УДК 004.413 (045)

Гріненко О.О.; Гріненко С.А.

КОНЦЕПТУАЛЬНІ ОСНОВИ МЕТОДІВ МОДЕЛЮВАННЯ ЕКОСИСТЕМ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Grinenko O.O., Grinenko S.A. Conceptual foundations of methods of software ecosystem modeling.

The study of software ecosystems consists of the analysis and construction of representations. This paper proposes to consider conceptual methods of software ecosystems modeling according to information interactions between its components. There were made analysis of several scientific papers, which deal with models and modeling of software ecosystems, i.e. i*, UML, Petri nets. The purpose of the article is to develop a conceptual basis of modeling techniques of software ecosystems. There were described the main components of software ecosystems, their relationships and methods for their modeling. It was paid great attention to Model Driven Architecture (MDA), which are used for software ecosystem modeling. MDA provides a common environment for software development that promotes interaction between groups of developers in the process of data analysis and validation of systems; engineers can find and fix errors on the early stages of designing of a system, when time and financial implications of changes in the system are minimized; MDA promotes the reuse of models to improve the system and derived systems with advanced features. Another important problem, which is considered in the article is an assessment of the software, its environment and components.

Гріненко О.О., Гріненко С.А. Концептуальні основи моделювання екосистем програмного забезпечення. Дослідження екосистем програмного забезпечення складається з аналізу і побудови уявлень про них. У даній статті пропонується розглянути концептуальні методи моделювання екосистем програмного забезпечення відповідно до інформаційної взаємодії між їх компонентами. Також у роботі описуються основні компоненти екосистем програмного забезпечення, зв'язки між ними і методи їх моделювання. Було приділено велику увагу модельно-орієнтованому підходу (Model Driven Architecture – MDA), який використовується для моделювання екосистем програмного забезпечення.

Гриненко А.А., Гриненко С.А. Концептуальные основы моделирования экосистем программного обеспечения. Исследование экосистем программного обеспечения состоит из анализа и построения представлений о них. В данной статье предлагается рассмотреть концептуальные методы моделирования экосистем программного обеспечения в соответствии с информационным взаимодействием между их компонентами. Также в работе описываются основные компоненты экосистем программного обеспечения, связи между ними и методы их моделирования. Было уделено большое внимание модельно-ориентированного подхода (Model Driven Architecture - MDA), который используется для моделирования экосистем программного обеспечения.

Вступ

На сьогодні стан програмного забезпечення (ПЗ) можна охарактеризувати глобальним поширенням, надвеликим обсягом, складними взаємодіями, інтеграцією в повсякденне життя людей, а також появою нових видів взаємодій при управлінні.

Така ситуація штовхає вчених та розробників на нові шляхи дослідження програмного забезпечення. Однією з нових тенденцій є розгляд систем програмного забезпечення з позиції біологічних екосистем. Такий підхід отримав назву екосистема програмного забезпечення (ЕПЗ). Як показав досвід таких брендів як Microsoft, Google, Apple побудова ЕПЗ є ефективним шляхом розвитку ПЗ.

Тому питання побудови моделей ЕПЗ є актуальним. Швидкий розвиток інформаційний технологій спонукає до систематизації ПЗ, яке можна регулювати на всіх стадіях його життєвого циклу. Правильно організована робота розробників, замовників, та користувачів програмного забезпечення – запорука успіху для компаній IT індустрії.

Одним з основних факторів ефективності ЕПЗ є інформаційна взаємодія між компонентами системи. Чим швидше і достовірніше надається інформація в процесі управління ПЗ тим більш злагоджено буде функціонувати екосистема.

Аналіз публікацій

Дослідження та побудова моделей ЕПЗ на даний час є актуальним питанням.

Поняття та визначення «екосистема програмного забезпечення» та супутні терміни активно використовуються виробниками і дослідниками ПЗ. Огляд web-сайтів провідних виробників ПЗ показує, що більшість з них застосовують поняття «екосистема ПЗ» [1, 2], позначаючи цим системи, що включають розробника, його ПЗ та партнерів. Наукові дослідження, що використовують поняття екосистем, на сьогоднішній день представлені кількома працями [2, 3]. У роботі [3] автори описують типові елементи екосистем та їх контекст, а також екосистеми розглядаються як рівень абстракції над проектами, який може бути описаний шляхом аналізу нижніх рівнів. У науковій роботі [4], присвяченій системам ультравеликого масштабу (Ultra-Large-ScaleSystems), наголошується, що в галузі існує тенденція до використання концепцій екосистем, для позначення соціально-технічних систем ПЗ. В Україні відомі роботи науковців [5–9], які протягом останніх 10 років вивчають та досліджують ЕПЗ. Наприклад, у роботі [5] представлено застосування екологічного підходу до дослідження ПЗ та формуються основні поняття розділу інженерії програмного забезпечення – екології програмного забезпечення.

Основними елементами екосистеми ПЗ є прикладні продукти та послуги, виробники продуктів та послуг, споживачі, зв’язки між ними. Важливим у дослідженні ЕПЗ є створення їх моделей, а також засобів для їх моделювання та дослідження основних задач, пов’язаних з моделюванням ЕПЗ.

Для опису моделей ЕПЗ застосовують наступні засоби моделювання i*, UML, мережі Петрі [8].

I* мова моделювання. Використовується для розуміння та моделювання ЕПЗ [8]. I* дає можливість розробникам отримати потрібну інформацію на ранньому етапі процесу розробки ПЗ.

Мова UML. Мова UML розвивається з 1994 року і є результатом злиття трьох найбільш відомих об’єктно-орієнтованих підходів: методу Буча, OTM, і OOSE. Вона об’єднує велику кількість різних графічних нотацій з метою впорядкування хаотичного набору графічних засобів, які використовуються під час створення ПЗ. Виступає універсальною мовою моделювання [8]. UML модель ЕПЗ можна представити за допомогою діаграми прецедентів. Діаграма прецедентів (Use Case Diagram) є графічним засобом специфікування вимог.

Мережі Петрі. На практиці моделювання систем часто доводиться вирішувати завдання, пов’язані з формалізованим описом і аналізом причинно-наслідкових зв’язків в складних системах, де одночасно паралельно протікає кілька процесів. Мережі Петрі є найпоширенішим в даний час формалізмом, що описує структуру і взаємодію паралельних систем і процесів [8, 10].

Формування мети дослідження

Мета статті полягає в розробці концептуальних основ методів моделювання ЕПЗ.

Основна частина

На сьогодні існує величезна кількість методів моделювання систем [10-17] і вибрати найкращий для опису ЕПЗ є складною задачею. При розробці моделі слід враховувати дві взаємно суперечливих тенденцій: прагнення до повноти опису і прагнення до отримання необхідних результатів максимально простими засобами. Досягнення компромісу ведеться зазвичай шляхом побудови серії моделей, що починаються з гранично простих і висхідних до високої складності (існує відоме правило: починай з простих моделей, а далі ускладнюй). Прості моделі допомагають глибше зрозуміти досліджувану проблему. Ускладнені моделі

використовуються для аналізу впливу різних чинників на результати моделювання. Такий аналіз дозволяє виключати деякі чинники з розгляду.

Виходячи з вищевикладеного можна запропонувати наступний метод моделювання ЕПЗ. Реалізація цього методу полягає у використанні модельно-орієнтованого підходу (MDA).

Одним із головних намірів MDA є відділення логіки предметної області від конкретної технології реалізації, дозволяючи цим двом частинам змінюватись незалежно одна від одної.

Наприклад, компанія Nissan Motor Co., LTD, використовуючи MDA для проектування системи зменшення вихлопу, сертифікованої за California Air Resources Board (CARB) згідно з стандартом Partial Zero Emission Vehicle (PZEV), скоротила час розробки на 50%, зменшила число датчиків і одержала премію від управління з охорони навколишнього середовища (США).

Компанія ABB також застосувала MDA для проектування розробки і перевірки ПЗ системи управління силових електронних блоків управління AC 800PEC для силових перетворювачів. В результаті час і вартість розробки скоротилась, процес розробки став більш ефективним,якість генерованого коду помітно підвищилася.

Отже, дамо загальні визначення основним поняттям MDA. Слід, спочатку, зазначити що серед спільноти, яка вивчає модельно-орієнтований підхід, в силу різних причин, не існує однозначного тлумачення основних термінів: моделі, мета-моделі й інших. Усе ж таки зупинимось на більш компромісних варіантах.

Означення 1. Модель – це спрощення існуючої системи, побудоване у відповідності до певної мети. В рамках даної статті будемо розглядати модель ЕПЗ.

Означення 2. Мета-модель – це модель мови моделювання.

Модель до мета-моделі знаходяться у відношенні відповідності**,** оскільки модель написана на мові, яку мета-модель визначає. Позначимо це відношення через $\mathcal{X}$. Мета-модель за означенням є моделлю, отже, вона написана на мові своєї мета-моделі – так званої мета-мета-моделі. Ланцюжок відношень модель-мета-модель теоретично може бути як завгодно довгим.

Щоб краще зрозуміти призначення різних стандартів в MDA, потрібно розглянути поняття ***рівнів моделювання*.**

OMG використовує чотирьохрівневу архітектуру, яка часто називається, характеризуючи її вміст, архітектурою 3+1. Рівні в цій архітектурі позначаються таким чином: M0, M1, M2, і M3 (див. рис. 1).

В архітектурі 3+1 бачимо троє відношень відповідності ($\mathcal{X}$): M1-M2, M2-M3, M3-M3 та одне відношення відображення (μ): M0-M1.

Наявність трьох рівнів моделювання, кожен з яких знаходиться у відповідності з вищим рівнем (крім M3), дає змогу проводити трансформації моделей на усіх 3-х рівнях.

Розглянемо кожен рівень моделювання більш детально.

M0. ЕПЗ.

Найнижчим рівнем є M0 – рівень досліджуваної нами системи. Модель відображає (μ) цю систему на рівні M1. Модель рівня M1 відповідає ($\mathcal{X}$) своїй мета-моделі, яка знаходиться на рівні M2. А мета-модель, у свою чергу, відповідає мета-мета-моделі на рівні M3. Завершується цей ланцюжок відношень рівнем мета-мета-моделі встановленням відношення відповідності на себе.

Для створення моделі рівня M1, з рівня M0 необхідно надати інформацію про компоненти за зв’язки ЕПЗ. ЕПЗ може бути представлена як сукупність компонентів: навколишнє середовище, людина та ПЗ. У якості зав’язків можна розглядати інформаційну взаємодію між компонентами, яка розділена на чотири категорії: дія, управління, реакція та плив.

Концептуально модель інформаційної взаємодії ЕПЗ буде мати наступний вигляд (рис.2.):

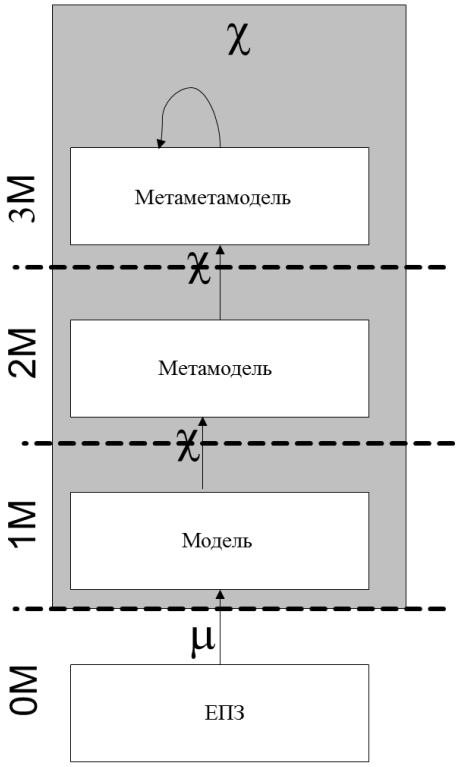

Рис. 1. Архітектура 3+1

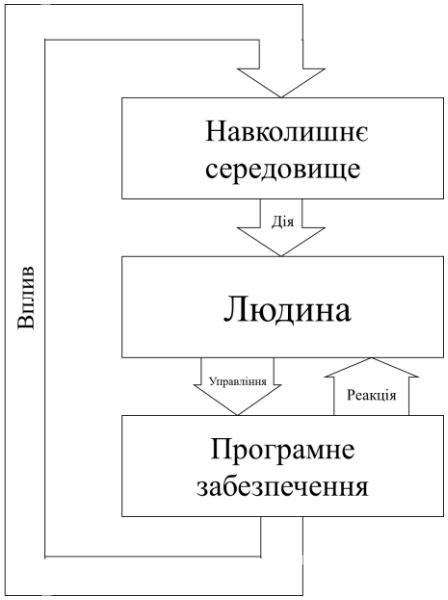

Рис. 2. Концептуальна модель інформаційної взаємодії ЕПЗ

Така модель дає уявлення про взаємодіючі компоненти екосистеми та категорії зав'язків, але не показано їх сутності. Не зрозуміло, яким чином навколишнє середовище діє на людину і як ця дія спонукає до управління ПЗ. Такої інформації недостатньо для побудови адекватної моделі ЕПЗ.

Тому, для створення моделі ЕПЗ рівня 1М спочатку необхідно вирішити задачу формалізації інформаційної дії в ЕПЗ. Основою вирішення цієї задачі буде теорія несилової взаємодії. Оскільки вона в числовій формі досить точно дозволяє ідентифікувати необхідні зміни у відношенні до категорій навколишнього середовища у суб'єктів системи (компонент – людина). Для цього необхідно стандартизувати бачення різноманітних компонентів навколишнього середовища і особливо їх взаємозв'язок з баченням представників зацікавлених сторін та менеджерів того результату, який заданий цілями ЕПЗ. Інструменти інформаційної дії на людину включатимуть в себе способи отримання необхідної інформації. Вирішення цієї задачі дозволить досягти мети – ефективно управляти ПЗ.

Управління інформацією повинне бути таким, щоб діяти на суб'єкти ЕПЗ таким чином, щоб вони оптимально реагували на готовність ПЗ і його частин на відхилення в плані управління ПЗ, на ситуацію в навколишньому середовищі та в середовищі самої екосистеми.

Для управління ПЗ важливим є не тільки факт відхилення його стану від планових змін, але й сама величина відхилення. Будемо розглядати два ключові відхилення, які впливають найбільше на рішення в управлінні ПЗ. Це відхилення у навколишньому середовищі та відхилення в стані ПЗ. Задамо функції, що визначають:

1. Дію на суб'єктів екосистеми ПЗ, яка обумовлена відхиленням

$\eta\left(V_e(t), V_e^*(t), G_e(t), G_e^*(t), S, t\right)$, де

$V_e(t)$ – план управління ПЗ;

$V_e^*(t)$ – фактичне управління ПЗ;

$G_e(t)$ – плановий стан ПЗ в момент часу *t*;

$G_e^*(t)$ – фактичний стан ПЗ в момент часу *t*.

$\eta\left(V_e(t), V_e^*(t), G_e(t), G_e^*(t), S, t\right)$ – інформаційна дія на суб'єкти ЕПЗ, що визначається часом, зацікавленою стороною, величиною відхилення в навколишньому середовищі та в стані ПЗ в момент часу *t*; *t* – момент часу; *S* – зацікавлена сторона.

2. Реакція зацікавленої сторони, яка є результатом відхилення

$$d\left(\eta\left(V_e(t), V_e^*(t), G_e(t), G_e^*(t), S, t\right), t_0\right),$$

де $d\left(\eta\left(V_e(t), V_e^*(t), G_e(t), G_e^*(t), S, t\right), t_0\right)$ – можливі реакції зацікавленої сторони *S*, що визначаються інформаційним впливом $\eta\left(V_e(t), V_e^*(t), G_e(t), G_e^*(t), S, t\right)$ в момент часу t_0; t_0 – момент часу, коли формується реакція зацікавленої сторони.

Визначення оптимальної інформаційної дії, яка повинна формуватися під час взаємодії компонентів ПЗ – людина було б зробити не складно, якщо це відносилось лише до одного фактору взаємодії. Для 4-х наведених впливів (навколишнє середовище, стан, зацікавлена сторона і час), в яких, скажімо, виділяється до 20 різноманітних значень (насправді їх набагато більше) буде отримано 20^4=160000 варіантів. Що неможливо описати в будь якій системі. Тому необхідне застосування спеціальних методів по визначенню найбільш інформативних впливів (дій) на суб'єкти ЕПЗ. І по окремим діям на них визначати оптимальну реакцію в тій чи іншій ситуації. Для цього спочатку сформуємо математичну модель декомпозиції інформаційних дій на суб'єктів екосистеми ПЗ.

Означення 3. Інформаційна дія на суб'єкти екосистеми – цілеорієнтоване надання суб'єктам ЕПЗ інформації про зміни в навколишньому середовищі.

Означення 4. Суб'єкти ЕПЗ – зацікавлені сторони управління ПЗ, особи, що приймають рішення, виконавці робіт та контрагенти.

Означення 5. Інформаційні впливи – екологічні звіти, технічне завдання на розробку ПЗ, відклики користувачів, звіти про роботу ПЗ, інше.

Для того, щоб визначити найбільш вагому дію на зацікавлених сторін, залежну від наведених чотирьох чинників, розглянемо суперпозицію функцій:

$$\eta\left(V_e(t), V_e^*(t), G_e(t), G_e^*(t), S, t\right) = f\left(\eta^1\left(V_e(t), V_e^*(t)\right), \eta^2\left(G_e(t), G_e^*(t)\right), \eta^3(S), \eta^4(t)\right),$$

де $\eta^1\left(V_e(t), V_e^*(t)\right)$ – інформаційна дія на суб'єктів, що визначається величиною відхилення в навколишньому середовищі;

$\eta^2\left(G_e(t), G_e^*(t)\right)$ – інформаційна дія на суб'єктів екосистеми ПЗ, що визначається величиною відхилення в стані програмного забезпечення;

$\eta^3(S)$ – інформаційна дія суб'єкту екосистеми ПЗ *S*;

$\eta^4(t)$ – інформаційна дія часу *t*.

Запропонована в теорії несилової взаємодії модель дає можливість отримати по спостереженню за проявами окремих об'єктів, чи по окремим експертним оцінкам, їх спільну дію на рішення по управлінню ПЗ. Нехай методами експертної оцінки, чи з застосуванням статистичних методів отримані суб'єктивні імовірності, чи оцінки вибору (в межах [0,1]) тих чи інших впливів на зацікавлених сторін в залежності:

1. Від відхилення в навколишньому середовищі:

$$p_{ej}^1 = f_1\left(\eta^1\left(V_e(t), V_e^*(t)\right) = F_j\right), \quad (1)$$

де $p_{ej}^1 = f_1\left(\eta^1\left(V_e(t), V_e^*(t)\right) = F_j\right)$ – оцінка того, що при заданих плановому і фактичному показниках в навколишньому середовищі необхідна інформаційна дія F_j; F_j – інформаційна дія на зацікавлених сторін (інформація про необхідність покращення, аналізу чи пристосування до навколишнього середовища за допомогою ПЗ або ліквідація шкідливих наслідків використання ПЗ).

2. Від стану компонента ЕПЗ:

$$p_{ej}^{2} = f_2\left(\eta^2\left(G_e(t), G_e^*(t)\right) = F_j\right), \quad (2)$$

де $p_{ej}^{2} = f_2\left(\eta^2\left(G_e(t), G_e^*(t)\right) = F_j\right)$ – оцінка того, що при заданих плановому і фактичному показниках зміни стану ПЗ необхідна інформаційна дія F_j.

3. Від зацікавленої сторони:

$$p_{ej}^{3} = f_3\left(\eta^3(S) = F_j\right), \quad (3)$$

де $p_{ej}^{3} = f_3\left(\eta^3(S) = F_j\right)$ – оцінка того, що зацікавлена сторона екосистеми ПЗ*S* буде вимагати інформаційну дію F_j.

4. Від того, на якій стадії знаходиться ПЗ:

$$p_{ej}^{4} = f_4\left(\eta^4(t_e) = F_j\right), \quad (4)$$

де $p_{ej}^{4} = f_4\left(\eta^4(t_e) = F_j\right)$ – імовірність того, що на фазі життєвого циклу ПЗ t_e буде інформаційна дія F_j ; t_e – фаза ПЗ.

По суті цими функціями задана випадкова поведінка системи інформування зацікавлених сторін *S,* відносно ПЗ, де з ймовірностями, які відповідають оцінкам $p_{ej}^{k}(X), k = \overline{1,4}$ реалізується вибір виду дії (впливу) на зацікавлені сторони F_j.

М1. Модель

Після отримання декомпозиції інформаційної дії в ЕПЗ можна перейти до побудови моделі.

По принципу чорної скриньки у якості вхідної інформації будемо мати компоненти ЕПЗ та інформаційні взаємозв'язки між ними, а вихідною інформацією є модель ЕПЗ (рис.3).

Рис. 3. Чорна скринька MDA

Побудова моделі ведеться певною мовою моделювання. Запропонованою мовою моделювання від OMG є Unified Modeling Language (UML), яка є мовою загального призначення. Але є можливість розширення під свою предметну область.

Візуальне моделювання в UML можна представити як деякий процес покрокового спуску від найбільш загальної і абстрактної концептуальної моделі вихідної системи до логічної, а потім, і до фізичної моделі відповідної програмної системи. Для досягнення цих цілей спочатку будується модель у формі так званої діаграми варіантів використання (use case diagram), що описує функціональне призначення системи або, іншими словами, те, що система буде робити в процесі свого функціонування. Діаграма варіантів використання є вихідним концептуальним поданням або концептуальною моделлю системи в процесі її проектування й розробки.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді множини сутностей та акторів, що взаємодіють із системою за допомогою так званих варіантів використання. При цьому актором (actor), або діючою особою, називають будь-яку сутність, що взаємодіє із системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, що може бути джерелом впливу на систему, що

моделюється, так, як визначить сам розробник. У свою чергу, варіанти використання (use case) слугують для опису сервісів, які система надає акторові.

Наведена вище діаграма варіантів використання, у свою чергу, може бути деталізована далі з метою більш глибокого уточнення поставлених до системи вимог і конкретизації деталей її наступної реалізації.

Побудуємо модель ЕПЗ з використанням інформаційних впливів на її елементи. Як було описано вище ЕПЗ складається з трьох сутностей Людина – ПЗ – Зовнішнє середовище.

М2. Мета-модель

Існує дві причини чому побудова мета-моделей важлива для модельно-орієнтованого підходу. По-перше, потрібний механізм для визначення недвозначних (формалізованих) мов моделювання. У модельно-орієнтованому підході такий механізм здійснюється через побудову мета-моделей (метамоделювання) [6].

По-друге, ключові для модельно-орієнтованого підходу механізми трансформації моделей використовують мета-моделі. Інструменти, що реалізують механізми перетворень, уміщують опис, як модель на вхідній мові може бути перетворено в моделі на вихідній мові. Ці правила використовують мета-моделі початкових і цільових мов для визначення перетворення.

Побудова мета-моделі здійснюється на основі аналізу атрибутів, функцій, структури та інших властивостей моделі. У залежності від виду моделей, можна виділити два підходи до створення мета-моделей:

- алгебраїчний (або безпосередній), що здійснюється на підставі аналізу моделей, як частини дійсності;
- логічний(або опосередкований), що здійснюється на підставі аналізу інформаційних потоків в моделі.

У першому випадку входом методу для побудови метамоделі є атрибути, структура, правила, операції та інші елементи моделі, що надані у формі фізичних величин, випадкових процесів, математичних функцій, фізичних законів та тому подібне. У другому випадку метамоделі будуються на основі моделі знань (зокрема, семантичної мережі, продукційної, фреймової, логічної моделі тощо)[13].

Зазначимо, що можливим є існування змішаних випадків, коли входом методу для побудови метамоделі є водночас як алгебраїчні так і логічні моделі.

У даній роботі розглядається метод розробки метамоделей на основі логічних моделей, що має важливе окреме значення у рамках сучасних ІТ.

Сутність підходу полягає у зануренні логічної моделі у деяку систему, що містить інформаційні зв'язки між об'єктами, та розглядається як метамоделі.

Таким чином, логічна модель розглядається як метамоделі(М2).

М3. Мета-мета-модель

Рівень мета-мета-моделі утворює початкову основу для усіх мета-модельних представлень. Головне призначення цього рівня полягає в тому, щоб визначити мову для специфікації метамоделі. Мета-мета-модель визначає модель мови UML на самому високому рівні абстракції і являється найбільш компактним її описом. З іншого боку, мета-мета-модель може включати декілька мета-моделей, чим досягається потенціальна гнучкість включення додаткових понять. Прикладами понять цього рівня служать мета клас, метаатрибут.

Слід зазначити, що семантика мета-мета-моделі не входить в опис мови UML. З одного боку, це робить мову UML простішою для вивчення, оскільки не потрібно знання теорії формальних мов і формальної логіки. З іншого боку, наявність мета-мета-моделі надає мові UML статус науковості, який потрібний йому для того, щоб бути непротирічною формальною мовою. Якщо ці особливості можуть представлятися мало цікавими для багатьох програмістів, то розробники інструментальних засобів ніяк не можуть їх ігнорувать.

В рамках даної статті ми використали мову моделювання MOF (Meta Object Facility). MOF – стандарт OMG, який визначає мову, яка слугує для визначення мов моделювання. MOF знаходиться на найвищому рівні (М3) моделі чотирьох рівневої архітектури OMG. MOF є мета-мета-моделлю для всіх мета-моделей. MOF використовується не лише для побудови мов моделювання, але й для побудови інструментів для побудови мов моделювання. Тому MOF забезпечений додатковою функціональністю.MOF, також

використовується для визначення потокового, або файлового формату обміну для моделей рівня М1. Кожна мова моделювання визначена у відповідності з мета-моделлю описаної в MOF. MOF визначає стандартний шлях проводити обмін для мов моделювання. Цей формат обміну базується на XML, і має назву XMI (XML Metadata Interchange). Оскільки MOF є мета-моделлю для себе, XMI може використовуватися також для генерації стандартного формату обміну для мета-моделей.

Розглянемо на прикладі використання методу MDA для приладобудівного підприємства.

Для побудови моделі рівня М1 зобразимо відношення замовника та користувача ПЗ. Вони взаємодіють через PRP-system. Користувач отримує необхідну інформацію з PRP-system, а замовник під впливом навколишнього середовища ініціює її розробку (рис. 4.).

Як видно на рис.5. з'явився суб'єкт Розробник, який виконує розробку PRP-system, а Кінцевий користувач в свою чергу надає вимоги до розроблюваної системи, Замовник виконує задачу управління розробкою.

Модель М3 ще більш деталізує і розкриває сутності взаємовідношень ЕПЗ та процеси розробки PRP-system.

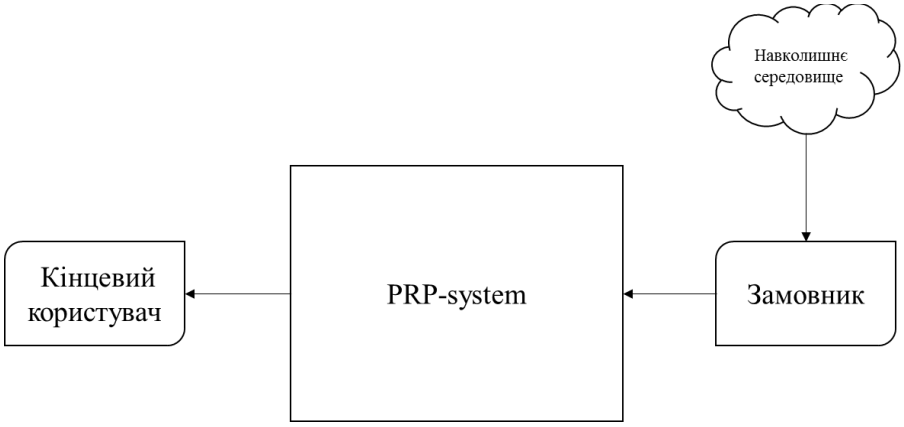

Рис. 4. Модель М1 ЕПЗ

Далі продовжуємо деталізувати відношення між компонентами системи. Додаються нові суб'єкти та сутності системи (рис. 5).

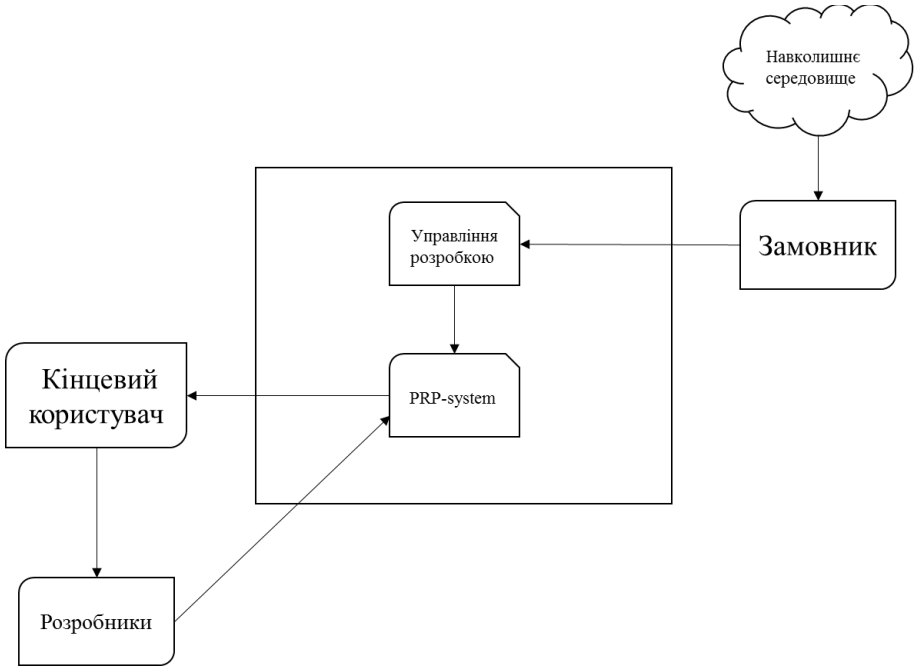

Рис. 5. Модель М2 ЕПЗ

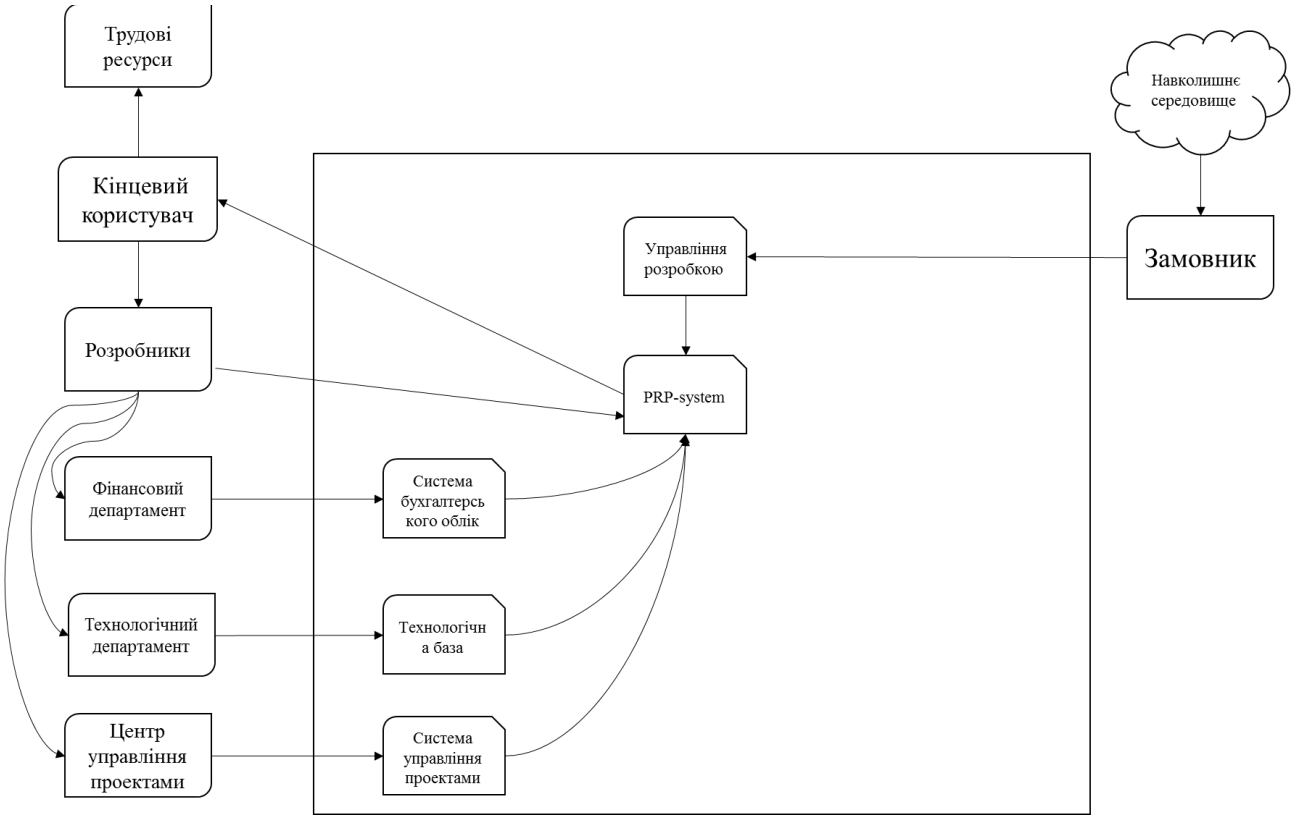

Рис. 6. Модель М3 ЕПЗ

Висновки

В статті запропоновано використання модельно-орієнтованого підходу MDA до моделювання ЕПЗ.

Такий підхід має ряд переваг: MDA надає загальне середовище для розробки ПЗ, що сприяє взаємодії груп розробників в процесі аналізу даних та перевірки системи; інженери можуть знайти та виправити помилки на ранніх стадіях проектування системи, коли час та фінансові наслідки зміни системи зводяться до мінімуму; MDA сприяє повторному використанню моделі для покращення системи та створення похідних систем з розширеними можливостями.

Ще однією з найважливіших задач в моделюванні ЕПЗ є задача оцінки стану ПЗ, його оточення та навколишнього середовища. Ця задача невідривна від процесів управління інформацією, яка в свою чергу використовується для прийняття раціональних рішень на всіх стадіях функціонування ЕПЗ. І MDA може забезпечити створення ЕПЗ на основі даних про інформаційну взаємодію між компонентами системи.

Отже, можна зробити висновки, що використання модельно-орієнтованого підходу найкраще підходить для моделювання ЕПЗ. Моделювання ЕПЗ потребує подальших досліджень. Для побудови адекватної моделі ЕПЗ необхідно спрогнозувати вплив ПЗ на навколишнє середовище і навпаки, а також розробити математичний апарат для моделювання ЕПЗ.

Список використаної літератури

1. Jansen, S. Understanding Software Ecosystems: A Strategic Modeling Approach [Text] / S. Jansen, J. Bosch // Proceedings or the Workshop or Software Ecosystems. – 2011. – P. 65 – 76.

2. Duinkerken, W. Transaction Cost Economics is Software Ecosystems [Text] / W. Duinkerken // Some empirical evidence. – 20 April 2009. – P. 22 – 37.

3. Dyba, T. Empirical studies or agile software development: A systematic review [Text] / T. Dyba, T. Dingsoyr // Informationand Software Technology. – 2008. – Vol. 50, № 9 –10. – P. 833 – 859. doi:10.1016/j.infsof.2008.01.006

4. Messershmitt, D. Ultra-Large-ScaleSystems [Text] / D. Messershmitt, C. Szyperski // UnderstandinganIndispensableTechnologyandIndustry. — London: MIT press, 2003. — 233 p.

5. Сидоров Н. А. Экология программного обеспечения [Текст] / Н. А. Сидоров // Інженерія програмного забезпечення. – 2010. – № 1. – С. 53 – 61.

6. Хоменко В. А. Екосистеми програмного забезпечення [Текст] / В. А. Хоменко // Вісник Національного технічного університету «ХПІ». – 2011. – № 23. – С. 114 – 118.

7. Луцький, М. Підтримка придатності та супроводження експлуатації програмного забезпечення авіаційної техніки [Текст] / М. Луцький, М. Сидоров, Ю. Рябокінь // Проблеми програмування. — 2010. — № 23. — С. 229–239.

8. Sidorov N. A. Software Ecosystems Modeling [Text] / N. A. Sidorov, O. O. Grinenko // Інженерія програмного забезпечення. – 2013. – № 2(14). – P. 38 – 48.

9. Гріненко О. О. Екосистема програмного забезпечення як система систем [Текст] / О. О. Гріненко // Наукоємні технології. – 2014. – № 3 (23). – С. 280 – 283.

10. Томашевський В. М. Моделювання систем [Текст]: підручник / В. М. Томашевський. – Київ, 2005. – 349 с.

11. Boucharas, V. Formalizing software ecosystem modeling [Text] / V. Boucharas, S. Jansen, S. Brinkkemper // Proceedings or the 1st international workshop or Open component ecosystems. – 2004. – P. 41–50. doi:10.1145/1595800.1595807

12. Jansen, S. A sense or community: A research agenda or software ecosystems [Text] / S. Jansen, A. Finkelstein, S. Brinkkemper // 31th International Conference or Software Engineering, Newand Emerging Research Track. – 2009. – P. 187 – 190. doi:10.1109/icse-companion.2009.5070978

13. Самарский А.А., Михайлов А.П. Математическое моделирование: Идеи. Методы. Примеры. – М.: Наука, 1997. – 320 с.

14. Аристов А.О. Теория квазиклеточных сетей: научная монография. – М: МИСиС, 2014. – 188 с.

15. Емельянов А.А., Власов Е.А., Дума Р.В. Имитационное моделирование экономических процессов: Учеб. пособие / Под ред. А.А. Емельянова. М.: Финансы и статистика, 2002. – 368 с.

16. Карпов Ю.Г. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5. – СПб.: БХВ-Петербург, 2005. – 400 с.

17. Кобелев Н.Б. Основы имитационного моделирования сложных экономических систем: Учеб. пособие. – М.: Дело, 2003. – 336 с.

Автори статті

Гріненко Олена Олександрівна – старший викладач кафедри інженерії програмного забезпечення, Навчально-науковий інститут комп'ютерних інформаційних технологій, Національний авіаційний університет, Київ, Україна. Тел. +380979125920. E-mail: Elena.Grinenko@livenau.net.

Гріненко Сергій Анатолійович – асистент кафедри інженерії програмного забезпечення, Навчально-науковий інститут комп'ютерних інформаційних технологій, Національний авіаційний університет, Київ, Україна. Тел. +380979710776. E-mail: SergGrinenko@gmail.com.

Authors of the article

Grinenko Olena Oleksandrivna – Senior Lecturer of Software Engineering Department, Scientific Research Institute of Computer Information Technologies, National Aviation University, Kyiv, Ukraine. Tel.: +380979125920. E-mail: Elena.Grinenko@livenau.net.

Grinenko Sergiy Anatiliyovich – assistant of Software Engineering Department, Scientific Research Institute of Computer Information Technologies, National Aviation University, Kyiv, Ukraine. Tel.: +380979125920. E-mail: SergGrinenko@gmail.com.

Дата надходження в редакцію: 11.01.2017 р. Рецензент: д.т.н., проф. В.В. Вишнівський