СИСТЕМИ УПРАВЛІННЯ

УДК 004.05

Барибін О. І.

МОДЕЛЬ ЯКОСТІ ДЛЯ ГНУЧКОЇ МЕТОДОЛОГІЇ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ SCRUM

У статті розглянуто проблему забезпечення якості програмного забезпечення та процесу розроблення програмного забезпечення під час використання гнучкої методології Scrum. На основі аналізування основоположних елементів цієї метології побудовано модель якості з використанням підходу GQM (goal, question, metric, або ціль, питання, метрика). Три аспекти якості – функційна якість, структурна якість та якість процесу – запропоновано як цілі, для яких сформовано питання, в рамках яких запропоновано набір відповідних метрик: рейтинг покращення, рівень видалення дефектів, ефективність видалення дефектів, коефіцієнт часу покращення, швидкість, час виконання, час циклу.

Ключові слова: *розроблення програмного забезпечення, гнучкі методології, Scrum, модель якості, метрики.*

Постановка проблеми в загальному вигляді. Розроблення сучасного програмного забезпечення потребує виконання низки складних технологічних процесів, до яких залучено безліч людей, що виконують різні види робіт. Організація цього процесу – важливе завдання, якісне вирішення якого дає змогу значно підвищити ефективність і продуктивність.

Вибір правильної методології розроблення програмного забезпечення суттєво впливає на його якість та успішність проекту в цілому. Наразі відомо дві великі групи технологічних підходів до розроблення програмного забезпечення, в рамках яких є різні методології [1-4]:

• суворі (класичні, передбачувані, жорсткі) підходи,

• гнучкі (адаптивні, легкі) підходи.

В умовах значної конкуренції на вільному ринку значного розвитку набули гнучкі (адаптивні) методології, під якими розуміють так звані легкі (полегшені), швидкі, гнучкі підходи, що адаптуються до зміни вимог до програмного забезпечення. Така велика кількість епітетів стосується єдиного англомовного терміна«agile», що в перекладі означає «гнучкий, моторний». Однак моторним зазвичай жодний технологічний процес не називають, а прикметник «гнучкий» якраз і стосується підходу або методології в цілому.

Основу методів гнучкого розроблення (agile development) відображено в маніфесті альянсу гнучкого розроблення (agile manifesto) [5], а характерною рисою гнучких методологій розроблення програмного забезпечення є пріоритет людського чинника, програмного коду, готовності до змін і спілкування як усередині команди, так і з замовником над усіма іншими компонентами проекту, зокрема повноти комплекту документації й дотримання плану [6–8].

За таких умов забезпечення якості процесу розроблення програмного забезпечення й самого програмного забезпечення стає складною та актуальною проблемою, яку може бути вирішено використанням чітко визначених методів та шкал вимірювання характеристик якості або так званих метрик [9].

Результати досліджень компанії Agile Survey (співтовариство Agile) щодо популярності

гнучких методологій [10] засвідчили, що найбільш використовуваною гнучкою методологією у світі є Scrum. Зауважимо, що термін Scrum асимільовано в українську мову без перекладу.

Відповідно основною **метою праці** є структуризація й актуалізація метрик якості розроблення програмного забезпечення стосовно методології Scrum.

У рамках цієї мети можна визначити такі основні **завдання праці**:

• аналізування процесу розроблення програмного забезпечення відповідно до методології Scrum та виділення найважливіших його елементів;

• обґрунтування й формування моделі якості та відповідних метрик для обраної методології.

Аналіз останніх досліджень і публікацій. Проведення вимірювань у рамках процесу розроблення будь-якого програмного забезпечення доцільно виконувати в рамках сформульованої моделі якості, в якій обов'язково має бути визначено так звані метрики, які являють собою визначений метод і шкалу вимірювання [9].

Є досить велика кількість підходів до формування моделі якості, з якими можна ознайомитися в [1, 3, 4, 6, 7], проте ці класичні моделі не можуть бути прямо застосовні до гнучких методологій через те, що їх, по-перше, розробляли для класичних методів розроблення програмного забезпечення, а, по-друге, в таких моделях велика кількість характеристик (атрибутів) і підхарактеристик призведе в разі їх використання до відходу від основних принципів гнучного розроблення програмного забезпечення. У сучасних літературних джерелах, наприклад [13, 14], для гнучких методологій пропонують моделі якості, які намагаються врахувати практично всі характеристики класичних моделей, що унеможливлює їх практичне використання.

У праці пропонується використання GQM (goal, question, metric, або ціль, питання, метрика) підходу, який визначенив Базилі [15], через його простоту до впровадження. В рамках цього підходу треба зосередити вимірювання якості відповідно певних цілей ефективності. Ці цілі ефективності також називають цілями вимірювань. Для кожної цілі вимірювання ставлять одне чи більше запитань, які характеризують те, як можна досягти цілі. Щоб оцінити досягнення цілі, треба відповісти на ці запитання. Для цього формують метрики, які надають нам інформацію, необхідну для відповіді на запитання. Інакше кажучи, метрики кількісно визначають чинник, який впливає на досягнення цілей вимірювання. Для підтримки інтерпретації конкретної метрики Базилі рекомендує створити модель інтерпретації та надавати її людям, які беруть участь у програмі вимірювання [15].

Загальні положення Scrum

Щоб обґрунтовано сформувати цілі, питання та метрики в рамках моделі GQM, розглянемо основоположні елементи методології, що розглядають.

Як і більшість гнучких методологій, Scrum націлено на мінімізацію ризиків зведенням розроблення до серії коротких циклів, які називають ітераціями та які зазвичай тривають кілька тижнів (рис. 1). Кожна ітерація, яка у Scrum має назву спринт, сама по собі має вигляд програмного проекту в мініатюрі й охоплює всі завдання, необхідні для розроблення готового до використання програмного забезпечення: планування, аналізування вимог, проектування, кодування, тестування й документування. Хоча окрема ітерація зазвичай не достатня для випуску нової версії продукту, мають на увазі, що гнучкий програмний проект готовий до випуску наприкінці кожної ітерації. Після закінчення кожної ітерації команда виконує переоцінку пріоритетів розроблення.

Процес розроблення програмного забезпечення згідно зі Scrum досить докладно описано в [2, 11, 12], тому зупинимося лише на найважливіших аспектах. Зокрема, суттєву роль у забезпеченні якості процесу розроблення й самого програмного забезпечення відіграють так звані ролі (учасники) й артефакти Scrum.

Учасниками Scrum є команда розробників, власник продукту, який представляє

замовника інформаційної системи, і Scrum-майстер, який виконує роботу керівника й консультанта.

Команда складається з 3–9 осіб, які виконують роботу (аналізують, виконують дизайн, пишуть код, тестують, готують документацію тощо). У Scrum команда є самокерованою та самоорганізованою.

Власник продукту є єдиною особою в команді, яка відповідає за журнал вимог до продукту (Журнал продукту або Product Backlog). У Product Backlog міститься перелік робіт, які треба виконати під час спринту, впорядкованих за їх важливістю (пріоритетом).

Scrum-майстер – це керівник, відповідальний за спроможність команди виконати поставлені цілі й подолати складнощі, що виникають у процесі розроблення проекту.

Послідовність розроблення інформаційного продукту і взаємодія власника продукту, Scrum-майстра та команди розробників показано на рисунку 1.

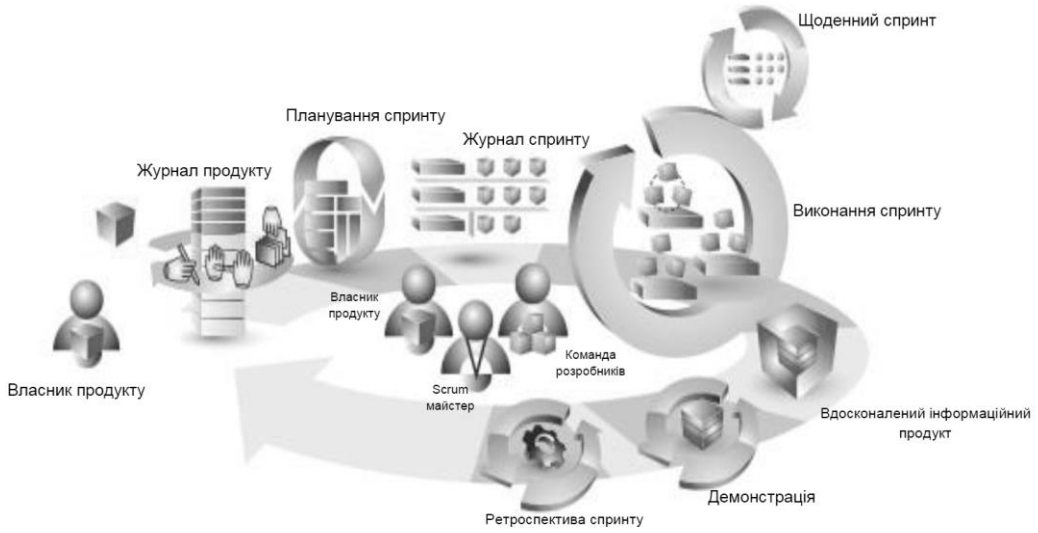

Рисунок 1. Розроблення інформаційного продукту, взаємодія учасників Scrum [11]

Журнал продукту (Product backlog) – це основа Scrum, який являє собою список вимог, завдань або, як їх називать у рамках методології, «історій користувача» або User Stories, впорядкованих за ступенем важливості. При цьому всі історії користувача описано так, щоб замовник інформаційного продукту правильно їх зрозумів.

Опис кожної історії містить ID (унікальний ідентифікатор), назву (короткий опис історії), важливість (ступінь важливості цього завдання, на думку власника продукту), попередню оцінку (початкова оцінка обсягу робіт, необхідного для реалізації історії порівняно з іншими історіями, яку вимірюють у так званих «точках» історії або story point'-ах та яка приблизно відповідає кількості ідеальних людино-днів).

Формулювання цілей моделі якості з використанням GQM-підходу. Аналізування метології Scrum дає можливість запропонувати три аспекти, які роблять вклад у якість програмного забезпечення: функційна якість, структурна якість та якість процесу.

Характеристики функційної й структурної якості може бути знайдено в ISO / IEC 25010 [9]. Функційну придатність визначають як ступінь, у якому продукт або система забезпечує функції, що задовольняють заявлені потреби в разі використання за певних умов, і ремонтопридатність визначають як ступінь ефективності, з яким продукт або систему може бути змінено за допомогою відповідного персоналу.

Вибрані характеристики є найважливішими для оцінювання, якщо розроблення програмного забезпечення генерує продукт, який задовольняє клієнта, спираючись на якість, що дає змогу прискорити процес розроблення й створює життєздатну платформу для майбутніх удосконалень і забезпечує доставлення в межах обмежень (сфера, розклад і вартість). Іншим важливим фактом є те, що обрані характеристики може бути оцінено без особливих додаткових зусиль у гнучкому середовищі розроблення.

Формулювання питань. Дотримуючися підходу GQM, сформулювано питання, які допомагають оцінити цілі з попереднього розділу.

Функційна якість. Для оцінювання функційної якості мети використаємо чинники функційної якості й зосередимося на найважливіших чинниках, які відповідають зазначеним вимогам. Ці чинники пов'язані з підхарактеристиками функційної придатності в ISO / IEC 25010, як показано в такому списку:

• функційна повнота – ступінь, у якому набір функцій охоплює всі зазначені завдання й цілі користувача;

• функційна коректність – ступінь, у якому продукт або система забезпечує правильні результати з необхідним ступенем точності.

Оцінка, за умови реалізації історій користувачів, пов'язана з функційною завершеністю. Питання, які пропонують для функційної якості, наведено в таблиці 1.

Таблиця 1

Питання функційної якості

П1.1 Скільки історій користувачів доставлено в спринті?	
Метрика	R_I – рейтинг покращення
Інтерпретація	Чим вища кількість історій користувачів, реалізованих і доставлених, тим вища функційна якість
П1.2 Скільки дефектів видалено в спринті?	
Метрика	R_{DR} – рівень видалення дефектів
Інтерпретація	Чим вища кількість дефектів, які є виправленими, тим вищий рівень функційної якості

Структурна якість. Структурна якість пов'язана з внутрішньою структурою програмного забезпечення. Щоб оцінити структурну якість, треба проаналізувати ремонтопридатність вихідного коду програмного забезпечення. В ISO/IEC 25010 ремонтопридатність визначають як ступінь ефективності, з яким продукт або систему може бути змінено за допомогою передбаченого персоналу та перелічено кілька підхарактеристик, які сприяють ремонтопридатності програмного забезпечення. Найбільший вплив мають:

• здатність до аналізування – ступінь ефективності, за яким можна оцінити вплив на продукт або систему передбачуваної зміни однієї чи кількох його частин, або діагностувати продукт щодо недоліків або причин відмов або для ідентифікації частин, які має бути змінено;

• здатність до модифікацій – ступінь, у якому продукт або система може бути ефективно модифіковано без внесення дефектів або погіршення якості наявного продукту;

• здатність до тестування – ступінь ефективності, за яким критерії тестування може бути встановлено для системи, продукту або компонентів і за яким тестування може бути виконано для визначення того, чи виконано ці критерії.

Інші аспекти програмного забезпечення ремонтопридатності в цій праці не використовують, щоб не перевантажувати процес вимірювання. Проте додаткові

характеристики ремонтопридатності може бути додано до моделі, якщо це буде визнано доцільним. Питання, які пропонують для структурної якості, наведено в таблиці 2.

Таблиця 2

Питання структурної якості

П2. Що таке ремонтопридатність програмних продуктів?	
Метрика	Ремонтопридатність програмного забезпечення
Інтерпретація	Чим вища ремонтопридатність програмного забезпечення, тим вища структурна якість програмного забезпечення

Якість процесу. Оцінку якості процесу зосереджено на відповідності термінів поставлення без оцінки відповідності бюджетів. Зазвичай на рівні бюджет планів окремих функцій не формується. Проте показники, визначені для оцінки відповідності термінів поставлення, може бути використано для оцінювання ефективності розроблення програмного забезпечення щодо витрачених розробниками зусиль. Завдяки підвищенню ефективності більше функцій може бути реалізовано з тією самою кількістю зусиль розробників.

Питання, що випливають з якості процесу, можна сформулювати так (табл. 3).

Таблиця 3

Питання якості процесу

П3.1 Яка потужність процесу розроблення програмного забезпечення?	
Метрика	V – швидкість
Інтерпретація	Чим вища потужність процесу розроблення програмного забезпечення, тим вища якість процесу
П3.2 Що таке час виконання?	
Метрика	T_L – час виконання
Інтерпретація	Чим менший час виконання процесу розроблення програмного забезпечення, тим вища якість процесу
П3.3 Що таке час циклу?	
Метрика	T_C – час циклу
Інтерпретація	Чим менший час циклу процесу розроблення програмного забезпечення, тим вища якість процесу
П3.4 Скільки зусиль витрачено на поліпшення щодо загального розширення зусиль?	
Метрика	R_{TE} – коефіцієнт часу покращення
Інтерпретація	Чим вищий коефіцієнт часу покращення, тим вища якість процесу
П3.5 Скільки помилок виправлено до доставлення?	
Метрика	E_{DR} – ефективність видалення дефектів
Інтерпретація	Чим вища кількість дефектів, виявлених і виправлених до поставлення програмного забезпечення, тим вище поліпшення функційної якості

Формулювання метрик

Показники, обрані як індикатори якості програмного забезпечення, визначено й описано нижче.

Рейтинг покращення. Рейтинг покращення є мірою пропускної здатності процесу. Пропускна здатність процесу – це кількість модулів, задіяних у процесі протягом певного

періоду часу. Під час розроблення програмного забезпечення історії користувачів розглядають як вхідні дані в процесі розроблення. Історія користувача проходить крізь різні етапи процесу розроблення програмного забезпечення. Результатом процесу розроблення є реалізація історії користувача у вигляді робочого коду. В Scrum історії користувачів формують на конкретний спринт. Метрику рейтингу покращення визначають так:

$$R_I = \frac{US_S}{S_{WD}},$$

де US_S – кількість історій користувачів, впроваджених у спринті,
S_{WD} – кількість робочих днів у спринті.

Рівень видалення дефектів. Метрику «Рівень видалення дефектів» використовують для аналізування кількості дефектів програмного забезпечення, які успішно видалено з програмних продуктів у конкретному спринті. Тільки помилки, знайдені/створені до спринту, враховують у цій метриці. Цей розрахунок тісно поєднано з метрикою «Ефективністю видалення дефектів». Формула для метрики «Рівня видалення дефектів» має такий вигляд:

$$R_{DR} = \frac{D_{BS}}{S_{WD}},$$

де D_{BS} – кількість дефектів з датою створення дефекту ранішою за дату початку спринту.

Ефективність видалення дефектів. Ефективність видалення дефектів обчислюють як відношення кількості дефектів, виявлених у спринті, до загальної кількості дефектів, вправлених у спринт. Формула має такий вигляд:

$$E_{DR} = \frac{D_D}{S_{ST}},$$

де D_D – кількість виявлених і виправлених дефектів під час спринту до доставлення програмного забезпечення замовникові;
D_{ST} – загальна кількість виправлених дефектів під час спринту до доставлення програмного забезпечення замовникові.

Коефіцієнт часу покращення. Метрика «Коефіцієнт часу покращення» розглядає історії користувача й дефекти. Для порівняння в метриці «Коефіцієнт покращення» розглядають тільки питання типу історії користувача і дефектів. Коефіцієнт часу покращення обчислюють так:

$$R_{TE} = \frac{T_{US}}{T_C},$$

де T_{US} – час, витрачений на впровадження історій користувачів;
T_C – час, витрачений на впровадження історій користувачів та видалення дефектів.

Ремонтопридатність програмного забезпечення. Heitlager та інші розробили модель якості SIG для оцінювання ремонтопридатності програмного продукту [16]. Ця модель використовує визначення ремонтопридатності в ISO / IEC 9126 для оцінювання ремонтопридатності програмного продукту. Пізніше цю модель скориговано, коли

ISO / IEC 9126 замінено ISO / IEC 25010. В ISO / IEC 9126 характеристика ремонтопридатністі мала такі підхарактеристики, як здатність до аналізування, здатність до змін, стабільність і здатність до тестування. Модель якості SIG визначає властивості програмного коду, які відображаються на підхарактеристики ремонтопридатності. Властивостями програмного коду, зіставленими з підхарактеристиками ISO / IEC 9126 в оригінальній моделі якості [16] та які може бути виміряно за допомогою метрик програмного коду, можуть бути, наприклад, метрика «Кількість рядків коду», яку використовують для вимірювання об'єму, або метрика «Цикломатична складність», яку використовують для вимірювання складності модуля. Оцінки для підхарактеристик ремонтопридатності агрегують з використанням усереднення.

Швидкість. У середовищі Scrum точки історії користувача прив'язані до історій користувача, які являють собою зусилля, необхідні для виконання конкретної історії користувача. Для метрики «Потужність процесу розроблення» програмного забезпечення з точки зору команди розробників розглянемо точки історії як одиниці продукції. Як одиниці часу розглянемо довжину спринту, в робочих днях. Обчислення метрики мають такий вигляд:

$$V = \frac{SP_S}{S_{WD}},$$

де SP_S – кількість точок історій впроваджених історій користувача у спринті.

Такий спосіб вимірювання швидкості спрощує планування спринту.

Час виконання. Метрика «Час виконання» бере свій початок в Kanban, який являє собою спосіб для управління потоком продукції. Його введено в Toyota Motor Corporation з метою підвищення ефективності виробничого процесу. Концепцію Kanban скориговано Андерсоном для використання в розробленні програмного забезпечення. Відповідно до Андерсона визначаємо метрику «Час виконання» в розробленні програмного забезпечення як період часу з ідентифікації історії користувача до виконання історії у вигляді робочого програмного коду. Реалізація історії користувача з більшою кількістю точок історії займає більше часу, ніж історії користувача з меншою кількістю точок історії. Ці зусилля враховують у метриці «Час виконання» для того, щоб домогтися кращої порівнянності між обраними історіями користувача. Отже, час виконання обчисюють так:

$$T_L = \frac{D_{USC} - D_{USB}}{SP_{US}},$$

де D_{USB} – дата відкриття історії користувача;

D_{USC} – дата завершення історії користувача;

SP_{US} – кількість точок історій в історії користувача.

Час циклу. На відміну від часу виконання, час циклу стосується тільки частини життєвого циклу. Час циклу охоплює період часу в життєвому циклі історії користувача зі стану «Відкрита» доти, доки історія користувача не отримує стану «Готова до розгортання». Стан «Готова для розгортання» означає момент часу, коли реалізацію історії користувача закінчено, й усі тести пройдено успішно. Команда розробників несе відповідальність тільки за цей конкретний період часу, оскільки рішення про доставлення/розгортання продукту лежить поза межами їх відповідальності.

Так само, як і для метрики «Час виконання», використовуємо точки історії для нормалізації, щоб отримати можливість порівнювати історії користувачів. Отже, формула

для обчислення часу циклу має такий вигляд:

$$T_C = \frac{D_{USD} - D_{USB}}{SP_{US}},$$

де D_{USD} – дата готовності історії користувача до розгортання.

Висновки.

Швидкий розвиток та широке використання гнучких методологій під час розроблення програмного забезпечення в цілому та методології Scrum обумовлює необхідність розроблення спрощених моделей якості як для самого програмного забезпечення, так і для процесу його розроблення. Незважаючи на те, що загалом проблема формулювання моделей якості не є новою і для класичних методологій її може бути вирішено прямим використанням відповідних міжнародних стандартів, для Scrum такий підхід не є оптимальним через можливе переобтяження процесу вимірювання і втрати гнучкості. Для побудови моделі якості Scrum обрано підхід GQM (goal, question, metric, або ціль, питання, метрика) через його простоту до впровадження. На основі детального розгляду методології Scrum такі три аспекти якості, як функційна якість, структурна якість та якість процесу, запропоновано як цілі, для яких сформовано питання, в рамках яких запропоновано відповідні метрики: рейтинг покращення, рівень видалення дефектів, ефективність видалення дефектів, коефіцієнт часу покращення, швидкість, час виконання, час циклу.

ЛІТЕРАТУРА

1. Claude Y. L., April. A. Software Quality Assurance. *Wiley-IEEE Computer Society*. Inc. 616. Hoboken. 2018.
2. Андреев А. Е., Кирносенко С. И. Адаптивные технологии разработки программного обеспечения : *Учеб. пособие. – ВолгГТУ. – Волгоград,* 2015. – С. 96.
3. Бахтизин В. В., Глухова Л. А. Метрология, стандартизация и сертификация в информационных технологиях : *Учеб. пособие. – В 2 ч. Ч.1. – Минск, БГУИР*, 2016. – С. 140.
4. Бахтизин В. В., Глухова Л. А. Метрология, стандартизация и сертификация в информационных технологиях : *Учеб. пособие. – В 2 ч. Ч.2. – Минск, БГУИР*, 2016. – С. 141–343.
5. Мартин Р., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C#.– *СПб. : Символ-Плюс*, 2011. – С. 768.
6. Walkinshaw Neil Software Quality Assurance. Consistency in the Face of Complexity and Change. *Springer International Publishing AG*. 2017. P. 186.
7. Андон Ф. И., Коваль Г. И., Коротун Т. М., Лаврищева Е. М., Суслов В. Ю. Основы инженерии качества программных систем. – *Академпериодика*. – *2-е изд*. – 2007. – С. 672.
8. Коцюба И. Ю., Чунаев А. В., Шиков А. Н. Методы оценки и измерения характеристик информационных систем : *Учеб. пособие. – СПб. : Университет ИТМО,* 2015. – С. 264.
9. ДСТУ ISO/IEC 25010:2016 Інженерія систем і програмних засобів. Вимоги до якості систем і програмних засобів та її оцінювання (SQuaRE). Моделі якості системи та програмних засобів (ISO/IEC 25010:2011, IDT).
10. Opens 12th Annual State of Agile™ Survey. URL: https://content.cdntwrk.com/files/aT05NjM2NTEmdj0yJmlzc3VlTmFtZT12ZXJzaW9ub25lLTEydGgtYW5udWFsLXN0YXRlLW9mLWFnaWxlLXJlcG9ydCZjbWQ9ZCZzaWc9YTFlY2RlYzYxYzAxNDFkZDVmMzc4MWQ2YzYxYjYxYTc%253D.
11. Демиденко М. А. Управління проектами інформатизації за методологією SCRUM : *Навч. посіб*. – *Нац. гірн. ун-т*. – 2016. – С. 80.
12. Dominik, Maximini. The Scrum Culture. Introducing Agile Methods in Organizations. *Second Edition Springer International Publishing AG 2015*. 2018. P. 3 – 23.
13. Usman M. Malik M. Haseeb Nasir A. J. An Efficient Objective Quality Model for Agile Application

Developmen International Journal of Computer Applications.Volume 85, January 2014. P. 19-24.
14. Tabassum A., Shahid N. B., Aneesa R. A., Iqra M., Imtiaz A. Optimized Quality Model for Agile Development: Extreme Programming (XP) as a Case Scenario. *International Journal of Advanced Computer Science and Applications*. 2017. Vol. 8, No. 4. P. 392 – 400.
15. Basili V.R., Lindvall M., Regardie M., Seaman C., Heidrich J., Munch J., Rombach D., Trendowicz A. Linking software development and business strategy through measurement. *Computer*. 43(4):57 2010.
16. Heitlager I., Kuipers T., Visser J. A Practical Model for Measuring Maintainability. *In International Conference on the Quality of Information and Communications Technology*. – 2007. – С. 30–39.

Барыбин А. И.

МОДЕЛЬ КАЧЕСТВА ДЛЯ ГИБКОЙ МЕТОДОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ SCRUM

В статье рассмотрена проблема обеспечения качества программного обеспечения и процесса разработки программного обеспечения при использовании гибкой методологии Scrum. На основе анализа основных элементов этой метологии построена модель качества с использованием подхода GQM (goal, question, metric, или цель, вопрос, метрика). Три аспекта качества – функциональное качество, структурное качество и качество процесса – были предложены в качестве целей, для которых сформированы вопросы, в рамках которых предложен набор соответствующих метрик: рейтинг улучшения, уровень удаления дефектов, эффективность удаления дефектов, коэффициент времени улучшения, скорость, время выполнения, время цикла.

Ключевые слова: *разработка программного обеспечения, гибкие методологии, Scrum, модель качества, метрики.*

O. Barybin

AGILE SOFTWARE DEVELOPMENT METHODOLOGY SCRUM QUALITY MODEL

The article considers the problem of quality assurance ащк software and software development process when using the Scrum agile methodology. Based on the analysis of the crucial elements of this methodology, a quality model was constructed using the GQM approach (goal, question, metric). Three aspects of quality – functional quality, structural quality, and quality of the process were proposed as goals for which the questions were formulated in which a set of relevant metrics was proposed: enhancement rate, defect removal rate, defect removal efectiveness, enhancement time ratio, velocity, lead time, cycle time.

Key words: *software development, agile methodology, Scrum, quality model, metrics.*

Рецензент: Меркулова К.В., канд.техн.наук, додент кафедри компютерних технологій, Донецікий національний університет ім. Василя Стуса, м. Вінниця