ПРОТИРІЧЧЯ ВИКОРИСТАННЯ AGILE ПРИ СТВОРЕННІ ПРОГРАМНОГО ПРОДУКТУ

УДК 004.9

ЯШИНА Оксана Миколаївна
к.т.н., ст. викладач кафедри інженерії програмного забезпечення Хмельницького національного університету
Наукові інтереси: дослідження та впровадження смарт-технологій у різні сфери життєдіяльності людини.
e-mail: ksusha_ja@mail.ru.

ВСТУП

До недавнього часу найбільш розповсюдженою моделлю планування та розробки програмного забезпечення була так звана каскадна модель, при якій виконання робіт здійснюється послідовно. Згідно підходу каскадної моделі всі фази проекту реалізуються послідовно, і перехід до наступної фази можливий тільки після повного та успішного завершення попередньої. Одним із основних властивостей даного підходу – високий рівень формалізації при визначенні вимог та проектування. Ця перевага методик, заснованих на цьому підході, – узгодження на старті проекту запланованих результатів, знімає ризики та підвищує прозорість. Разом з тим, формалізація знижує можливості по тонкому налаштуванню робіт, врахуванню змін зовнішнього середовища та вимог; а процедури узгоджень, висока деталізація планування підвищують вартість проекту. Таким чином пріоритет формального виконання робіт перед термінами, вартістю та якістю, бюрократизація та жорстке слідування черговості фаз проекту стали такими аспектами каскадного підходу розробки програмного забезпечення, що найбільш критикується.

Отже, основні проблеми, що виникають при розробці програмного забезпечення: якість, вартість, надійність. У зв'язку з цим правильне планування та організація процесу розробки програмного забезпечення є основою досягнення бажаного результату в очікувані терміни, з очікуваним рівнем якості та адекватним бюджетом. Серед загальнопоширених проблем процесу розробки програмного забезпечення зустрічаються такі:

1.Зміна вимог безпосередньо в процесі розробки.

2.Нечіткий розподіл відповідальності за виконувану роботу та її результат.

3.Безперервний потік дрібних, швидко змінюваних вимог, що відволікають розробників та менеджерів від основного напрямку робіт.

4. Як наслідок, зрив термінів, роздування бюджетів, втрата якості [1].

Для вирішення завдання успішної організації процесу розробки ПО була створена гнучка методологія розробки програмного забезпечення.

Метою статті є визначення теоретичних аспектів застосування сучасних методологій розробки програмного забезпечення, а також обґрунтування доцільності та виокремлення протиріч використання гнучких методологій при створенні програмного продукту у різних сферах життєдіяльності людини.

АНАЛІЗ ДОСЛІДЖЕНЬ ТА ПУБЛІКАЦІЙ

Методологічною основою нашого дослідження стали праці таких дослідників як Кніберг Х., Schwaber Ken, Карпов Д.В., Пушкарьов А. Н., Ярошок А.С., та багато інших. Деякими загальними аспектами використання гнучких методологій займались Карпов Д.В., Пушкарьов А.Н. Використання SCRUM у своїх роботах описують Бичков І.В., Putu Adi Guna Permana, Попендик М.Т. Загалом же в теперішній час виконується велика кількість різноманітних досліджень, присвячених застосуванню гнучких методологій під час розробки програмного продукту. Однак, потрібно відмітити, що проблема протиріччя у використанні Agile займає незначне місце у вітчизняних та зарубіжних роботах.

Не зважаючи на те, що дослідження в галузі застосування гнучких методологій постійно проводяться, наукових робіт, присвячених доцільності їх використання в тій чи іншій сфері розробки та впровадження програмного забезпечення недостатньо, що і обумовило виявлення до цього питання підвищеного наукового та практичного інтересу.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Гнучка методологія розробки - маніфест, який визначає спосіб мислення і містить основні цінності та принципи, на яких базується кілька підходів (фреймворків) до розробки програмного забезпечення (хоча останнім часом спостерігається тенденція та спроба застосування гнучкої методології розробки в інших напрямках діяльності, окрім інформаційних технологій), що являє собою інтерактивну розробку, періодичне (динамічне) надання (оновлення) вимог від замовника та їх реалізації за допомогою самоорганізованих робочих груп, сформованих з експертів різного профілю (розробники, тестувальники і т.д.). На даний момент існує безліч фреймворків (методологій), підходи яких базуються на гнучкій методології розробки, наприклад такі, як: Scrum, Екстремальне програмування, FDD, DSDM і т. д.

Agile – одна і методологій ітеративної та покрокової розробки програмного забезпечення (ПЗ), на противагу традиційній лінійній методології «водоспад». Методологія гнучкої розробки визначає систему методів проектування, розробки і тестування на протязі всього життєвого циклу ПЗ. Методи гнучкої розробки (наприклад, SCRUM) засновані на оперативному реагуванні на зміни за рахунок застосування адаптивного планування, спільного вироблення вимог, раціоналізації систем, самоорганізованих крос-функціональних групах розробників, а також покрокової розробки ПЗ з чіткими часовими рамками. Цей підхід використовується в багатьох сучасних проектах розробки комерційного ПЗ.

В основі гнучкої методології розробки лежить ліберально-демократичний підхід до управління і організації праці команд, члени якої сконцентровані на розробці конкретного програмного забезпечення. За рахунок того, що розробка ПЗ із застосуванням гнучкої методології визначає серії коротких циклів (ітерацій), з тривалістю 2-3 тижні, досягається мінімізація ризиків по завершенню кожної ітерації Замовник приймає результати і видає нові або коригувальні вимоги, тобто контролює розробку і може на неї відразу впливати. Кожна ітерація включає в себе етапи планування, аналізу вимог, проектування, розробку, тестування і документування. Зазвичай однієї ітерації мало для випуску повноцінного програмного продукту, але при цьому після закінчення кожного етапу розробки повинен з'являтися «відчутний» продукт або частина функціоналу, яку можна подивитися, протестувати і видати додаткові або коригувальні заходи. На основі виконаної роботи, після кожного етапу, команда підводить підсумки і збирає нові вимоги, на підставі чого вносить корегування в план розробки ПЗ.

Однією з основних ідей Agile, є взаємодія всередині команди і з замовником лицем до лиця, що дозволяє швидко приймати рішення і мінімізує ризики розробки програмного забезпечення, тому команду розміщують в одному місці, з географічної точки зору. Причому в команду входить представник замовника (англ. product owner – повноважний представник замовника або сам замовник, який представляє вимоги до продукту; таку роль виконує менеджер проекту від замовника або бізнес-аналітик).

Основоположними принципами Agile-маніфесту є:

1. Задоволення потреб замовника є найвищим пріоритетом, завдяки регулярному та ранньому постачанню цінного програмного забезпечення.

2. Зміна вимог вітається, навіть на пізніх стадіях розробки. Agile-процеси дозволяють використовувати зміни для забезпечення замовнику конкурентної переваги. Це важливий принцип, в існуючих реаліях, коли продукція високотехнологічних галузей застаріває дуже швидко. Якісь нові вимоги до продукту можуть бути сформовані вже на фінальній стадії розробки у зв'язку із змінами конкурентного чи ринкового середовища. Цей принцип не може бути реалізовано в каскадній моделі, або буде коштувати дуже дорого.

3. Працюючий продукт слід випускати якомога частіше, із періодичністю від декількох тижнів до декількох місяців. Це, в першу чергу, відноситься до продуктів, що мають версійність.

4. Протягом всього проекту розробники та представники бізнесу повинні щоденно працювати разом.

5. Над проектом повинні працювати мотивовані професіонали, тобто такі спеціалісти, яких мотивує сам проект. Щоб робота була зроблена, повинні бути створені гарні умови та забезпечено підтримку.

6. Безпосереднє спілкування є найбільш практичним та ефективним способом обміну інформацією як із самою командою, так і всередині команди.

7. Працюючий продукт – основний показник прогресу.

8. Інвестори, розробники та користувачі повинні мати можливість підтримувати постійний ритм безкінечно. Agile допомагає налагодити такий стійкий процес розробки.

9. Постійна увага до технічної досконалості та якості проектування підвищує гнучкість проекту.

10. Простота – мінімізація зайвої роботи.

11. Найкращі вимоги, архітектурні та технічні рішення виникають у самоорганізованих команд.

12. Команда повинна систематично аналізувати можливі способи покращення ефективності та відповідно корегувати стиль своєї роботи.

На основі цінностей та принципів Agile були сформовані та набули подальшого розвитку такі гнучкі методології розробки [4,5]:

— Agile Modeling (AM) – даний підхід в своїй основі визначає процедури моделювання (в т.ч. перевірку моделі коду) і документування в рамках розробки програмного забезпечення. У меншій мірі описані процедури проектування і побудови діаграм на мові UML. Також залишаються поза увагою області розробки, тестування, управління проектом, впровадження та супровід.

— Agile Unified Process (AUP) – уніфікована версія методології RUP (IBM Rational Unified Process), яка була сформирована Скоттом Амблером. AUP визначає модель створення програмного забезпечення в рамках бізнес-додатків.

— Agile Data Method (ADM) – набір ітеративних методик гнучкої розробки програмного забезпечення, в рамках яких робиться упор на формування вимог та рішень засобами співробітництва різних крос-функціональних команд.

— Dynamic Systems Development Method (DSDM) – ітеративний та інкрементний підхід, що базується на концепції швидкої розробки додатків – Rapid Application Development (RAD), упор в якому робиться на максимальне залучення кінцевого користувача до розробки програмного продукту.

— Essential Unified Process (EssUP) – підхід, що містить в собі методи ітеративної розробки програмного забезпечення, з упором на архітектуру продукту та напрацьовані практики команди. Ідея полягає в тому, що використовуються тільки ті практики та методи, які застосовні в конкретній ситуації. На основі вибраних методів та практик визначається цільовий процес. На відміну від RUP, де всі практики та методи взаємопов'язані, в даному випадку з'являється гнучкість та можливість виокремити із всього доступного об'єму саме необхідні елементи (методи та практики).

— Extreme programming (XP) – ідея екстремального програмування полягає в тому, щоб використовувати вже існуючі найкращі практики в області розробки програмного забезпечення, піднявши їх на новий (екстремальний) рівень. Наприклад, на відміну від звичайної практики, коли один програміст послідовно перевіряє написаний код за своїм колегою, в екстремальному програмуванні дана перевірка здійснюється паралельно, що збільшує швидкість випуску продукту, але й ризики також.

— Feature driven development (FDD) – основне обмеження, яке накладається в рамках даного підходу, це «кожна функція повинна бути реалізована не більше, ніж за два тижні».

— Getting Real (GR) – в рамках даного підходу виключено процедури функціональних специфікацій, що використовуються для веб-додатків. Розробка розпочинається від зворотного, спочатку розробляється інтерфейс та дизайн, а потім сама функціональність.

— OpenUP (OUP) – даний підхід визначає ітеративно-інкрементальний метод розробки програмного забезпечення. Розроблений на основі RUP. В рамках даного методу визначено життєвий цикл розробки (фаза запуску, фаза уточнення, фаза розробки та передачі замовнику). Завдяки визначеній етапності та контрольним точкам, підвищується ефективність контролю та моніторингу ходу реалізації проекту, як наслідок своєчасного прийняття рішень по проекту.

— lean software development – даний підхід засновано на концепції бережливого керування виробничим підприємством (lean production, lean manufacturing).

— Scrum – один із самих розповсюджених підходів гнучкої розробки програмного забезпечення, що визначає правила керування процесом розробки із застосуванням існуючих практик розробки. Упор здійснюється на включеність замовника у процес (можливість після кожного етапу змінювати чи уточняти вимоги до створюваного продукту), що дозволяє вчасно визначати відхилення та внести необхідні зміни.

У [5] виділяють такі переваги Agile:

1. Швидке виявлення неправильних підходів. Є багато документальних підтверджень тому, що в сфері технологій, чим пізніше виявляється невірно вибраний шлях, тим важче все виправити. Завдяки щоденному відслідковуванню робочого прогресу, Agile дозволяє швидко виявити помилки.

2. Швидке прийняття рішень. Більшість рішень приймається партнерами, що працюють в одному приміщенні. При виникненні будь-яких питань звичайна практика має на меті збір працівників в одному місці для їх обговорення. Дуже рідко виникає необхідність в проведенні офіційних зібрань, узгодження графіка яких може зайняти кілька днів.

3. Співпраця виливається в безліч переваг. З огляду на те, що технічні та бізнес-групи мають такі самі обов'язки, вони однаково зацікавлені в досягненні успіху. Технічні співробітники розуміють труднощі, які відчувають бізнес-підрозділи з поточним середовищем, в той час як бізнес-співробітники розуміють технічні труднощі розробки нової програми. При виникненні будь-яких проблем вони стають відомі всім учасникам групи, і дуже часто рішення знаходять люди, що працюють над зовсім іншими завданнями.

4. Зміни визнаються неминучими і вітаються. Зрозуміло, що на самому початку проекту ніхто не може точно уявити, як повинна працювати система. Багато каскадних проектів стикаються з «аналітичним паралічем» через тиск, пов'язаний з необхідністю розробки технічних вимог, перш ніж можна приступити безпосередньо до роботи. При гнучкій моделі розробка системи здійснюється в процесі декількох циклів і поправки вносяться на ходу.

5. Кінцевий продукт містить найбільш корисні функції. Так як бізнес-співробітники беруть безпосередню участь у розвитку продукту, вони ефективно визначають функції, які підвищують його цінність. Навпаки, визначення всіх вимог на початкових етапах проекту може привести до розробки непотрібних або обмежено корисних функцій.

6. Більш привабливе середовище для нового покоління. Сьогодні дуже популярні розмови про залучення покоління, народженого в кінці XX століття. Agile підходить для цього ідеально, так як молодим співробітників подобається динамічна робоча обстановка, що спонукає до співпраці.

7. Більш висока якість коду готового продукту. Проекти, засновані на гнучкій методології відрізняються суттєво меншою кількістю дефектів та помилок.

8. Бізнес-група відчуває більше задоволення від підсумкових результатів. В ході вивчення рівня задоволеності замовників виявилось, що проекти, засновані на гнучкій моделі розробки, набирали значно більше балів, ніж каскадні проекти. Власне, дуже рідко гнучкі проекти набирали менше п'яти балів за п'ятибальною шкалою.

9. Складання технічної документації займає менше часу і містить менше помилок. З огляду на те, що документація обмежується робочими продуктами, необхідними для виконання поставлених завдань (побажання користувачів, тестові сценарії і т. д.), то в ній відображається те, що було реалізовано, а не те, що могло бути заплановано. Можливості відстеження в процесі аудиту набагато ширші, так як узгоджуються всі дискретні функції, на противагу однократному твердженню кількох сотень сторінок документації. При традиційних підходах до розробки витрачається величезна кількість часу на складання документації, яка часто не оновлюється або не використовується. При каскадній моделі проміжні етапи часто пов'язані зі складанням документації, а не з фактично працюють кодом.

10. Більш просте обслуговування додатків. Відразу кілька розробників працюють над кожною частиною системи, що впливає безпосередньо на якість програмного продукту.

IT-експерт Боб Ронан (Bob Ronan) у [6] виокремлює такий ряд переваг, які слід врахувати топ-менеджерам IT-фірм, що займаються розробкою ПЗ при переході на Agile.

1. Технічний та бізнес-підрозділ спільно розташовується у відкритій зоні. Перші Agile-послідовники починали з ключових співробітників (менеджер проектів,

системні та бізнес-аналітики, розробники, тестувальники), але з часом процес охопив всіх працівників. Наприклад, з технологічної сторони сьогодні поширена практика об'єднання інженерів з аналізу та обробки даних, адміністраторів баз даних і персоналу з технологічних операцій в одному підрозділі, навіть якщо вони знаходяться там тимчасово.

2. Сценарії тестування розробляються до початку етапу програмування. У разі розробки на основі тестів розробникам надається можливість відразу перевірити працездатність свого коду. Незважаючи на той факт, що в реальності багато команд намагаються реалізувати цей крок до початку етапу програмування, найважливіше – це складання сценаріїв тестування, особливо сценаріїв модульного тестування, які часто не використовуються взагалі в каскадній моделі.

3. Ранкові збори проводяться щодня. В ході таких зборів команда збирається в коло і кожен учасник розповідає про те, що він зробив вчора, що планує зробити сьогодні, і з якими проблемами він зіткнувся. Всі питання, які потребують більше часу, ніж коротке обговорення, документуються і розглядаються пізніше. При цьому, дуже важливо щоб всі стояли, так як це підкреслює необхідність бути коротким.

4. Процес складається з робочих циклів тривалості від однієї до чотирьох тижнів, які називають «спринти». На виході кожного циклу виходить робочий код. Двотижневі спринти забезпечують збалансовану можливість створити щось справді цінне при збереженні почуття терміновості. Кожен цикл включає в себе:

— «ігрове планування», що дозволяє вирішити, які функції будуть вбудовані в поточну версію, виходячи з пріоритетного списку попередньо зібраних «побажань користувачів». Побажання користувачів, являють собою високорівневий опис бізнес-завдань, які повинен вирішувати додаток. Спочатку вони розробляються під час робочої наради, на якій присутні всі співробітники технічних і бізнес-підрозділів, а потім поступово оновлюються, щоб гарантувати, що позиції з найбільшою комерційною цінністю знаходяться зверху;

— розробку макетів екранів та тестові сценарії;

— написання коду і тестування взаємодії компонентів системи. Слід відзначити той факт, що часто код приводиться до загальної основи, дозволяючи всім працювати з однією і тією ж версією, що дозволяє швидко виявляти проблеми інтеграції;

— демонстрацію робочого коду робочій групі та керівництву технічного або бізнес-підрозділу;

— ретроспективне обговорення того, що в рамках циклу пройшло добре, а що слід змінити в майбутньому. Найкраща практика полягає в розробці побажань користувачів для обліку всіх можливостей, окреслених під час ретроспективного огляду, та їх опрацювання в рамках наступного циклу, так як це підкріплює концепцію безперервного вдосконалення.

Разом з тим виділяють й недоліки при переході на Agile [1,7,8]:

1. Нерозуміння ролі керівництва при впровадженні методології. Перехід на гнучкі методології має на меті кардинальну зміну задач та методів роботи керівників. Цей перехід можна сформулювати як перехід від управління до напрямку, від наказів та вказівок до рекомендацій. Перша помилка, що допускається при такому переході – підсвідоме прагнення зберегти за собою владу, що веде до все того ж управління. Друга не менш серйозна помилка прямо протилежна – неправильне розуміння цього переходу може призвести до усунення менеджерів від керівних функцій, коли керівник перестає давати вказівки, але так і не починає давати рекомендації. Роль керівника при цьому зводиться до чисто формальних, секретарських функцій. Причиною цього може бути і неготовність керівників давати аргументацію своєї думки (в старій моделі їм цього часто робити не доводилося) або боязнь невиконання висловлених рекомендацій.

2. Побудова «система», що не володіє необхідною гнучкістю. Відсутність досвіду роботи за новою методологією веде до того, що новий процес впроваджується за інструкціями, буква до букви, що веде до негнучкості і бюрократизації.

3. Початок впровадження не з «основами». При переході на нові методології керівництво переслідує конкретні вигоди, які повинна забезпечити методологія: висока продуктивність, якість і т. п. При цьому деякі практики нового процесу більш очевидно ведуть до цих вигід, а деякі – зовсім не очевидно. У зв'язку з цим виникає спокуса запровадити спочатку більш «вигідні» практики, а потім вже інші. При цьому не враховується,

що деякі практики є базовими по відношенню до інших, і впровадженню другого без першого неможливо.

4. Змінюються робочі місця, але не змінюються звички. Проголошення нових правил і дотримання цих правил – це принципово різні речі. Недостатнє розуміння нової ідеї всіма членами команди або слабке усвідомлення вигод від переходу на новий процес, слабка мотивація, відсутність перехідного періоду з явно вираженими послабленнями в графіку робіт і кількості завдань – це далеко не повний список помилок, здатних привести до того, що новий процес може дотримуватися лише формально, а в дійсності саботуватися членами команди.

5. Всі вимірювати (збирати дані), але ні на що не реагувати. Нескінченний аналіз ситуації, замість безперервних поліпшень. Більшість гнучких методологій мають на увазі збір даних і обговорення помилок, допущених на попередньому етапі, перед початком наступного. Поширені помилки в цій сфері – збір даних без їх подальшого аналізу або невірна, занадто поверхнева інтерпретація зібраних даних.

6. Обходитися без підтримки. Відсутність досвіду роботи команди за новою методологією і впровадження по букві інструкції містить багато помилок, невірні інтерпретації і нерозуміння. Тільки участь в процесі переходу досвідченого інструктора, який має безпосередній досвід роботи за новим процесом, може позбавити команду від безлічі невірних поворотів і тупиків або в окремих випадках навіть від побудови абсолютно невірної методології.

Взагалі ж, Agile погано описує процеси управління вимогами, можна сказати, що таке поняття просто відсутнє, тому гнучка методологія розробки не має на увазі під собою довгострокового планування (планування здійснюється на короткострокову перспективу), як наслідок пропущений крок формування плану розвитку продукту або іншими словами дорожньої карти продукту [9,10]. Оскільки планування короткострокове (на найближчу ітерацію розробки), а замовник по закінченню кожної ітерації приймає продукт і виставляє нові вимоги, сам продукт може помінятися в коренях, а виставлені нові вимоги часто суперечить структурі та архітектури продукту, що вже поставляється клієнтам. По великому рахунку, якщо замовника не до кінця розуміє, що хоче побачити в результаті (кінцевий продукт), а розуміння приходить під час розробки (це трапляється в 90% випадках), процес розробки перетворюється у формалізовану та легалізовану бюрократію, коли продукт допрацьовується нескінченно, поки не закінчуються гроші, або замовник не переключає увагу на інший продукт. Справедливості заради, слід зазначити, що замовник знає на що йде і сам вирішує, платити за розробку продукту чи ні, оскільки команди розробників просто виконують вимоги замовника. Однак, реально, в роботі це призводить до хаосу та зриву термінів, що породжує нові вимоги, які змінюють не в кращий бік продукт. Більш того, знижується якість продукту, що розробляється, тому що Agile визначає підхід до розробки, в рамках якого необхідно швидке реагування на зміни в найбільш простий і швидкий спосіб. Код пишеться не дотримуючись вимог платформи, на якій розробляється продукт, з'являється безліч обхідних рішень та дефекти, а така конструкція не дуже стійка та небезпечна, зростає обурення клієнтів від частих збоїв в роботі програмного забезпечення. Бізнес на виході отримує втрати, падає якість планування. Тому деякі з експертів асоціюють Agile більше з підходом щодо вдосконалення вже готового продукту, ніж розробки нового.

ВИСНОВКИ

Отже, проаналізувавши ряд наукових робіт, що стосуються використання гнучких методологій розробки програмного забезпечення можна дійти висновку про те, що не зважаючи на велику популярність Agile, існують й протиріччя у їх використанні з наступних причин:

1. Переважна більшість керівників не люблять ризикувати, а зміна процесу розробки пов’язана із різними змінами.

2. Гнучка методологія досить часто не спрацьовує, зокрема коли йде мова про життя та здоров’я людини (медицина, будівництво житла та доріг, охорона безпеки і т.д.). В зазначених випадках каскадна модель, яка є більш строгішою та формалізованішою, має свої переваги.

Разом з тим, можна зробити висновок про те, що аби Agile-метод запрацював, необхідне існування групи, яка буде мати можливість та бажання його випробувати. Також цій групі необхідно надати повну

свободу дій у виборі методів та засобів, що підходять для конкретної організації.

Отже, Agile найбільше підходить для проектів із «відкритим кінцем», наприклад, запуск інтернет-сервісів, розробка комп'ютерних ігор, операційних систем. Однак, гнучкість може приводити до втрати фокусу та втрати передбачуваності. Дуже важливо відокремлювати помилки застосування гнучкого підходу від недоліків самої методології. Перш, ніж будуть реалізовані всі переваги підходу, потрібно буде деякий час на адаптацію до реалій конкретного бізнесу.

ЛІТЕРАТУРА

1. Karpov D.V. Hybkaya metodolohyya razrabotki prohrammnoho obespecheniya // Vestnik Nizhehorodskoho universiteta im. N.I. Lobachevskoho. – Nyzhehorodsksy hosuniversitet im. N. I. Lobachevskoho, 2011. – S. 227-230.
2. Kniberh Kh. Scrum i XP: Zametki s peredovoy.C4Media, 2007. – 94 c.
3. Schwaber Ken. Agile Project Management With. Prentice Hall PTR, 2004. – 163 c.
4. Schwaber Ken. Agile Software Development With Scrum. Prentice Hall PTR, 2002. – 158 c.
5. Schwaber Ken, Sutherland J. Scrum Development Process, in OOPSLA Business Object Design and Implementation Workshop. – Springer, London, 1997.
6. Ronan Вов. 10 reasons you should be using agile – Mode of access: World Wide Web: http://www.cio.com/article/3078178/agile-development/10-reasons-you-should-be-using-agile.html (viewed on April 13, 2017). – Title from the screen.
7. Putu Adi Guna Permana. Scrum Method Implementation in a Software Development Project Management // (IJACSA) International Journal of Advanced Computer Science and Applications. – 2015. – Vol. 6. № 9. – P. 199-205.
8. Thomas D., Hansson D. H. Agile Web Development with Rails. 2007. P. 17-64. 2.
9. Cohn M. Succeeding with Agile: Software Development Using Scrum. 2008. P. 53-81.
10. Vol'fson B. Hybkye metodolohyy razrabotky. – M.: Eksmo, 2013. – 112 c.

Рецензент: *д.т.н., доц. Бойко Ю.М., Начальник НДЧ*