ISSN 2311-4738 (print)
ISSN 2413-3000 (online)

УДК 005.8: 004.052.42

С. А. МУРАВЕЦЬКИЙ, С. О. КРАМСЬКИЙ

ПЛАНУВАННЯ ПРОЦЕСІВ ЗАБЕЗПЕЧЕННЯ ЯКОСТІ У ВЕЛИКИХ ТА ГЕОГРАФІЧНО РОЗПОДІЛЕНИХ ГІБРИДНИХ ІТ–ПРОЕКТАХ

Розкрито особливості операційно-проектної діяльності у великих, географічно розподілених ІТ-проектах. Надано структуру процесів з забезпечення контролю якості кінцевого продукту. Розглянуті сучасні методи інтеграції процесів тестування у процеси розробки програмного забезпечення. Розглянуті групи існуючих моделей планування та розробки програмних продуктів. Надана стисла характеристика процесів контролю якості у кожній групі. Надані рекомендації з адаптації належних процесів контролю якості у гібридному проекті.

Ключові слова: ІТ-проект, якість, метод, процес, програмне забезпечення, операційно-проектна діяльність.

Вступ. Проектний контекст у безперервний розвиток індустрії інформаційних технологій зробив програмне забезпечення невід'ємним елементом діяльності людини та бізнесу. Сучасні напрямки інформаційного удосконалення, такі як Internet of Things, націлені на побудову середовища у якому пристрої під управлінням програмних продуктів керуватимуть усілякими повсякденними процесами суспільного та особистого життя. У цих умовах можна сказати, що необхідність створити якісний продукт набуває нового сенсу, зростає вплив якості на економічний результат проекту.

Аналіз останніх публікацій та досліджень. У роботі [1] зазначено – комп'ютеризація дала можливість автоматизувати численну кількість професій та позбавити продукти їх праці негативного впливу «людського фактору», скоротити кількість виробничих дефектів, підвищити якість кінцевого продукту проекту. У той же час виробництво програмного забезпечення є суто «мануальною» справою. Програмний код розробляють люди, а кінцевий продукт у повної мірі потрапляє під дію того самого «фактору».

У статті [2] розглядається перспектива впливу інновацій тренду Internet of Things на повсякденне життя людини. Наприклад програмно регульовані світлофори, реверсивний рух транспорту, «розумне» дорожнє покриття що надає інформацію про свій стан (пошкодження, ожеледь) на бортові комп'ютери автомобілів та у відповідні муніципальні структури, тощо. Спектр застосування систем, які збирають сенсорні дані із середовища та переробляють їх на інформацію чи дії корисні для споживача є віртуально необмеженим. Можливо припустити, що масштаб подібних проектів буде великим, привабливим для інвесторів, а продукти істотно впливатимуть на якість повсякденного життя людини.

Забезпечення належних стандартів якості програмного продукту може бути вирішальним фактором успіху чи провалу такого проекту.

У роботі [3] зазначено що тільки у автомобільній індустрії за останні 10 років мільйони транспортних засобів були відкликані через несправності у програмному забезпеченні. Збитки з цього приводу обчислювалися у мільярдах доларів США. Відомі авто – концерни частково втратили свою долю на ринку, та репутацію бренду.

Дефектація програмного продукту на стадії виробництва набагато дешевша ніж у прикладах наданих вище [1]. Тому тестування, основний інструментарій з підвищення контролю якості, повинне бути детально сплановано та проведено у повному обсязі.

На рис. 1 представлені градації вартості виправлення дефектів у складних програмних продуктах залежно від фази розробки в якій вони були знайдені. Цифрами на рис. 1 відображено порядок ціни, тобто у фазі дизайну вартість є в двічі більшою ніж у фазі формування вимог до продукту.

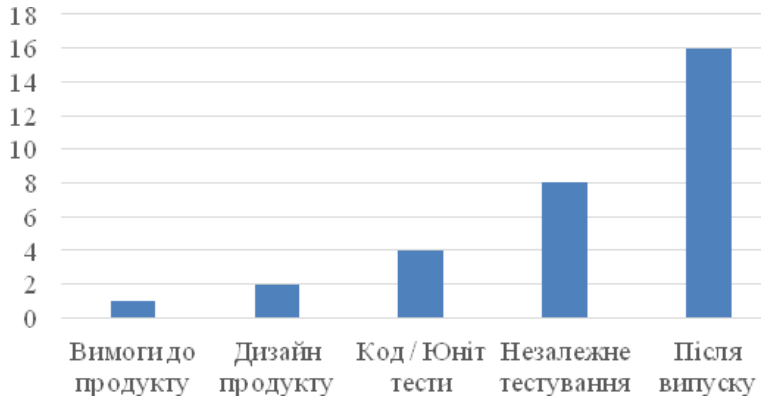

Рис. 1 – Відносна вартість виправлення програмного дефекту у складній системі

Розглянемо фази надані на рис. 1.

Вимоги до продукту. Продукт має бути саме таким, яким він був замовлений. Згідно з [4] істотні зміни до вимог потребують повного перегляду проектних планів.

Дизайн продукту. Перевірка відповідності архітектури програмного продукту до поставлених вимог.

Код та «юніт» тести. Перевірка примітивних, низькорівневих функцій продукту проекту.

Незалежне тестування, або *тестування з точки зору кінцевого користувача*. Перевірка комплексу сценаріїв використання системи згідно із суттю продукту проекту та смислу потреби яку він задовольняє.

Як бачимо фази істотно відрізняються між собою. Кожна з цих фаз потребує планування належних заходів тестування. Стратегія контролю якості має інтегруватися у обрану на проекті модель розробки програмного забезпечення.

Мета дослідження. Розкрити особливості планування якості у гібридному IT-проекті великого обсягу [5]. Висвітити проблематику координації географічно розподілених (віртуальних) команд у рамках проекту. Надати рекомендації до забезпечення якості продукту у такому особливому проекті.

Виклад матеріалу. Методи забезпечення якості програмного продукту істотно зв'язані з моделлю виробництва продукту проекту. Фундаментально функція контролю якості є сервісною відносно до функції розробки. У контексті IT – проекту команда з тестування надає послугу команді розробників, продуктом якої є виявлені несправності, недоліки, рекомендації щодо поліпшення окремих функцій та рекомендації з планування послідовності виправлення дефектів. Тобто існує прямий зв'язок між обраною методологією та формуванням належних процесів контролю якості проекту.

Принципово існуючі моделі можна розподілити на три групи: каскадні або жорстко плановані; гнучкі [6, 7]; процесно-орієнтовані із фокусом на продукт проекту [8];

На рис. 2 надана схема каскадної методології на прикладі моделі водоспаду (англ. waterfall).

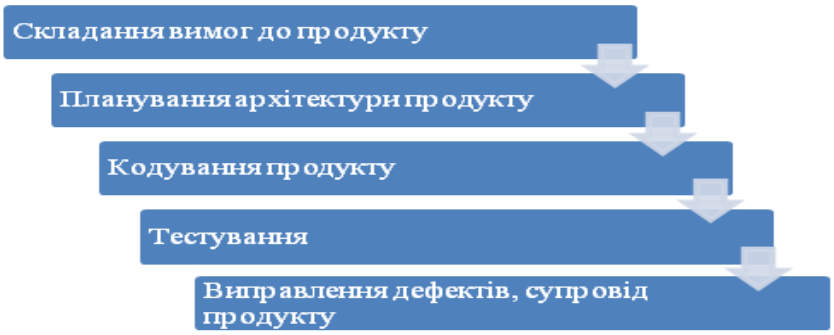

Рис. 2 – Каскадна модель виробництва програмних продуктів

Посилаючись на рис. 2 бачимо, що фази створення продукту проекту жорстко закріплені. Групи робіт починаються послідовно, виключно після завершення попередньої групи.

Група якості є передостанньою, що як зазначено на рис. 1 значно підвищує вартість виправлення дефектів продукту. Очевидно, що у разі виявлення фундаментальної проблеми на фазі тестування проект буде потрібно майже переробити наново. Також можна сказати, що таки проекти є довготривалими і мають великі ризики пов'язані із завершенням проекту у встановлений термін. Важливо зазначити, що між стартом фази розробки (кодування) продукту та початком тестування може пройти декілька місяців. Тому відповідні команди IT проектів часто працюють окремо, на відстані, практично не контактуючи одна з одною. Комунікація та синергія існує лише на рудиментарному рівні, що також негативно впливає на швидкість реалізації такого проекту [9].

Застосування каскадних моделей у сучасній IT індустрії викликає сумніви. Сучасний тренд – максимально скоротити виробничий цикл, бажано без втрати якості, випускати нові продукти та оновлення як можна швидше. У світі мобільних додатків та веб-сайтів продукт часто відновлюється кожен тиждень, у деяких випадках навіть частіше. Каскадні моделі не в змозі забезпечити таку швидкість, їм бракує гнучкості процесів та механізмів адаптації до зміни вимог від продукту проекту.

Далі розглянемо чи відповідає сучасному ринку наступна група моделей – гнучкі (англ. agile). На рис. 3 відображен приклад гнучкої моделі – Scrum.

Гнучкі моделі зазначають, що фази розробки проекту, існуючи у каскадній моделі, є необхідними. Про те процеси супроводжуючі їх мають бути кардинально змінені. Каскадний путь вимоги – дизайн – кодування – тестування – виправлення помилок має здійснюватися ітеративно доки продукт не буде готовий до випуску. Кожна ітерація (спринт) має реалізовувати певний під – продукт, надавати у результаті готову до використання функцію із належним рівнем якості. Комунікація між командами проекту, та між командами і представником замовника, або стейкхолдерів, має бути безперервною. Модель Scrum передбачає однойменні щоденні Scrum – мітинги, мета яких обмін важливою проектною та технічною інформацією між членами команди IT – проекту, та корекцію планів. Ітерації мають бути короткотривалими (1 – 4 тиждня), а управління діями команди проекту всередині ітерації суто тактичним.

На місце скрупульозного планування графіку робіт, або WBS прийнятого у каскадній моделі, приходить планування засноване на адаптації до змін, швидкому прийнятті рішень та зміни завдань для конкретної команди проекту чи члена команди.

Черга робіт замінюється списком із якого роботи обираються згідно із пріоритетами проекту у конкретний момент часу. Роботи, які потрапили до ітерації, плануються у її рамках. Також стирається грань між командами програмістів і контролю якості, є єдина команда IT – проекту і одна мета – завершити спринт в строк та у повному обсязі.

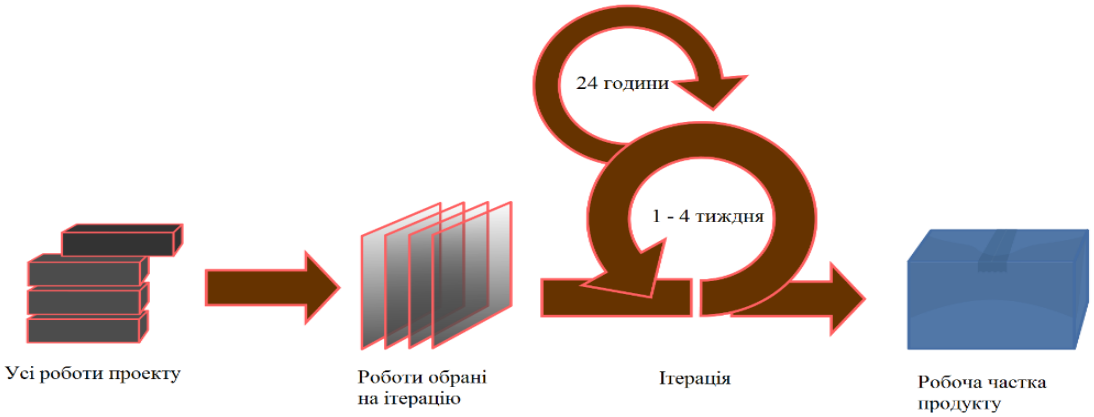

Рис. 3 – Гнучка модель Scrum

Гнучкі моделі значно підвищують вимоги до процесів забезпечення якості програмних продуктів. У каскадній моделі менеджер відділу якості мав змогу планувати віртуально будь-який обсяг робіт, та відокремлено виконувати їх протягом відповідної фази. У рамках agile темпи виконання тестів мають бути набагато швидшими, обсяг покриття продукту тестами часто меншим. Інженери з тестування є невід'ємною частиною усіх фаз планування, Scrum – мітингів, тощо. Багато уваги слід приділяти саме управлінню ризиками якості, людського фактору та автоматизації тестування. Програмні продукти мають схильність до регресії. Зміна чи нарощування функціоналу в однієї частині продукту проекту може призвести до збоїв у інших частинах, які вже були протестовані, та удосконалені раніше. Залежно від комплексності системи обсяг регресійних дефектів може сягати 30% від загального. Тому тестувати одне й те саме протягом циклу розробки проекту потрібно декілька разів. Регресійне тестування є одним із найдорожчих заходів відділу якості. Також слід пам'ятати, що інженери з контролю якості, як будь-яка людина піддаються «людському фактору». Тобто інколи помиляються. М'який компонент проекту – людина. Помилка при тестуванні – потенційний збиток в проекті. Тому зростають вимоги до рівня кваліфікації інженерів саме з контролю якості. Автоматизація тестування є ключовим елементом вчасного виконання проекту на базі гнучкої моделі. Бажано автоматизувати нову функціональність у тому ж спринті у яку вона розробляється, та використовувати процес безперервної інтеграції та випуску (англ. Continuous Integration / Continuous Delivery). Цей процес забезпечує щоденну збірку продукту з урахуванням усіх останніх змін, автоматичне тестування та звітування у неробочий час. Тобто кожного дня команда має у своєму розпорядженні актуальну у даних момент часу версію продукту проекту, та інформацію про загальний статус якості цієї версії. Вартість регресійного тестування мінімізується. Програмісти значно скоріше отримують зворотній зв'язок з приводу регресії. Регресійні дефекти виправляються у тому – ж спринті. Так само мінімізується вартість планування та управління стабілізацією продукту проекту. Структурна схема процесу безперервної інтеграції та випуску надана на рис 4.

Рис. 4 – Continuous Integration / Continuous Delivery

Кожній з етапів, окрім першого, із зазначених на рис. 4 генерує зворотній зв'язок з приводу якості системи. Будь яка несправність, знайдена у процесі призведе до зміни програмного коду, потім до нової збірки, запуску «юніт» тестів і так далі. Гнучкі методи займають лідируючи позиції у сучасній індустрії виробництва програмних продуктів. Значна більшість дискретних проектів, які мають бути почати з нуля, та не орієнтуються на багаторічний розвиток та глобальні метаморфози продукту може бути виконана на базі гнучкої методології. Далі розглянемо гібридний проект, реалізація якого гнучкими методами пов'язана із певними ускладненнями. Характеристика проекту і продукту наведена у табл. 1.

Таблиця 1 – Характеристика гібридного проекту

Проект	Тип	Гібридний проект
	Діяльність	Операційно-проектна; випуск оновлень до існуючого програмного продукту; реструктурізація застарілих існуючих виробничих процесів (каскадна модель – гнучка модель).
	Команди IT проекту	Географічно розподілені; часові пояси GMT + 2, GMT + 7, GMT + 8; команди проекту з розробки та тестування знаходяться у різних поясах; досить різні національні культури (менталітет) ведення бізнесу; близько сотні інженерів.
Програмний продукт	Низький обсяг автоматизації; 30% модулів не підлягає автоматизації; співвідношення нових функцій до регресійних складає 10 / 90.	

Основуючись на табл. 1 означимо проблематику використання Scrum у даному випадку. Істотна різниця у часових поясах розташування проектних команд лімітує комунікаційні можливості, ускладнює використання щоденних зустрічей – ключового управлінського інструменту. Також на обмін інформацією негативно впливає мовний бар'єр, великий обсяг виробничої групи та культурні відмінності між локаціями. Іще один фактор впливу часового поясу – швидкість обробки зворотного зв'язку отриманого від тестів. Часто на момент знайдення проблеми відповідальний інженер вже закінчив робочий день. Низький обсяг покриття автоматичного тестування та великий обсяг регресійного тестування нівелює концепцію спринту.

Оптимізація процесів (прийнятність) у такому проекті можлива за допомогою моделі виробництва із третьої групи – процесно – орієнтованих із фокусом на продукт, додаючи специфічні до конкретного випадку вдосконалення. Третя група моделей базується на ідеї розглядання комплексного проекту, як програми проектів меншого обсягу. Виділення під – продуктів у рамках продукту проекту, та виконання кожного з них за допомогою найбільш ефективної, прибуткової моделі.

У розглянутому випадку рекомендується виділити наступні під – проекти: автоматизації регресійного контенту; розробки нових функцій; модернізації виробничих процесів; «ко-локації» або зосередження розробки та управління під – проектом суто у одній географічній локації. Останній пункт надає можливість реалізовувати під – проекти суто у рамках гнучких методів, використовуючи усі корисні властивості притаманні цій групі методів.

Висновки. У роботі розглянуті основні моделі виробництва програмних продуктів, розглянуті особливості їх використання в гібридному проекті. Надані рекомендації до вибору методології, адаптації її під кожний конкретний гібридний IT проект та організації процесів контролю якості.

Список літератури: **1.** *Black, R.* Foundations of Software Testing: ISTQB Certification [Text] / *R. Black, E. Van Veenendaal, D. Graham.* – 3rd ed. – Cengage Learning EMEA, Cheriton House, North Way, Andover, Hampshire, SP10 5BE, United Kingdom, 2012. – 242 с. **2.** *Burrus, A.* The Internet of Things is far bigger than anyone realizes [Електронний ресурс] / *A. Burrus.* – Режим доступу : http://www.wired.com/insights/2014/11/the-internet-of-things-bigger/ **3.** *Sullivan, M. K.* How Software Issues in Cars Cost Automakers Billions [Електронний ресурс] / *M. K. Sullivan.* – Режим доступу : http://blog.smartbear.com/code-review/how-software-issues-in-cars-cost-automakers-billions/ **4.** A Guide to the Project Management Body of Knowledge (PMBOK. Guide) [Text]. – Fifth Edition, USA : PMI, 2013. – 586 с. **5.** *Сидорчук, О. В.* Методологічні засади управління гібридними проектами [Текст] / *О. В. Сидорчук, Р. Т. Ратушний, Л. Л. Сидорчук* // Вісник НТУ «ХПІ». – 2015. – № 1. – С. 66–71. **6.** Manifesto for Agile Software Development [Електронний ресурс]. – Режим доступу : http://agilemanifesto.org/. **7.** A guide to the Scrum Body of Knowledge (SBOK Guide), 2013 Edition, © 2013 SCRUMstudy, a brand of VMEdu, Inc., 410 N 44th Street, Phoenix, Arizona 85008 USA, 340 с. **8.** PRINCE2 Agile, AXELOS Limited 2015, TSO, PO Box 29, Norwich, NR3 1GN, 360с. **9.** *Крамський С. О.* Плануваня команд проектів крюінговими компаніями [Текст] / *С. О. Крамський* // Вост.-европейский жур. передов. технологий. – № 1/3 (43). – Х. 2010. – С. 70–72.

References: **1.** Black, R., E. Van Veenendaal, Graham, D., (2012). *Foundations of Software Testing: ISTQB Certification 3rd Edition.* Hampshire, United Kingdom, Cengage Learning EMEA. **2.** Burrus, A. (2014). The Internet of Things is far bigger than anyone realizes. *wired.com.* Retrieved from http://www.wired.com/insights/2014/11/the-internet-of-things-bigger/ **3.** Sullivan, M. K. (2015). How Software Issues in Cars Cost Automakers Billions. *blog.smartbear.com.* Retrieved from http://blog.smartbear.com/code-review/how-software-issues-in-cars-cost-automakers-billions/ **4.** A Guide to the project management body of knowledge (PMBOK guide 5th editon). (2013). USA: PMI Standards Committee, 589 **5.** Sydorchuk, O. V., Ratushniy, R. T., & Sydorchuk, L. L. (2015). Metodologichni zasady upravlinnya gibrydnymy proektamy [Methodological principles of hybrid project management]. *Visnyk NTU «HPI» - Bulletin "KhPI", 1,* 66–71 [in Ukrainian]. **6.** Manifesto for Agile Software Development. (2001). *agilemanifesto.org.* Retrieved from http://agilemanifesto.org/ **7.** A Guide to the Scrum Body of Knowledge (SBOK Guide). (2013). Ed. Phoenix, AZ: SCRUMstudy, a brand of VMEdu, Inc. **8.** PRINCE2 Agile (2015). Norwich, PO Box 29 : AXELOS Limited. **9.** Krams'kyy, S. O. (2010). Planuvanya komand proektiv kryuinhovymy kompaniyamy [Project teams planning in crewing companies]. *Vost.-evropeyskyy zhur. peredov. tekhnolohyy, 1,* 70-72 [in Ukrainian].

Надійшла (received) 25.11.2015

Відомості про авторів / Сведения об авторах / About the Authors

Муравецький Святослав Аркадійович – магістрант, Одеський Регіональний Інститут Державного Управління, м. Одеса, тел.: +38 (093) 465-19-65, e-mail: smuravetskiy@lohika.com.

Muravetskiy Svyatoslav Arkadiyovich – undergraduate, Odessa Regional Institute of Public Administration, Odessa, tel: +38 (093) 465-19-65, e-mail: smuravetskiy@lohika.com.

Крамський Сергій Олександрович – кандидат технічних наук, старший викладач, Одеський Регіональний Інститут Державного Управління, м. Одеса, тел.: +38 (093) 295-99-10, e-mail: morsubs@rambler.ru.

Kramskiy Sergiy Oleksandrovych – Candidate of Technical Sciences, PhD, Odessa Regional Institute of Public Administration, Odessa, tel: +38 (093) 295-99-10, e-mail: morsubs@rambler.ru.