https://en.wikipedia.org/wiki/ZigBee
24. Wordpress [Електронний ресурс]: Wireless Energy Transfer By Using Electromagnetic Induction / Satu Tanggapan – 2010 – URL: https://muhammadaltway.wordpress.com/2010/10/18/wireless-energy-transfer-by-using-electromagnetic-induction/
25. Steuer, J. (1992) Defining Virtual Reality: Dimensions Determining Telepresence. J. Commun., 42(4), pp. 73–93.
26. Blender 3d [Електронний ресурс]: Моделирование низкополигонального динозавра Артём Гильванов – 2014 – URL: https://blender3d.com.ua/modelirovaniye-nizkopoligonalnogo-dinozavra-v-blender/
27. Huang, F.-C. , Chen, K. , and Wetzstein, G. (2015). The Light Field Stereoscope: Immersive Computer Graphics Via Factored Near-Eye Light Field Displays With Focus Cues, ACM Trans. Graphics, 34(4), pp. 60:1–60:12.
28. Bochenek, G. M. , and Ragusa, J. M. (2004). Improving Integrated Project Team Interaction Through Virtual (3D) Collaboration, Eng. Manage. J., 16(2), pp. 3–12.

Рецензент статті
д.т.н., проф. Бушуєв С.Д.

Стаття рекомендована до публікації 18.09.2018 р.

УДК 005.8

В.А. Ситник, П.О. Тесленко, Д.І Бедрій, О.І. Шерстюк

УПРАВЛІННЯ ПРОТОТИПУВАННЯМ ТА РИЗИКАМИ ІТ-ПРОЕКТІВ З ВІДКРИТИМ КОДОМ

Запропонована методика аналогового моделювання ІТ-проектів. Методика теоретично обгрунтована та дозволяє на ранніх етапах прогнозувати час та вартість майбутнього проекту. Визначено критерії обгрунтування вибору ІТ-проекту з відкритим кодом, створено життєвий цикл вибраного классу проектів. Сформульовано алгоритм застосування теорії подібності для визначення часу прототипування проекту. Визначено ризики відкриття коду, шляхи їх розв'язання та обсяги планування. Методика може бути застосована при створенні нових ІТ-проектів різних класів. Дж. 25.

Ключові слова: управління ІТ-проектами, теорія подібності, системний аналіз, критеріальне рівняння, життєвий цикл, аналіз розмірностей, управления ризиками ІТ-проектов, об'єм планування.

JEL O220

Постановка проблеми в загальному вигляді, її зв'язок з важливими науковими та практичними задачами. Управління проектами розробки програмного забезпечення з відкритим кодом на даний момент стало великим полем як наукової, так і практичної діяльності. В галузі створення програмного забезпечення задіяно понад два мільйони інженерів по всьому світу, 20% яких становлять менеджери. В Україні за оцінками асоціації "ІТ–України" кількість інженерів і менеджерів на 2017 рік становить 175 тис. осіб, до того ж кожен рік навчальні заклади випускають понад 13 тисяч фахівців. До цього, приблизно 38% всіх розробників беруть участь у ІТ-проектах створення програмного забезпечення з відкритим вихідним кодом.

У зв'язку з такою популярністю програмних проектів з відкритим кодом і їх значенням для індустрії розробки програмного забезпечення виникає необхідність володіти знаннями з їх управління. Однак останні дослідження були спрямовані лише на опис моделі і на дослідження конкретних випадків її

використання. Внаслідок цього питання вивчення життєвого циклу і системи управління, які забезпечують основу знань для ефективного управління проектами, залишаються відкритими. Невивченість існуючих методів виконання процесів управління, особливо управління часом і вартістю, також заважає ефективному управлінню та успішному завершенню проектів в рамках заданих обмежень часу, вартості та якості.

Все це призводить до необхідності дослідження способів підвищення ефективності управління IT-проектами з відкритим кодом шляхом розробки нових моделей оцінки часу, нової методики управління життєвим циклом проекту.

Аналіз досліджень і публікацій та виділення невирішених раніше частин загальної проблеми. Аналіз останніх досліджень в галузі управління програмними проектами показав, що лише ефективне управління ними може гарантувати їх успішне завершення в рамках обмежень часу, вартості та якості. За даними дослідницької групи Standish Group [1], з 50000 відстежених програмних проектів через неефективне управління 18% завершилися невдало, 53% потребували додаткових витрат часу і тільки 29% успішно завершилися. За іншими даними [2], більше 50% проектів виявляться невдалими. За результатами [1], в середньому бюджет проектів перевищується на 189%, а витрачений час на 222% перевищує оцінене. При цьому реалізується в середньому всього 69% заявленої в специфікації функціональності.

Описана вище ситуація пояснюється тим, що "... розробка програм є концептуально складним заняттям" [3], труднощі створення програмних систем виникають через їх "нескорочувану сутність", до властивостей якої відносяться складність і мінливість проекту [4].

Однак, останнім часом з'явилося багато нових моделей управління програмними проектами, які, не дивлячись на неможливість революційного прориву в галузі, дозволяють підвищити ефективність управління. Одним з таких способів стала модель розробки програмного забезпечення з відкритим кодом. Програмні проекти з відкритим кодом відомі на практиці як успішні проекти, що завершуються вчасно, з мінімальними витратами, і в яких виготовляється якісний продукт. За даними роботи [5], приблизно 27% всіх фірм–розробників програмного забезпечення виконують такі проекти. Крім того, існує більше 170000 некомерційних відкритих проектів, в яких беруть участь більше мільйона чоловік.

Варто відзначити [6], що останні дослідження були спрямовані лише на опис моделі і на дослідження конкретних випадків її використання. Водночас, питання вивчення життєвого циклу і системи управління як основи знань для ефективного управління проектами, залишаються відкритими. Невизначеність методик управління часом і вартістю, ризики віднесення проекту до певного класу, заважає ефективному управлінню та успішному завершенню проектів в рамках заданих обмежень часу, вартості та якості.

Все це спонукає до вивчення способів підвищення ефективності управління програмними проектами з відкритим кодом шляхом розробки нових моделей оцінки часу, обсягів планування, управління ризиками вибору певної моделі життєвого циклу IT-проекту [7, 8, 9].

З огляду на дослідження Evans Data Corporation [10], які показують, що понад 35% розробників по всьому світу задіяні в проектах з відкритим кодом, і ця цифра зростає на 2-5% на рік, а також з огляду на величезну кількість і успішність як комерційних, так і некомерційних проектів, потрібно визнати важливість відкритої моделі розробки програмного забезпечення та необхідність досліджень як самої моделі, так і способів підвищення ефективності управління програмними проектами, що виконуються в цій моделі.

Однак, необхідно звернути увагу на порівняно невелику частку комерційних проектів (15%), що пояснюється багатьма аналітиками відсутністю теоретичного обґрунтування основних принципів виконання таких проектів. Як зазначається в [11], "найбільша проблема індустрії відкритих початкових кодів, що перешкоджає її подальшому розвитку, сьогодні полягає в тому, що навіть прихильники відкритого коду в більшості своїй ще не розуміють, що відкрита модель є дійсно самостійною. Ніхто з прихильників відкритого коду не розуміє, який інструмент виявився у них в руках. А з цього вже витікає фізична неможливість використовувати його “ефективно” [12], що підтверджується і численними побоюваннями, висловлюваними менеджерами і керівництвом компаній. Так, наприклад, в Sim Microsystems, незважаючи на позитивний досвід відкриття вихідного коду OpenOffice та шестирічний досвід його подальшої розробки у відкритій моделі, більше двох років ухвалювали рішення про відкриття вихідного коду Java і переведення проекту Java на відкриту модель [13-15].

Мета статті. Розробка методики управління створенням програмного забезпечення з відкритим вихідним кодом є важливою і своєчасною. Розв’язанням такого завдання є формальний опис моделі відкритих початкових кодів, складових її фаз і дій, розробка моделі життєвого циклу, часу прототипування і системи управління ризиками прийняття рішення про приналежність проекту до певного класу. Все це надасть необхідну основу знань для організацій, які планують, або розробляють програмні проекти вибраного класу.

Виклад основного матеріалу дослідження.

Розробка моделі життєвого циклу IT проектів. Опис моделі життєвого циклу і її складових, необхідний для успішного управління проектами, складається з:

- схематичного опису для визначення фаз і їх послідовності;
- діаграм розподілу робіт за фазами проекту;
- таблиць ідентифікації завдань і дій, які виконуються в життєвому циклі, які потрібні для подальшого розподілу ресурсів проекту [16].

Детальний формальний опис в такому вигляді визначає методологію управління проектами, що повинна виконувати всі процеси життєвого циклу відповідно до стандартів ISO / TEC 12207 [17] і ДСТУ 3918–1999 [18].

Модель життєвого циклу програмних проектів визначається сукупністю загальних і специфічних ознак.

Загальні ознаки моделі визначаються як:

- множина фаз виконання проекту;
- зміст технічних робіт по кожній фазі;
- розподіл інженерних і технічних ресурсів за фазами;
- послідовність виконання визначених фаз;
- перехідні процеси між фазами.

Специфічні ознаки, наприклад відкритої моделі, наступні:

- середовище виконання відкритого проекту значно ширше, ніж у будь–яких інших програмних проектів;
- нормальною практикою виконання проекту є постійне залучення сторонніх ресурсів, як матеріальних, так і трудових;
- множина виконавців проекту не є ні фіксованою, ні чітко визначеною.

Визначення фаз відкритої моделі і їх послідовності здійснюється шляхом аналізу ходу виконання вже завершених проектів.

Для аналізу були обрані найбільш характерні проекти, що представляють всі типи відкритих проектів:

Аналіз проводився за допомогою:

- хронологічних даних про проекти з електронної енциклопедії Wikipedia;
- інформації зі списків розсилки та веб–сайтів проектів;
- планів випуску та анонсів.

Результати аналізу показують, що за основу можна взяти хронологічні дані про управлінські дії в проекті KDE. Дані про інші проекти показують багато в чому аналогічну картину.

Інструкцією слугувала коротка характеристика дії, що класифікує його як етап (початок, або завершення) певної фази життєвого циклу. Анотація виконувалася з метою визначення фаз життєвого циклу.

Кожна фаза розробки, виходячи з даних системи управління версіями і планів випуску нових версій, розподіляється на три основних етапи. Перший етап – виконання одночасного прототипування, кодування і документування з метою випуску нестабільної, але в повному обсязі функціональної "α" версії продукту. Наступний етап, після випуску "α" версії, включає в себе одночасне кодування, документування та модульне тестування з метою випуску більш стабільної і повнофункціональної "β" версії продукту. І, нарешті, в останній етап входить інтеграція і кваліфікаційне тестування продукту з метою випуску стабільної і повнофункціональної наступної версії продукту (з проміжними опційними випусками версій вигляду "release candidate"). За результатами проведеного вище аналізу і змістом проектної інформації, виконано схематичне відображення фаз життєвого циклу відкритої моделі і визначено зміст процесів, що виконуються в кожній з фаз.

Візуальний аналіз моделі показує, що вона має процеси, властиві каскадній та спіральній моделям. Як і в спіральній моделі, до неї включені процеси аналізу та управління ризиками і підтримки менеджменту. Передбачена розробка програмного продукту при використанні методу прототипування або швидкої розробки додатків за допомогою застосування мов програмування і засобів розробки четвертого і вище покоління. Одночасно, кожен цикл моделі відображає базову концепцію, яка полягає в тому, що цикл являє собою набір операцій, якому відповідає така ж кількість стадій, що і в каскадній моделі.

З іншого боку, відкрита модель вносить нову концепцію обробки і управління зовнішніми змінами. Вони оголошуються невід'ємною частиною процесу виконання проекту, і їх завданнями стають, в першу чергу, вирішення проблеми здійсненності проекту, а, по-друге, розв’язання проблеми ризиків, пов'язаних з персоналом, та складністю проекту. Також цикли визначення проекту і огляду вимог суміщені у відкритій моделі з метою спрощення її використання і прискорення процесу виконання проекту. Модель ділиться на квадранти, кожному з яких відповідає мета і кожен з яких складається з множини фаз для досягнення цієї мети.

Цілі, що відповідають квадрантам моделі, можуть бути визначені як:

– визначення цілей, альтернатив і обмежень. Тут визначаються цілі всього проекту, визначається робоча характеристика, перелік функцій, що виконуються, вирішальні чинники досягнення успіху, програмно-апаратний інтерфейс. Визначаються альтернативні способи реалізації продукту (частин продукту): повторне використання, розробка, кооперація, придбання. Визначаються обмеження, що накладаються на використання альтернативних варіантів: час, інтерфейс, обмеження на продукт, зовнішнє середовище, тощо [19];

– оцінка альтернатив, ідентифікація і розв’язання ризиків. Виконується апробація (прототипування) і оцінка альтернатив, визначених у попередньому

квадранті, оцінюються і розв’язуються ризики, приймається рішення про продовження (або припинення) робіт над проектом [20];

– розробка продукту. Виконується, власне, розробка коду, тестування, документування, складання і випуск продукту. Тут продукт потрапляє до замовника, його (продукту) життєвий цикл трансформується в цикл з первинною метою поліпшення і доробки;

– планування наступної фази. Продукт, що надійшов до замовника, в своєму новому життєвому циклі проходить фази впровадження, супроводу та експлуатації, тоді як в життєвому циклі проекту проводиться підготовка до випуску наступної версії продукту. На підставі даних про впровадження, супровід та експлуатацію випущеного продукту, а також на підставі експертних оцінок і оглядів продукту у процесі розробки виконується планування модифікації процесу і продукту, що впроваджуються в наступному циклі.

Процеси управління проектом. У відповідності до [21] процеси управління IT-проектами діляться на дві категорії: процеси, орієнтовані на продукт, зосереджені на визначенні та створенні проекту, і процеси управління проектами, що описують і впорядковують роботи в проекті. Процеси, орієнтовані на продукт були визначені вище в формальному описі життєвого циклу. Процеси ж управління проектом повинні бути описані окремо, але за умови часового накладення і взаємозв'язку з процесами, орієнтованими на продукт.

При розробці програмного забезпечення, в тому числі з відкритим кодом, виділяються [22] кілька основних процесів управління, що відповідають 5-ти групам процесів управління, визначених в [21]. Решта процесів, що відносяться до продукту, і процеси управління ресурсами, які виконуються протягом всього життєвого циклу, розглядати на діаграмах недоцільно.

1. Визначення цілі і сфери дії програмного проекту. Цей процес втілює процеси ініціалізації і планування. При виконанні проекту визначається одна або кілька цілей, досягнення яких закладається в плані менеджменту (SPMP, Software Project Management Plan) за умови обмежень в ресурсах і якості. План менеджменту і технічне завдання проекту визначають методологічну сторону середовища проекту, а технічна сторона визначається життєвим циклом[23].

2. Відбір і управління командою розробників. Відбір команди – це найбільш важливий процес з групи процесів виконання відкритого проекту. Зазвичай селекція команди, формування колективу розробників неабияк впливає на інші дії життєвого циклу розробки програмного забезпечення. У відкритій моделі важливим є попереднє планування і формальний опис процесу приєднання сторонніх розробників. Виконання цього процесу і менеджмент стороннього персоналу є невід'ємна частина фаз розробки IT-проекту з відкритим кодом [24].

3. Забезпечення надійності. Цей процес відноситься до великої групи процесів здійснення контролю за проектом. Процесу приділяється особлива увагу з огляду на його важливість для кінцевого користувача, і він безпосередньо впливає на успішність проекту. Процес створення надійного програмного забезпечення повинен включати в себе, у відповідності до [22], прогнозування, запобігання і усунення помилок, а також реалізацію методик забезпечення стійкості до відмов.

Аналогове моделювання часу прототипування в програмних проектах. Оскільки програмні продукти, що створюються, відносяться до різних "класів", узагальнення досвіду їх створення можливо тільки всередині одного класу, або всередині ще більш вузьких груп.

Найбільш досконалим і перевіреним на практиці апаратом аналогового моделювання є теорія подібності, основні положення якої були розроблені ще на початку минулого століття і найбільш повно узагальнені А.А. Гухманом [25].

Основна ідея теорії подібності полягає в тому, що в межах певного класу явищ, або процесів, виділяються групи, в яких можливо узагальнення даних одиничного досвіду. Узагальнити досвід виконання ІТ-проекту – означає дати можливість оцінити трудомісткість, вартість і тривалість виконання аналогічного проекту, тобто створити аналогічну (подібну) систему. Це можливо, якщо співвідношення між безрозмірними ознаками (критеріями подібності), що характеризують властивості системи, представити у вигляді апроксимуючої степеневої функції, що найбільш часто використовується:

$$\pi = A \cdot \pi_1^a \cdot \pi_2^b \cdot \ldots \cdot \pi_n^z,$$

де $\pi, \pi_1, \pi_2, \ldots, \pi_n$ – безрозмірні величини, критерії подібності;

A – коефіцієнт пропорціональності між визначальними і такими, що визначаються, критеріями;

$a, b, \ldots, z$ – показники ступеня за критеріїв.

Оскільки проект виконується в часі і не має просторових характеристик, то з умов подібності початкових і граничних умов такими, що застосовуються, залишаються тільки умови подібності початкових умов.

Такі початкові умови для програмних систем, як:

- процес і методи розробки продукту;
- мова програмування;
- платформа для цільової системи;
- інструменти, що використовуються, повинні бути подібні для систем однієї групи.

Отже, умовами подібності програмних проектів (процесів їх виконання) є:

- належність до одного класу;
- подібність життєвих циклів розробки;
- подібність констант зовнішнього середовища виконання розробки;
- подібність початкових умов входження в життєвий цикл ІТ-проекту.

Методика застосування апарату теорії подібності для побудови аналогових моделей програмних систем багато в чому ідентична випадку фізичних систем. Для побудови аналогової моделі необхідно застосувати метод аналізу розмірностей, для чого виконати наступну послідовність дій.

1. Ввести множину показників, що визначають стан програмної системи.

2. Побудувати систему одиниць виміру і задати розмірності обраних показників.

3. Представити співвідношення між розмірними величинами у вигляді функції

$$a = A \cdot \nu_1^{\alpha} \nu_2^{\beta} \cdot \ldots \cdot \nu_k^{\ell} \nu_{k+1}^{\ell+1} \cdot \ldots \cdot \nu_n^z \qquad (1)$$

де a – величина, що визначається;

$\nu_1, \nu_2, \ldots, \nu_n$ – величини, що визначають;

n – кількість розмірних величин, що визначають;

k – кількість початкових розмірностей.

4. Знайти співвідношення між безрозмірними величинами, що представляють кількісні співвідношення тієї ж системи у вигляді, визначеному π–теоремою (теоремою Бекінгема):

$$\pi = f_1\left(\pi_1, \pi_2, ..., \pi_{n-k}\right), \tag{2}$$

де $\pi, \pi_1, \pi_2, ..., \pi_{n-k}$ – безрозмірні величини.

Для цього необхідно виконати наступні дії:

- підставити розмірності замість самих величин в рівняння (1);
- знайти суму показників ступеня при однакових основних одиницях вимірювання і записати вирази з сумою в систему рівнянь;
- розв'язати отриману систему рівнянь, виразивши через довільно обрані k показників ступеня інші $n-k$;
- підставити виражені показники ступеня в (1) і перетворити залежність до вигляду (2), угруповуючи величини ν_i при однакових показниках ступеня.

5. Обчислити коефіцієнти пропорційності і показники ступеня критеріїв моделі.

В критеріальному рівнянні (2) вигляд функціональної залежності, а саме коефіцієнти пропорційності і показники ступеня критеріїв моделі, визначаються за даними вже виконаного проекту. Для кожного класу процесів буде спостерігатися рівність показників ступенів і коефіцієнтів пропорційності, а також для кожної групи процесів буде спостерігатись рівність критеріїв. Рівняння (2), отримане на 4-му кроці зазначеної вище методики, буде абстрактною аналоговою моделлю процесів виконання програмних проектів.

Рівняння (2) з конкретними значеннями коефіцієнтів пропорційності і показників ступеня критеріїв буде аналоговою моделлю процесів виконання програмних проектів одного класу.

Управління відкритими проектами передбачає збір особливого типу метрик. Розмір програмного коду оцінюється або в функціональних точках або в числі рядків коду SLOC (Source Lines of Code). Кількість розробників враховується двома параметрами – кількістю основних розробників d_c і обсягом залучення сторонніх розробників δ_d, виміряним кількістю розробників, залучених до проекту в одиницю часу. Якість контролюється параметрами b – кількістю внесених помилок і темпом виправлення помилок, r_b – кількістю помилок, що виправляються в одиницю часу. Продуктивність визначається групою параметрів, які враховують зовнішні і внутрішні зміни. Враховується середній обсяг внутрішніх змін (тобто внесених основними розробниками), а саме:

c_ν – кількість одиниць коду, вироблених людиною в одиницю часу, і зовнішні зміни як обсяг надходження доробок в одиницю часу r_b і середній обсяг цих доопрацювань;

p_ν – кількість одиниць коду на одне доопрацювання.

Залежність між обумовленими і визначальними показниками еволюції програмної системи апроксимується як

$$t = A \cdot s^{\alpha} \cdot d_c^{\beta} \cdot \delta_d^{c} \cdot r_p^{d} \cdot p_\nu^{e} \cdot b^{f} \cdot r_b^{s} \cdot c_\nu^{h} \tag{3}$$

Рівняння (3) записане в термінах розмірностей величин, що входять до нього, приводить до співвідношення вигляду

$$\begin{aligned} день = SLOC^{a} \cdot люд^{b} \cdot \left(\frac{люд}{день}\right)^{c} \cdot \left(\frac{дооп}{день}\right)^{d} \times \\ \times \left(\frac{SLOC}{дооп}\right)^{e} \cdot деф^{f} \cdot \left(\frac{деф}{день}\right)^{g} \cdot \left(\frac{SLOC}{день \cdot люд}\right)^{h} \end{aligned} \quad (4)$$

Тут кількість розмірних величин $n = 9$, кількість незалежних розмірностей $k = 5$. Відповідно, необхідно отримати $n - k = 4$ критерія.

Система рівнянь, складена з показників ступенів для кожної розмірності з рівняння (4) має вигляд:

$$\begin{aligned} &\text{день:} 1 = -c - d - g - h \\ &\text{SLOC:}\ 0 = a + e + h \\ &\text{люд:}\ 0 = b + c - h \\ &\text{деф:}\ 0 = f + g \\ &\text{дооп:}\ 0 = d - e \end{aligned} \quad (5)$$

Складаючи перше рівняння з другим і друге з третім в системі (5), а також виражаючи інші величини з останніх трьох рівнянь, отримаємо розв'язок системи:

$$\begin{aligned} &d = e \\ &f = -g \\ &c = a + f - 1 \\ &b = 1 - e - f - 2a \\ &h = -a - e \end{aligned} \quad (6)$$

Підставляючи значення показників з (6) в (5), отримаємо

$$\begin{aligned} t = A \cdot s^{a} \cdot d_c^{1-2a-e-f} \cdot \delta_d^{a+f-1} \cdot r_p^{e} \times \\ \times b^{f} \cdot p_v^{e} \cdot r_b^{-f} \cdot c_v^{-a-e} \end{aligned} \quad (7)$$

Угруповуємо в (7) величини з однаковими показниками ступеня:

$$\frac{t\delta_d}{d_c} = A\left(\frac{s\delta_d}{d_c^2 c_v}\right)^{a}\left(\frac{\delta_d b}{d_c r_b}\right)^{f}\left(\frac{r_p p_v}{d_c c_v}\right)^{e}, \quad (8)$$

де виділяються чотири безрозмірних критерії:

$F_t = \frac{t \cdot \delta_d}{d_c}$ - характеристика часу відкритого проекту;

$F_{scope} = \frac{s \cdot \delta_d}{d^2 \cdot c_\nu}$ - характеристика масштабу проекту;

$F_{qua} = \frac{b \cdot \delta_d}{d_c \cdot r_b}$ - характеристика якості продукту;

$F_{ext} = \frac{r_p \cdot p_\nu}{d_c \cdot c_\nu}$ - характеристика обсягу зовнішніх змін.

У підсумку абстрактна аналогова модель еволюції програмної системи загального призначення, що розробляється за відкритою моделлю, виражається критеріальним рівнянням:

$$F_t = A \cdot F_{scope}^a \cdot F_{qua}^f \cdot F_{ext}^e$$

Аналіз ризиків вибору відкритої моделі. Після обґрунтування вибору відкритої моделі для програмного проекту, необхідно визначити ризики, пов'язані з цим вибором. У разі значних ризиків необхідно скласти план їх розв’язання, або переглянути вибір моделі.

Для програмних проектів з відкритим кодом виділяються наступні джерела ризиків:

• ризики середовища, пов'язані із середовищем виконання проекту:

- $E_{технол}$ - невизначеності технологій, що використовуються;
- $E_{коорд}$ - координація множини учасників проекту;
- $E_{складн}$ - складність системи;

• ризики гнучкості, пов'язані з використанням швидких методів розробки:

- $A_{масшт}$ - масштабованість і критичність;
- $A_{прост}$ - спрощеність проектування;
- $A_{рот}$ - ротація і плинність кадрів:
- $A_{нав}$ - недостатні навички персоналу в швидких методах;

• ризики планування, пов'язані з використанням методів розробки з переважною часткою планування:

- $P_{зм}$ - частота змін;

- $P_{шв}$ - необхідність отримання швидких результатів;
- $P_{вин}$ - виникнення нових вимог;
- $P_{нав}$ - недостатні навички персоналу в методах з плануванням;

• ризики відкритості, пов'язані із застосуванням відкритих методів розробки:

- $O_{сист}$ - несистемність виробленого продукту;
- $O_{код}$ - небезпека відкритості коду;
- $O_{вит}$ - можливість появи породжених проектів;
- $O_{раз}$ - разовий характер виробленого продукту;
- $O_{нав}$ - недостатні навички персоналу у відкритих методах.

Ризики з наведеного вище переліку оцінюються і порівнюються. Завданням порівняння є з'ясування ступеню ризику використання відкритої моделі. Саме порівняння виконується на якісному рівні, і за його результатами ризикам надається рейтинг.

Для оцінки ризику будується дерево рішень. За методом аналітичної ієрархії формуються матриці попарних порівнянь на кожному рівнів критеріїв та рівня альтернатив; обчислюються вектори пріоритетів, індекси та відношення узгодженості оцінок ризиків. Після узгодження оцінок ризиків відбувається синтез глобальних пріоритетів альтернатив відносно фокуса ієрархії – глобального рівня ризику щодо вибору відкритої моделі.

Обсяг планування визначається шляхом виконання послідовності наступних дій:

1. Оцінка ризику. Потрібно оцінити ризики середовища, гнучкості, планування і відкритості. У разі неоднозначності або невизначеності в оцінках, застосувати прототипування або збір і аналіз даних про схожі проекти.

2. Порівняння ризику.

2.а) У разі домінування ризиків гнучкості, збільшити обсяг планування.

2.б) У разі домінування ризиків планування, збільшити гнучкість, зменшуючи планування.

2.в) У разі домінування ризиків відкритості, переглянути рішення про відкритість проекту.

3. Аналіз ризику. У разі якщо в проекті існують елементи, для кожного яких задовольняються умови 2а, 2б або 2в, розбити проект на підпроекти.

4. Модифікація життєвого циклу. Розробити стратегію прийняття ризиків проекту інтегруючи плани прийняття кожного з ідентифікованих ризиків.

5. Виконання проекту. Відстежувати ризики по ходу виконання проекту і, в разі появи дисбалансу, виконати переоцінку обсягу планування і відкритості.

Аналіз ризиків дозволяє визначити обсяг планування як якісно, так і кількісно. В результаті застосування методу для ІТ проекту або для його окремих підпроектів визначається необхідність у збільшенні або зменшенні обсягу планування. У разі, якщо ризики відкритості домінують і для проекту не існує оптимальної стратегії зменшення таких ризиків, відкрита модель розробки застосовуватися не повинна, і рішення про відкритості проекту має бути переглянуто.

Зокрема, при виконанні дії 4 виробляється стратегія управління проектом або групою підпроектів, що мінімізує ідентифіковані ризики. Для кожної групи

ризиків – відкритості, гнучкості та планування, приймаються рішення про, відповідно, закриття частин або всього проекту, збільшення, або зменшення обсягу планування. Потім для кожного джерела ризику розробляється стратегія його розв'язання і вносяться зміни до життєвого циклу проекту шляхом додавання необхідних процесів, або розв'язання, або додаткового контролю ризику.

Для кількісного визначення обсягу планування необхідно розрахувати очікуване грошове значення двох груп ризиків – ризиків недостатнього планування $R_{план}$ та раннього виведення продукту на ринок $R_{рин}$.

$$R_{план} = P(L) \cdot S(L),$$
$$R_{рин} = P(L) \cdot S(L),$$

де $P(L)$ - ймовірність втрат,

$S(L)$ - розмір втрат.

Ризики $R_{план}$ і $R_{рин}$ розраховуються для різних величин витраченого часу і трудовитрат на планування з метою отримання залежностей,

$$R_{план} = f(x),$$
$$R_{рин} = \varphi(x),$$

де t - обсяг трудовитрат або часу на планування. Мінімум функції сумарного ризику $R(t)$ буде визначати необхідний обсяг планування $t_{план}$:

$$R(t) = R_{план}(t) + R_{рин}(t),$$
$$R(t_{план}) \to \min \Leftrightarrow \left(\frac{\partial R}{\partial t}\right) = 0.$$

Аналіз залежності $R_{план}$ показує, що чим менше інвестується в планування, тим більша ймовірність того, що плани будуть недосконалими, неоднозначними, мати прогалини. Також збільшуються можливі фінансові втрати через постійні перероблення і уточнення, втрати або ротації персоналу. У той же час, чим точніший план, тим менше ймовірність втрат через планування і менше втрат.

Аналіз залежності $R_{рин}$ вказує на те, що малі обсяги планування призводять до раннього виведення продукту, що розробляється в проекті, на ринок і, отже, отримання більшого прибутку. Надмірне планування призведе до затримок у випуску продукту і, отже, дозволить конкурентам перехопити ініціативу і призведе до підвищення втрат.

Функція $R(t)$ суми очікуваного грошового еквіваленту ризиків містить точку мінімальних очікуваних втрат із значенням абсциси $t_{план}$, яка і є шуканим кількісним виразом необхідного обсягу планування.

Висновки та перспективи подальших досліджень в даному напрямку. Розроблена модель життєвого циклу програмних проектів з відкритим програмним кодом, яка визначає перелік і послідовність дій для виконання в проекті, визначає місце процесів управління в життєвому циклі.

Сформульована узагальнена методика побудови аналогової моделі для оцінки часу прототипування програмних проектів.

Отримана абстрактна аналогова модель еволюції програмної системи загального призначення, яка виражається критеріальним рівнянням.

Здійснена реалізація методики управління програмними проектами, яка включає:

- обґрунтування вибору моделі методом аналізу характерних категорій;
- аналіз ризику вибору моделі;
- створення аналогової моделі життєвого циклу проекту;
- визначення часу прототипування проекту.

Проведено аналіз ризиків вибору певного типу IT-проекту, визначено напрямки розв'язання ризиків та обсяги планування. Зроблено висновок про можливість застосування розробленої методики для прогнозування можливості успішного завершення різних класів IT-проектів на ранніх стадіях проектування.

ЛІТЕРАТУРА

1. CHAOS Report [Electronic Resource]. – The Standish Group International. Inc. http://www.projectsmart.co.uk/docs/chaos–report.pdf
2. Jian, Z. Why IT Projects Fail // Computervvorld. – 2005. – Vol. 39, № 6. – P. 31–32.
3. Glass, R. L. System Development Glass Column // System Development. – 1988. –Vol. 1.№1.–P. 4–5.
4. Брукс, Ф. Мифический человеко–месяц или как создаются программные системы. – СПб.: Символ–Плюс. 2006. – 304с.
5. Mystery Solved! Linux is Cheaper – PERIOD. / The Standish Group International, Inc. // VirtualBEACON. – 2004. № 347. – P. 1–3.
6. Ройс, У. Управление проектами по созданию программного обеспечения. Унифицированный подход. – М.: Издательство "Лори". 2002. – 424с.
7. Sytnyk, V.A., Bulashov, V.V. Methodology for managing the development of it projects with open source/ 5th International conference on Eurasian scientific development in 2018: new methods and solutions». Proceedings of the Conference (September 02, 2018). Premier Publishing s.r.o. Vienna. 2018.46 p. ISBN-13 978-3-903197-73-2
8. Ситник, В.А., Булашов, В.В. Аналогова модель управління IT проектами з відкритим кодом//5-th International Conference on Information technology and interactions (IT&I-2018). Taras Shevchenko National University of Kyiv, November 20-21, 2018.
9. Тесленко, П.А. Эволюционная теория и синергетика в управлении проектами / П.А.Тесленко // Управління проектами та розвиток виробництва: Зб.наук.пр. – Луганськ: вид-во СНУ ім. В.Даля, 2010. - № 4(36). - С. 38-44.
10. Schindler, E. OSS/Linux Development Survey [Electronic Resource] // Evans Data Corporation Strategic Reports http://evaiisdata.coin/reports/viewRelease.php?reportID=7.
11. Routh, E.T. A Treatise on the Stability of a Given State of Motion. – London: Taylor & Francis, 1975. – 297 p.
12. Wlieeler, D.A. Why Open Source Software / Free Software (OSS/FS. FLOSS, or FOSS)? Look at the Numbers! // ComputerWorld. - 2005. - N° 7. - p. 26.
13. Little, G., Healey, M. Worldwide Open Source Sendees 2007–2011 Forecast [Electronic Resource]. URL: http://www.idc.com/getdoc.isp?containerId=20S2 55.
14. NetCraft Internet Research Analysis [Electronic Resource] http://www.netcraft.coin.

15. Ghosh, R.A. Study on the economic impact of FLOSS on innovation and competitiveness of the EU ICT sector: Final Report / UNU–MERIT. – Contract ENTR/04/112. – The Netherlands. 2006. – 2S7p.
16. Barska, I. Algorithm of Distributing the Team Load for IT-Project / Barska I., Teslenko P., Fesenko T., Voznyi O. // Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). — Warsaw : University of Technology, 2015. — p. 559 – 562.
17. ISO TEC 12207:1995. Software life cycle processes. – Geneva. Switzerland: International Organization for Standardization. – 57p.
18. ДСТУ 3918–1999 (ISO IEC 12207:1995) Державний стандарт України. Інформаційні технології. Процеси життєвого циклу програмного забезпечення. – К.: Держстандарт України, 2000. – 49 с.
19. Teslenko, P. Increasing probability of successful projects complete / P. Teslenko, S. Antoshchuk V.Krylov // Proceedings of the International Research Conference at the Dortmund University of Applied Sciences and Arts took place on June 30th -July 1st 2017 for the seventh time. — 2017. — Dortmund : the Dortmund University. — P. 28-30
20. Teslenko, P. 3-Level Approach to the Projects Planning / P. Teslenko, S. Antoshchuk, D. Bedrii, H. Lytvynchenko // XIII th International Scientific and Technical Conference «Computer science and information technologies» 11-14 September, 2018. — Lviv, 2018. — pp. 195-198.
21. Керівництво з питань проектного менеджменту / Під ред. С.Д. Бушуева. – К.: Видавничий дім "Деловая Украина". 2000. – 198 с.
22. Шафер, Д.Ф., Фартредт, Р.Т., Шафер, Л.И. Управление программными проектами: достижение оптимального качества при минимуме затрат. – М.: Издательский дом “Вильямс”, 2003. – 1136 с.
23. Барская, И.С. Особенности принятия решения на этапе инициации проектов создания корпоративных информационных систем / И.С. Барская, П.А. Тесленко, В.Ю. Денисенко // Управління проектами та розвиток виробництва: Зб.наук.пр. – Луганськ: вид-во СНУ ім. В.Даля, 2014. - №1(49). - С. 32-39.
24. Шерстюк, О.І. Использование метода ранжирования при формировании необходимого набора компетенций команды проекта / Вестник НТУ «ХПИ». Серия: стратегическое управление, управление портфелями, программами и проектами. – 2018. – Вып. 2 (1278). – С. 31–37.
25. Гухман, А.А. Применение теории подобия к исследованию процессов тепло–массообмена.–М.: Высшая школа, 1974. – 328с.

Рецензент статті
д.т.н., проф. Медведєва О.М.

Стаття рекомендована до
публікації 18.09.2018 р.