,U)-графа, в якому елементи множини істинності, відповідних міткам вершин і дуг є елементами довільно вибраної решітки, а не тільки інтервалу [0,1]. Нехай L – повна решітка значень істинності, U (назване простором елементарних тверджень, атрибутів) – довільно вибрана множина. (L,U)-нечіткий граф, G – направлений граф, в якому кожна дуга e помічена значенням з множини L, а кожна вершина n помічена елементами L-значимої множини (U_n, ζ_n), де $U_n \subseteq U$ – множина елементарних тверджень, заданих у вершині n, а $\zeta_n : U_n \to L$ – функція, що призначає значення з множини L кожному елементу U_n. Видаляючи мітки вершин і ребер G, отримаємо граф, який представляє собою "чітку" модель програмної архітектури (граф носія нечіткого графа G).

Трансформаційні методи розробки програмних систем в нечіткому просторі

Нечіткі моделі у трансформаційних методах використовуються з двома цілями: для отримання найбільш "якісної" реалізації шляхом оцінки можливих перетворень в нечіткому просторі моделювання; для врахування зміни вимог користувача, ступінь невизначеності яких може варіюватися, що викликає автоматичну генерацію і вибір найбільш відповідного способу реалізації (декомпозиції) системи за наявності відповідних перетворень. Необхідними додатковими елементами даного підходу є подальша дефазіфікация нечіткої платформнозалежної моделі, що була згенерована, а також збереження і представлення нечіткого простору моделювання як робочого продукту в термінології метамоделі SPEM [20].

Трансформація нечітких графів включає два окремі поняття. Одне з них – це правило заміни, яке встановлює відповідність між вершинами і може бути формалізоване як часткова функція на множині вершин. Інше поняття – це "збіжність", при її наявності можуть відбуватися відповідні правила заміни. Збіжність має виділяти певні підграфи в графах, які підлягають трансформації.

Продукція над нечіткими графами $p = (I \xrightarrow{l} L;\ I \xrightarrow{r} R)$ – множина, що складається з нечітких графів *L, I, R* і двох морфізмів $I \xrightarrow{l} L$ та $I \xrightarrow{r} R$. Безпосереднє перетворення $G \Rightarrow H$ на основі *p* та *m* (або $G \overset{p,m}{\Rightarrow} H$) з нечіткого графа *G* у нечіткий граф *H* задано двома діаграмами (1) та (2) в категорії нечітких графів **FGraph.**



(1) (2)

У процесі застосування продукції при наявності нечіткої збіжності між $L \cup I$ та елементами початкового графа G, видаляються елементи *L*, додаються елементи *R*, елементи *I* залишаються незмінними. Так, кожна продукція задає точку (L,R) в нечіткому просторі моделювання, де *L* – позначення виміру, а *R* – нечітка координата в цьому вимірі.

Застосуванням трансформації нечіткого графа моделі архітектури *M* програмної системи (тобто моделі UML) є перетворення нечіткого графа моделі заміною підграфа, що виділяє збіжність у лівій частині правила на підграф, визначаючого правило трансформації. Застосування включає такі 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 нечіткій збіжності (отримання контекстного нечіткого графа); вставка у контекстний граф нечіткого графа g^* і отримання результуючого графа моделі *M'*.

До формалізації нечітких графових граматик [18, 19], які є узагальненням послідовних графових граматик та враховують основні види нечіткості, що виникають як при побудові базових категорій нечітких об'єктів, так і при опису трансформацій нечітких графів, породжуваних нечіткими множинами, використовується теоретико-категорний підхід.

Множину всіх нечітких графів, побудованих на множині міток Σ, позначимо G_Σ. Нечітка графова граматика – це система $FGG=(S,P,\Delta,L,\varphi)$, де $S\in G_\Sigma$ – початковий нечіткий граф граматики *FGG*, *P* – кінцева множина правил трансформації графів, $\Delta\in\Sigma$ – множина термінальних символів, *L* – множина ваги (наприклад, дистрибутивні решітки з 0 і 1), $\varphi: P\to L$, $\varphi(p)$ – ступінь належності виведення продукції p. Породжувана нечіткою граматикою *FGG* мова включає всі нечіткі графи $G\in G_\Sigma$, помічені на множині міток Δ, які виводяться з початкового графа *S* за допомогою застосування правил з множини *P*, тобто

$$L(FGG)=\{G\in G_\Sigma \mid S\overset{*}{\underset{P}{\Rightarrow}} G\}.$$

Нехай задана граматика *FGG* і нечіткі графи $G_u, G_v\in G_\Sigma$, де G_Σ – множина об'єктів категорії **FGraph**. При цьому G_u безпосередньо породжує G_v зі ступенем $\min(\mu(g(p)),\varphi(p))$, якщо знайдеться така нечітка збіжність $g(p): G_u\to G_{lp}$, яка виділяє у графі G_u підграф, відповідний лівій частині правила зі ступенем $\mu(g(p)): G_u \overset{\mu(g(p))*\varphi(p)}{\underset{p}{\to}} G_v$.

Передбачається, що трансформації над окремими компонентами виконуються одночасно. Таке паралельне виконання розподілених дій може бути виражене паралельною продукцією на нечітких графах. Продукція нечітких графів $\hat p_1+\hat p_2=(\hat I_1+\hat I_2\xrightarrow{\hat l_1+\hat l_2}\hat L_1+\hat L_2;\ \hat I_1+\hat I_2\xrightarrow{\hat r_1+\hat r_2}\hat R_1+\hat R_2)$ називається паралельною продукцією, що складається з $\hat p_1$ і $\hat p_2$, де $\hat p_1=(\hat I_1\xrightarrow{\hat l_1}\hat L_1;\hat I_1\xrightarrow{\hat r_1}\hat R_1)$ і $\hat p_2=(\hat I_2\xrightarrow{\hat l_2}\hat L_2;\hat I_2\xrightarrow{\hat r_2}\hat R_2)$ – окремі продукції нечітких графів, $\hat L_1+\hat L_2$, $\hat I_1+\hat I_2$ і $\hat R_1+\hat R_2$ – копродукти графів і $\hat l_1+\hat l_2$ а $\hat r_1+\hat r_2$ – породжені морфізми.

Нехай P^+ – найменше розширення множини *P*, яке включає всі паралельні продукції $\hat p_1+\hat p_2$ для $\hat p_1,\hat p_2\in P^+$. Множина усіх можливих специфікацій архітектури, що задається граматикою *FGG* представлена класом всіх можливих перетворень, що починаються з S та використовують множину P^+ продукцій графів, тобто $S\overset{P^+}{\Rightarrow}{}^*_{di} G$.

Нечіткі розподілені графові граматики є певним узагальненням нечітких графових граматик, що дозволяє описувати припустимі перетворення мережних структур розподілених компонентів (модулів). Для цього категорія нечітких графів узагальнена до категорії розподілених нечітких графів (FD-графів), об'єктами якої виступають графи компонентів, кожний з яких, в свою чергу містить нечіткий граф [19]. Формально визначені необхідні і достатні умови для здійснення перетворень FD-графів, при яких не порушується цілісність його структури.

Зміна форми мережі компонентів (модулів) протягом кроку перетворення надає можливість моделювати динамічні мережні структури компонентів, що задають сервіс-орієнтовану архітектуру. При цьому в одному правилі нечіткої граматики можливо задавати зміну структури графа одночасно в декілька різних мережних компонентах. Наприклад, паралельно створюються порти комунікації як компонента-клієнта, так і компонента, що забезпечує сервіс.

На рівні платформно-залежної моделі розглядаються аспекти поведінки системи, пов'язані з її функціональністю, причому використовується тільки основна обчислювальна інфраструктура, заснована на нечітких компонентах, портах, і з'єднувачах розподілених систем та їх типах відповідно.

У правилах трансформації нечітких графів передбачається, що порти можуть бути відкриті або закриті, з'єднувачі можуть бути створені або видалені, але завжди абстрагуються від механізмів для пошуку правильних партнерів-компонентів і відносять це питання до визначеного для платформи стилю. Крім того, основним механізмом комунікації виступає обмін повідомленнями через встановлені з'єднувачі (канали).

Рівень сервіс-орієнтованої архітектури моделюється як система перетворень нечітких розподілених графів і представляє специфічні механізми для публікації і пошуку сервісів. Правила трансформації нечітких графів включають розширений набір нечітких елементів ПНМ-рівня. Центральний елемент тут – ServiceDescription, що описує певний сервіс. Відношення knows зазначає які компоненти мають доступ до опису. Існування такого відношення knows – попередня умова для з'єднання з сервісом. Окрім повідомлень "Запит" і "Відповідь",

використовуються три спеціальні типи повідомлень для взаємодій з послугами пошуку сервісу. Перше представляє опис сервісу пошуку для публікації, друге посилається на тип порту для якого замовник сервісу вимагає відповідного сервісу, третє повертається сервісом відкриття і містить опис, що задовольняє запиту.

Аспекти забезпечення сервісу, безпеки і мобільності можуть бути представлені більш специфічними рівнями, які формують ієрархію уточнення вниз до моделей специфічних платформ.

Обидві вищезазначені моделі представляють різні рівні абстракції платформи, тому коректне співвідношення між ними визначене як відповідне поняття конкретизації архітектури. Для кожного кроку трансформації $s = (G \Rightarrow P)$ в платформно-незалежній моделі, що задається нечіткою графовою граматикою $\wp^{нез}$, послідовність застосування правил $s^{SOA} = (G^{SOA} \Rightarrow^{*} P^{SOA})$ нечіткої графової граматики $\wp^{SOA}$ сервіс-орієнтованої моделі являє собою коректну конкретизацію кроку s якщо G^{SOA}, P^{SOA} – структурна конкретизація нечітких графів G та P відповідно. Даний принцип використовується для перевірки коректності специфікації нечіткої моделі сервіс-орієнтованої архітектури.

До характеристик сервіс-орієнтованої архітектури (які також називають її атрибутами [21]) належать такі як сумісність (інтероперабельність), надійність, придатність сервісів, рівень безпеки, виробність, масштабованість, розширюваність, адаптивність тощо. Трансформації сервіс-орієнтованої архітектури з урахуванням окремої характеристики можуть бути задані відповідними правилами нечітких графових граматик.

Нечіткий трансформаційний підхід до моделювання сервіс-орієнтованих архітектур дозволяє оцінити якість альтернативних трансформацій системи при невизначеності вимог до системи (нечіткий простір моделювання). При цьому в нечіткій моделі зазначені альтернативи не зникають повністю при конкретизації платформно-залежної моделі системи і враховуються якщо умови функціонування програмної системи змінюються.

Вибір відповідного рішення серед декількох альтернативних трансформацій здійснюється в нечіткому просторі моделювання з урахуванням численних параметрів оцінки (атрибутів). При цьому ваги і оцінки кожного параметра задаються нечіткими числами або словами, тому реалізовано метод нечіткого багатоатрибутного прийняття рішень.

Для кожного правила нечіткої графової граматики $p = (I \xrightarrow{l} L; I \xrightarrow{r} R)$ що може бути застосована до моделі системи визначимо описи, що дають словесну оцінку атрибутів трансформацій і відповідні їм лінгвістичні змінні. Нехай P – множина альтернативних правил трансформації, A – множина атрибутів альтернативних трансформацій, а добуток $P \times A$ – область визначення змінних, тоді змінні, значення яких знаходяться в множині D, назвемо лінгвістичними змінними $L(p,a)$, які зазначають оцінку атрибуту a альтернативної трансформації p. Словник D описів лінгвістичних змінних складається шляхом зіставлення описів L_i і нечітких чисел U_i у інтервалі $[0,1]$. При цьому зіставлення утворюється на основі домовленості з фахівцями, тобто якщо отримана лінгвістична оцінка $L_i (i = 1,...,k)$, то використовуючи словник D, можна отримати нечітку оцінку U_i.

Елементи нечіткого графа $(R - I)$, які додаються в результаті трансформації визначають точки (L,R) які мають бути оцінені в нечіткому просторі моделювання. Пов'яжемо з правилом трансформації атрибут a. Тоді оцінка за атрибутом a правила нечіткої графової граматики задано як Т-норму $\overset{T}{*}$ на всіх елементах $(R - I)$:

$$T_a(R - I) = \mu_a(r_1) \overset{T}{*} \mu_a(r_2) \overset{T}{*} ... \overset{T}{*} \mu_a(r_n),$$

де $R - I = r_1 \cup r_2 \cup ... \cup r_n$.

Решта атрибутів $A - a$ в результаті застосування окремого правила залишається незмінною.

При багатоатрибутному прийнятті рішень стосовно трансформацій перш за все здійснюється оцінка атрибутів і властивостей моделей, пов'язаних з багатоатрибутними рішеннями. В результаті шляхом лінгвістичного зіставлення результату операції $T(R - I)$ зі словником D описів лінгвістичних змінних отримуємо вербальне представлення характеристики певної трансформації, наприклад "середня", "істотна", "значна" для атрибуту адаптивності.

На основі оцінки атрибутів робиться оцінка альтернативних трансформацій. Тобто, процес отримання лінгвістичної оцінки Z функції F за лінгвістичним представленням оцінки альтернативних трансформацій і їх атрибутів (значенням лінгвістичних змінних $L_1, L_2, ..., L_k$), можна записати у вигляді

$$Z = F(L_1, L_2, ..., L_k).$$

Власне, процес оцінювання F можна побудувати у вигляді наступних кроків:

1. Переклад значень L_i лінгвістичної змінної i-го атрибута у нечіткі числа U_i на основі словника D.

2. Застосування оцінювальної функції, в результаті якого визначається нечітке число U_o узагальненої оцінки об'єкта на основі нечітких чисел оцінок атрибутів. Лінійна узагальнена функція оцінки може бути

представлена як $F(U)=\sum_{i=1}^{k} c_i U_i$, де нормалізований вектор коефіцієнтів $\sum_{i=1}^{k} c_i = 1$. Побудова такої функції може також здійснюватися на основі навчальних прикладів як процес формування таких нечітких правил, щоб при отриманні вхідних значень функція видавала коректне вихідне значення, яке має найменшу погрішність. При цьому в ролі навчальних прикладів можуть виступати результати класифікації атрибутів модулів на основі програмних метрик.

3. Лінгвістична апроксимація шляхом співставлення словника і нечіткої узагальненої оцінки. При цьому визначається лінгвістичне представлення Z для узагальненої оцінки об'єкта. Операція лінгвістичного співставлення визначається як:

$$Z_0 = W_o \varepsilon D \left| [\max\{\mu_{W0}(t) \wedge \mu_{V0}(t)\} \geq \max \max\{\mu_W(t) \wedge \mu_V(t)\}] \right.,$$

де $W_0 \left| P(W_i) \right.$ позначає W_0 що задовольняє умові P, μ_{V0} зазначає функцію належності U_o, μ_{Wi} – функції належності слів W_i включених в словник D.

У процесі прийняття рішень вибирається трансформація, дефазифікація функції належності якої U_o, дає найбільше значення.

Висновки

Розглянутий підхід надає можливість керувати процесом еволюційної зміни моделей програмної архітектури на основі прийняття рішень в нечіткому просторі моделювання стосовно атрибутів функціонування цільової платформнозалежної архітектури. В процесі зміни вимог до рішення, яке надається модулями цільової платформи, процес трансформації може бути направлений продукціями, заданими нечіткою графовою граматикою. Зрозуміло, що оскільки трансформації та метамоделі також представляються нечіткими графами, то їх еволюція також задається у вигляді нечітких графових граматик.

Описану процедуру прийняття рішень щодо вибору трансформації на основі нечіткого представлення характеристик програмних архітектур реалізовано в експериментальному середовищі і протестовано на серії архітектур-зразків. Це середовище надає інструментарій для трансформації нечітких моделей та метамоделей. При цьому моделі архітектури та їх трансформації перетворюються у систему нечітких лінгвістичних продукційних правил, які для більшої точності представлення нечітких графових моделей автоматично перетворюються до набору, так званих, немережевих правил.

1. *Парасюк И.Н.*, *Сергиенко И.В.* Пакеты программ анализа данных: технология разработки. – М.: Финансы и статистика, 1988. – 159 с.
2. *Сергиенко И.В.*, *Парасюк И.Н.*, *Провотар А.И.* О применении категорных методов в computer science // Кибернетика и системный анализ. – 2000. – № 4. – С. 3–11.
3. *Парасюк И.Н.*, *Провотар А.И.*, *Заложенкова И.А.* О методологии композиционного структурно-модульного программирования // Кибернетика и системный анализ. – 1995. – № 1. – С. 146–154.
4. *Парасюк И.Н.*, *Калита А.В.*, *Провотар А.И.* CASE – система структурно-модульного программирования: алгебра морфизмов // Кибернетика и системный анализ. – 1993. – № 2. – С. 140–146.
5. *Miller J.*, *Mukerji J.* MDA Guide Version 1.0. OMG Document: omg/2003-05-01, 2003. www.omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf
6. *Kleppe A.*, *Warmer J.*, *Bast W.* MDA Explained. The Model Driven Architecture: Practice and Promise. – Addison-Wesley Professional, 2003. – 192 p.
7. *Dictionary* of Information Science and Technology. – London, Melbourne, Singapore: IDEA Group Reference, 2007. – 770 p.
8. *Nagl M.* A Tutorial and Bibliographical Survey on Graph Grammars // Lecture Notes in Computer Science volume 73: Graph-Grammars and Their Application to Computer Science and Biology. New York: Springer-Verlag, 1979. – P. 70–125.
9. *Ehrig H.* Introduction to the Algebraic Theory of Graph Grammars (a survey) // Lecture Notes in Computer Science volume 73: Graph-Grammars and Their Application to Computer Science and Biology. New York: Springer-Verlag, 1979. – P. 1–69.
10. *Rekers J.*, *Schurr A.* Defining and parsing visual languages with layered graph grammars // J. of Visual Languages and Computing. – 1997. – Vol. 8, N 1. – P. 27–55.
11. *Zhang D.-Q.*, *Zhang K.*, *Cao J.* A context-sensitive graph grammar formalism for the specification of visual languages // The Computer. – 2001. – Vol. 44, N 3. – P. 186–200.
12. *Сергієнко І.В.*, *Парасюк І.М.*, *Єршов С.В.* Нечіткий трансформаційний підхід до розробки програмних систем // Проблемы программирования. – 2004. – № 2/3. – С. 122–132.
13. *Басс Л.*, *Клементс П.*, *Кацман Р.* Архитектура программного обеспечения на практике. – СПб: Питер, 2005. – 576 с.
14. *Парасюк І.М.*, *Єршов С.В.* Методи аналізу програмних архітектур, представлених нечіткими графовими моделями // Проблемы программирования. – 2006. – № 1/2. – С. 101–110.
15. *Ершов С.В.* К проблеме формализации объектно-ориентированных методов разработки программного обеспечения на основе нечеткой логики // Компьютерная математика. – 2003. – № 2. – С. 62–77.
16. *Основы* инженерии качества программных систем / Ф.И. Андон, Г.И. Коваль, Т.М. Коротун, В.Ю. Суслов / Под ред. И.В. Сергиенко. – Киев: Академпериодика, 2002. – 504 с.
17. *Сергієнко І.В.*, *Парасюк І.М.* Нечіткі інформаційно-діагностичні технології: проблеми становлення // Вісн. НАН України. – 2002. – № 7. – С. 21–28.
18. *Парасюк И.Н*, *Єршов С.В.* Категорный подход к построению нечетких графовых грамматик // Кибернетика и системный анализ. – 2006. – № 4. – С. 80–96.
19. *Парасюк И.Н*, *Єршов С.В.* О трансформации нечетких графов, задаваемых FD-грамматиками // Кибернетика и системный анализ. – 2007. – № 2. – С. 129–147.
20. *OMG.* Software Process Engineering Metamodel Specification. OMG document formal/05-01-06. – 2005. – 99 p.
21. *O'Brien L.*, *Bass L.*, *Merson P.* Quality Attributes and Service-Oriented Architectures. Technical Note, CMU/SEI-2005-TN-014. – 2005. – 29 p.