МАТЕРІАЛИ МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ СТУДЕНТІВ І АСПІРАНТІВ «ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 2013»

УДК 004.412:004.415.53(043.2)

Дишлевий О.П., Драпушко Ю.В.
Національний авіаційний університет

ЗАСІБ МОНІТОРИНГУ ТА АНАЛІЗ МЕТРИК ДЕФЕКТІВ ЯКОСТІ ПРОГРАМНОГО КОДУ

Стаття присвячена підбору метрик для покращення та спрощення процесу оцінки якості програмного забезпечення. Проводився аналіз метрик дефектів якості програмного коду. В процесі роботи була розроблена методика оцінки якості програмного коду, що базується на метриках дефектів якості програмного коду, а саме щільність дефектів.

Статья посвящена подбору метрик для улучшения и упрощения процесса оценки качества программного обеспечения. Проводился анализ метрик дефектов качества программного кода. В процессе работы была разработана методика оценки качества программного кода, основанный на метриках дефектов качества программного кода, а именно плотность дефектов.

The article is devoted to the selection of metrics to improve and simplify the process of assessing the quality of software. An analysis of defects of quality metrics software. In the process, developed a technique for evaluating the quality of code, based on the metrics of code quality defects, such as the density of defects.

Ключові слова: *тестування програмного забезпечення, метрика, якість, дефект якості програмного коду, щільність дефектів*

Вступ

Одночасно зі збільшенням обсягів використання апаратного забезпечення, зростає роль і програмного забезпечення (ПЗ). Зростаюча конкуренція, мінливе зовнішнє середовище і необхідність постійно впроваджувати нові продукти та послуги для клієнтів спонукають компанії пришвидшувати темпи автоматизації та використовувати різноманітне за походженням і якістю виконання ПЗ [1]. Водночас все актуальнішою постає проблема ризиків, які виникають унаслідок неякісного ПЗ. Мова йде як про цілеспрямовані атаки злочинного походження, так і збої в роботі ПЗ, які виникають внаслідок ненавмисних помилок у програмному коді чи є наслідком інших недоліків при розробці ПЗ [2]. Таким чином, існує актуальна задача оцінки якості програмного коду для подальшого його вдосконалення чи заміни альтернативними рішеннями вищої якості [3].

Незважаючи на активну роботу, яка проводиться вченими у напрямку вирішення проблеми оцінки якості програмного коду, задача в цілому не є вирішеною. Існуючі методики часто не дають цілісної картини про реальний стан коду, різні показники нерідко показують протилежні та неузгоджені результати [1].

Все зазначене вище обумовлює актуальність удосконалення підходів до оцінки якості програмного коду, формування цілісної картини по певному проекту, його придатності для використання практичних задач, а також напрямків для підвищення якості. Особливої актуальності в даному контексті набуває проблема моніторингу та аналізу дефектів у програмному коді. А також необхідний засіб, який реалізую даний функціонал.

Огляд останніх досліджень

Поняття якості програмного коду є складовою якості програмних продуктів (ПП). Якість коду може визначатись різними критеріями.

Першою широко відомою моделлю якості ПЗ стала запропонована в 1977 році МакКолом [4]. У ній характеристики якості розділені на три групи:

1) Фактори (factors), що описують ПЗ з позиції користувача та задаються вимогами.

2) Критерії (criteria), що описують ПЗ з позиції розробника і задаються як мети.

3) Метрики (metrics), використовувані для кількісного опису та вимірювання якості.

У 1978 Боем запропонував свою модель, що по суті представляє собою розширення моделі МакКолу [5].

У 1991 році в якості стандартної була прийнята модель якості ПЗ ISO 9126 [6]. Ця модель не є прямим розширенням раніше запропонованих. У ній оцінка якості ПЗ заснована на трьохрівневому розгляді.

1) Цілі (goals) - те, що ми хочемо бачити в ПЗ.

2) Атрибути (attributes) - властивості ПЗ, що показують наближення до цілей.

3) Метрики (metrics) - кількісні характеристики ступеня наявності атрибутів.

Вибір метрик був зроблений, ґрунтуючись на методі Альбрехта, а також на методології CMMI. CMMI містить набір рекомендацій у вигляді практик, реалізація яких, на думку розробників моделі, позволяє реалізувати цілі, необхідні для повної реалізації певних галузей діяльності. Найбільш відомою є модель CMMI for Development, орієнтована на організації, що займаються розробкою програмного забезпечення, апаратного забезпечення, а також комплексних систем [7].

Цілі статті

Основними цілями статті є:

- Покращення та спрощення процесу оцінки якості ПЗ
- Розробка методики аналізу якості ПЗ за допомогою метрик дефектів
- Розробка засобу моніторингу якості ПЗ

Поняття якості ПЗ та тестування у стандартах

Поняття якості ПЗ є достатньо широким. З точки зору користувача (власника) системи якість визначається у придатності програми до вирішення його прикладних задач і відповідності його вимогам. Така характеристика якості ПЗ є зовнішньою. З точки зору розробника якість ПЗ характеризується такими факторами як зручність її розвитку, супроводження, розуміння коду та ін. Така характеристика є внутрішньою [8]. Оскільки сучасне ПЗ в своїй основі мають, як правило, стандартизоване апаратне забезпечення, ОС та відрізняються програмним кодом, то в рамках даної роботи увага буде акцентуватися на якісних характеристиках саме ПЗ, які за своєю природою відповідають наведеним характеристикам для ПЗ взагалі.

Очевидно, що якість програмного коду як основи ПЗ визначає як зовнішні так і внутрішні якісні характеристики системи в цілому. Таким чином, за умов використання надійних методів оцінки якості програмного коду, вдасться сформувати надійний механізм, який хоч і не дозволить отримати всеохоплюючу характеристику якості ПЗ у широкому розумінні (мова йде, насамперед, про такі показники як здатність системи коректно вирішувати поставлені задачі, відповідати очікуванням замовника та ін.), однак дозволить виявити прогалини у системі, до усунення яких використання певної інформаційної системи (ІС) буде являтися джерелом підвищеного ризику.

Традиційно у інженерії програмного забезпечення кількісною характеристикою якості програмного коду є метрика, як певна чисельна величина, яка дозволяє отримати значення деякої властивості ПЗ. Метрики є поширеним і достатньо надійним механізмом оцінки внутрішньої будови програм.

Якість ПЗ – ступінь, в якій ПЗ володіє необхідною комбінацією властивостей [9].

Якість ПЗ – це сукупність характеристик ПЗ, що відносяться до його здатності задовольняти встановлені та передбачувані потреби [10].

Дефект ПЗ – це недолік ПП, що є причиною його неочікуваного виконання [8, 10]. З точки зору користувача, дефект - це те, що не відповідає його очікуванням до програмного забезпечення.

Дефект якості програмного коду – недолік програмного продукту, який відображає невідповідність характеристик ПЗ встановленим і передбачуваним потребам [8, 10].

Поняття якості програмного коду є складовою якості ПП. Якість коду може визначатись різними критеріями. Деякі з них мають значення тільки з точки зору людини. Наприклад, форматування тексту програми — неважливо для комп'ютеру, але може мати велике значення для супроводу. Багато з існуючих стандартів кодування, що визначають специфічні для мови програмування угоди та задають низку правил, мають на меті полегшити супровід ПЗ в майбутньому. Також існують інші критерії, що визначають якість написання коду, наприклад, такі, як структурованість — ступінь логічного розділення коду на блоки [11].

Стандартом ISO прийнято наступне визначення тестування. Тестування - це спостереження за функціонуванням ПЗ в специфічних умовах з метою визначення ступеня відповідності ПЗ вимогам до нього [12].

Це визначення показує, що:

1. Тестування саме по собі не змінює ПЗ, а значить, не здатне впливати на ті метрики якості, які залежать тільки від самого ПЗ.
2. Тестування може служити методом контролю якості ПЗ, а саме тих його характеристик, які проявляються при функціонуванні ПЗ.

Незважаючи на істинність першого твердження, тестування здатне змінити якість ПЗ в тій його частині, яка відноситься до сприйняття ПЗ зацікавленими особами. Деякі метрики якості відображають таке сприйняття, і на їх значення тестування здатне впливати.

Вибір метрик для методики розрахунку

Метрика - це кількісний масштаб і метод, який може використовуватися для вимірювання[13]. Введення і використання метрик необхідно для поліпшення контролю над процесом розробки, зокрема над процесом тестування [14].

Мета контролю тестування полягає в отриманні зворотного зв'язку і візуалізації процесу тестування. Необхідну для контролю інформацію збирають (як вручну, так і автоматично) і використовують для оцінки стану і прийняття рішень, таких як покриття (наприклад, покриття вимог або коду тестами) або критерії виходу (наприклад, критерії закінчення тестування). Метрики також можуть бути використані для оцінки прогресу виконання запланованих робіт і освоєння бюджету [15].

Якість ПЗ може бути описана великою кількістю різнорідних характеристик. Поняття якості програми – багатопланове та може бути виражено адекватно тільки деякою структурованою системою характеристик або атрибутів. Така система характеристик називається моделлю якості, в якій метрики використовуються для кількісного описання та вимірювання якості [3,16, 17].

Вибір метрик варто зробити, ґрунтуючись на методі Альбрехта, а також на методології CMMI, як найбільш часто використовувані при оцінці якості [3,7,10].

Capability Maturity Model Integration (CMMI) - набір моделей (методологій) вдосконалення процесів в організаціях різних розмірів та видів діяльності. CMMI містить набір рекомендацій у вигляді практик, реалізація яких, на думку розробників моделі, позволяє реалізувати цілі, необхідні для повної реалізації певних галузей діяльності.

Найбільш відомою є модель CMMI for Development, орієнтована на організації, що займаються розробкою програмного забезпечення, апаратного забезпечення, а також комплексних систем.

Важливим показником якості програмного коду є щільність дефектів. Щільність дефектів (defect density) – кількість дефектів на одиницю розміру продукту. Є різні різновиди цього показника [7].

Метрика Альбрехта. Метрика дефектів якості програмних засобів. Існують два різновиди цієї метрики:

- Заснована на рядках коду
- Заснована на функціональних покажчиках.

Метрика, заснована на функціональних покажчиках використовується для непрямої оцінки рівня якості процедурно-орієнтованих ПС. Перевага - легкість обчислення. Недолік - результати ґрунтуються на суб'єктивних даних. Використовуються не прямі, а непрямі вимірювання.

Формула метрики:

$$DQ = \frac{\text{Кількість дефектів}}{FP} \quad (2.1)$$

$$FP = F * \left(0.65 + 0.01 * \sum_{i=1}^{14} k_i\right) \quad (2.2)$$

FP - функціональні покажчики.

$$F = \sum_{i=1}^{5} f_i \quad (2.3)$$

F – загальна кількість функціональних покажчиків.

Значення коефіцієнтів регулювання складності k залежать від відповідей на 14 запитань, що стосуються впливу певних факторів на виконання функцій програмного забезпечення, на які відповідає людина.

Саме через суб'єктивність значення даного виду метрики, була обрана метрика, заснована на рядках коду.

Дані на вхід:

- кількість виявлених помилок,
- кількість рядків коду для даної функціональності

Виконання розрахунків

DQ = Кількість помилок/Кількість рядків коду (DQ - щільність дефектів) [18].

SLOC = $\sum_{i=1}^{N} S_i$ (кількість рядків коду), де N – кількість структурних блоків у коді, S_i – обсяг і-го блоку.

Обґрунтування метрики

За результатами вимірювань визначити, чи досягнута мета, чи вирішене завдання, чи отримані відповіді на питання підрахунку кількості дефектів. Мета цієї діяльності - сформувати попередні оцінки, які дозволять:

- Пред'явити замовнику коректні вимоги за вартістю і витратами на розробку програмного продукту.
- Скласти план програмного проекту.

При виконанні оцінки можливі два варіанти використання LOC- і FP-даних:

- В якості оціночних змінних, що визначають розмір кожного елемента продукту.
- В якості метрик, зібраних за минулі проекти і входять до метричний базис фірми.

Інтерпретація даних

Показник щільності дефектів обчислюють як відношення загальної кількості знайдених дефектів до кількості тестових процедур, виконаних для даної функціональності або сценарію використання системи. Так, у разі виявлення високої інтенсивності помилок в конкретній функціональності необхідно провести причинно-наслідковий аналіз. Чи є функціональність занадто складною і тому в ній можна очікувати високої щільності помилок? Чи є проблеми, пов'язані з проектуванням або реалізацією функціональності? Чи не мало місце невірне (або недостатнє) виділення ресурсів для реалізації даної функціональності з причини неправильної оцінки її ризику? Можливий також висновок про те, що відповідальним за дану функціональність розробникам потрібне додаткове навчання.

Універсальна метрика якості коду. *Дані на вхід:*

- Кількість знайдених дефектів під час перегляду коду.
- Час.

Виконання розрахунків

CRD = Кількість дефектів/хвилина, де CRD - code review defects.

Обґрунтування метрики

Дана метрика вимірюється в ході процесу перегляду коду (code review). Review – діяльність, що проводиться для встановлення придатності, адекватності та ефективності, яку має досягнути об'єкт [16]. Для того, щоб переконатись, що код короткий, простий та відповідає діловій меті «переглядач» запускає секундомір і починає дивитись код. Кожен раз, коли він зустрічає в програмі помилку, додає до свого лічильника +1. Таким чином, розділивши показник лічильника на час, отримаємо метрику якості коду.

Alternative defect density (Альтернативна щільність помилок). *Дані на вхід:*

- Кількість нових помилок після додання коду
- Кількість рядків доданого коду

Виконання розрахунків:

ADD = Кількість дефектів/ Кількість рядків доданого коду

Обґрунтування метрики:

Оцінка якості програмного продукту та ефективності доданого коду.

Дані на вихід:

Кількість помилок на 1000 рядків коду

Інтерпретація даних

Типовим вважається кількість дефектів 1-5 на 1000 рядків коду.

Defect density (для програміста). Щільність помилок для коду автора. Отримати дану метрику можна, наприклад, за допомогою MSR Tools.

Дані на вхід:

- Кількість помилок
- Кількість рядків коду певного автора

Виконання розрахунків

DDP = Кількість помилок/ Кількість рядків коду певного автора

Обґрунтування метрики

Якісний аналіз даної метрики допоможе визначити проблемні участки коду, ефективність роботи конкретного програміста, попередити програміста про потенційну небезпеку змін, розподілити ресурси тестування, оптимально обрати дату релізу.

Дані на вихід

Кількість помилок на 1000 рядків коду

Інтерпретація даних

Типовим вважається кількість дефектів 1-5 на 1000 рядків коду.

Розробка засобу

На рис. 1 відображено зв'язок між акторами та прецедентами.

Актор – це користувач системи. Прецедент – це функціональність системи, завдяки якої користувач може отримати конкретний, вимірюваний та необхідний результат. Він визначає один з варіантів використання системи та описує типовий спосіб взаємодії користувача з системою.

Після запуску програми відкривається вікно Code Analyzer з активною першою вкладкою Defect density (рис. 2). Кнопка «Browse file..» відкриває вікно діалогу відкриття файлу. Обираємо файл з кодом та завантажуємо. Назва файлу відображується в лівому текстовому блоку, зміст - в правому. Кнопка «Analyze» обраховує кількість рядків коду завантаженого файлу. Кнопка «Calculate» обраховує метрики DQ, ADD, DDP за формулами після введення значень в необхідні поля.

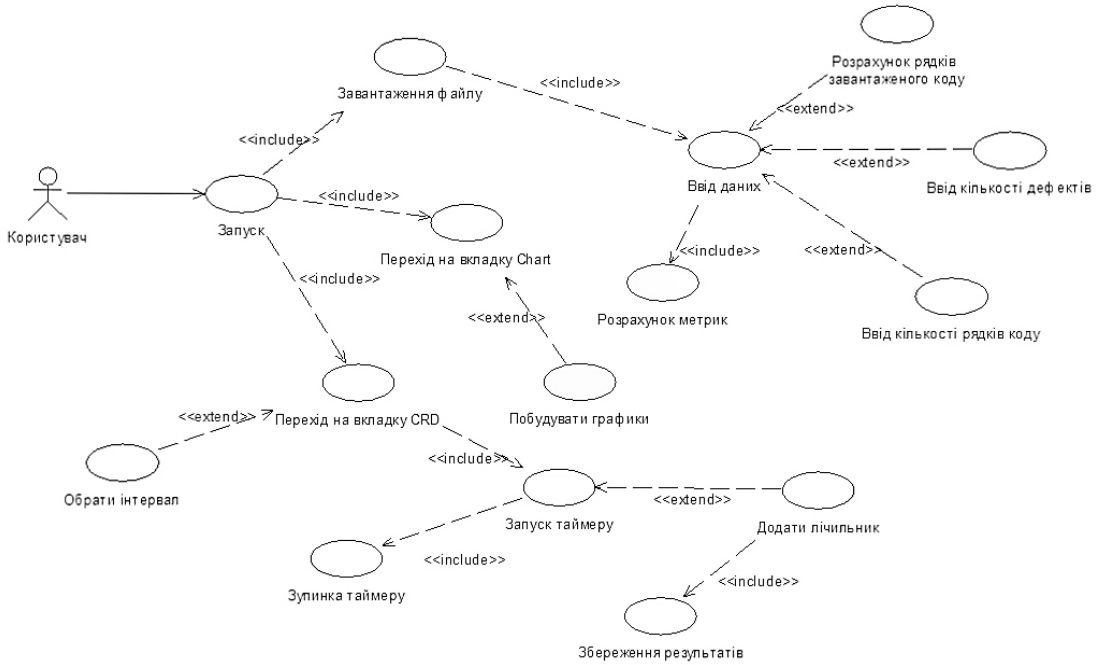

Рис.1 Діаграма варіантів використання засобу СА

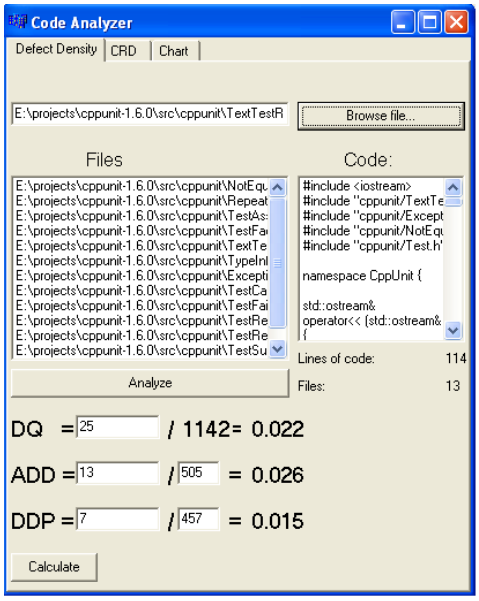

Рис.2 Інтерфейс програми, вкладка Defect density

Для розрахунку метрики CRD необхідно перейти на другу вкладку (рис. 3).

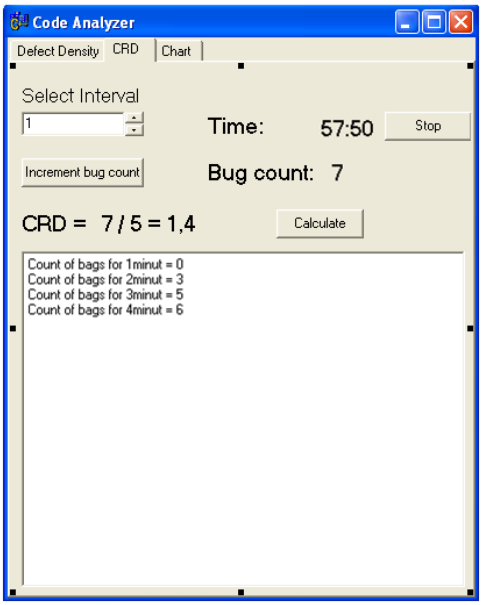

Рис.3 Інтерфейс програми, вкладка CRD

Встановити інтервал лічильника можна елементом «Select Interval». Кнопка «Start» запускає таймер «Time» та відображається підрахунок часу. Зупиняє таймер кнопка «Stop». В ході перегляду коду знайдений дефект дорівнює одному натисканню кнопки «Increment bug count» при інтервалі 1. «Bug count» відображає загальну кількість лічильника. В текстовому блоці відбувається фіксування кількості дефектів за хвилину.

Вкладка «Chart» призначена для побудови графіків вимірюваних метрик (рис. 4).

На першому графіку під назвою «Defect density» зображена метрика CRD, що розраховується у відповідній вкладці програми. На осі x відкладається час у хвилинах, на вісі y – кількість дефектів.

На другому графіку під назвою «Alternative defect density» зображені метрики DQ, ADD, DDP, що обраховуються у вкладці програми «Defect density». На вісі x зображені три метрики, на вісі y – кількість дефектів.

Будує графіки кнопка «Build».

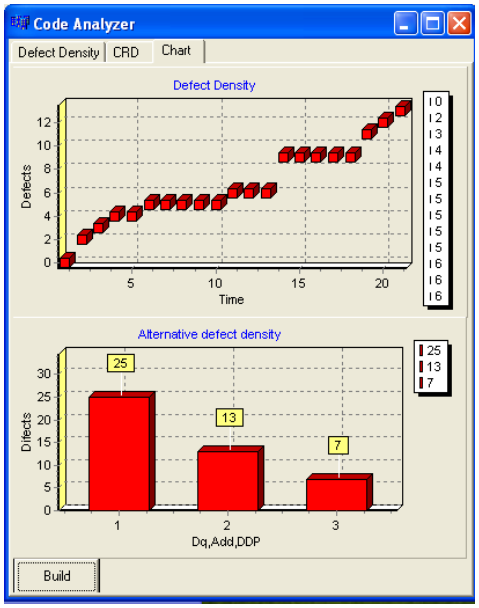

Рис.4 Інтерфейс програми, вкладка Chart

Висновки

Як показав аналіз, приведений раніше у статті, на даний момент не існує єдиного зручного засобу для вимірювання усього необхідного переліку метрик дефектів.

Для того, щоб обрахувати обрані метрики для оцінки якості програмного коду потрібно використовувати декілька програмних засобів окремо, перемикаючись між ними та записуючи результати вручну. Наприклад, потрібно поміряти кількість рядків коду в одному засобі; на калькуляторі або в Excel розраховувати непрямі метрики за формулами, створювати таблицю для побудови графіків, окремо їх будувати. Для визначення метрики CRD додатково потрібно вмикати таймер та запам'ятовувати кількість знайдених дефектів, що в свою чергу відволікає безпосередньо від перегляду.

Розроблений засіб автоматизує процес вимірювання та розрахунків метрик,

пришвидшує та спрощує роботу інженера з якості. З допомогою засобу рівень організації оцінки якості програмного коду в проекті значно покращується.

Використання у методиці розрахунку метрик дефектів якості програмного коду проводиться на основі даних отриманих під час тестування.

Розроблений засіб допомагає організовувати процес у тестуванні визначення якості програмного забезпечення та здійснювати аналіз метрик.

Список використаних джерел

1. *Janusz Laski, William Stanley* Software Verification and Analysis. An Integrated, Hands-On Approach - Springer-Verlag London Limited, 2009.-205p.
2. *Tharwon Arnuphaptrairong*. Top Ten Lists of Software Project Risks : Evidence from the Literature Survey. IMECS International MulityConference of Engineers and Conputer Scientistis. VOL 1, March 2011.-732-737pp.
3. Handbook of Software Quality Assurance. Fourth Edition / *editor G. Gordon Schulmeyer*. - ARTECH HOUSE, INC., 2008.-485p.
4. *J. McCall, P. Richards, G. Walters*. Factors in Software Quality. three volumes, NTIS AD-A049-014, AD-A049-015, AD-A049-055, November 1977.
5. *B. W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. MacLeod, and M. J. Merritt*. Characteristics of Software Quality. North Holland Publishing Company, 1978.-524p.
6. International Standard ISO/IEC 9126. *Information technology – Software product evaluation – Quality characteristics and guidelines for their use.* International Organization for Standardization, International Electrotechnical Commission, Geneva, 1991.
7. *CMMI for Development V1.3* [Электронный ресурс] – 2010. - Режим доступа: http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=9661
8. International Standard ISO 8402:1994 *Quality management and quality assurance* [Электронный ресурс] – 1994. - Режим доступа: http://www.iso.org/iso/catalogue_detail.htm?csnumber=20115
9. 1061-1998 IEEE *Standart for Software Quality Metrics Methodology* [Электронный ресурс] – 1998. - Режим доступа: http://standards.ieee.org/findstds/standard/1061-1998.html
10. *Marc McDonald; Robert Musson; Ross Smith* The Practical Guide to Defect Prevention - Microsoft Press, 2007.-480p.
11. Тестирование и качество ПО[Электронный ресурс] Качество программного обеспечения / The Institute of Electrical and Electronics Engineers – 2004. - Режим доступа: http://software-testing.ru/files/se/3-10-software_engineering_quality.pdf
12. International Standard ISO/IEC/IEEE 29119-1:2013 *Software and systems engineering – Software testing – Part1: Concepts and definitions* [Электронный ресурс] – 2013. - Режим доступа: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=45142
13. International Standard ISO/IEC 14598-1 *Information technology — Software product evaluation* [Электронный ресурс] – 1999. - Режим доступа: http://www.iso.org/iso/catalogue_detail.htm?csnumber=24902
14. Program Verification Systems, *Метрики кода програмного обеспечения* [Электронный ресурс] – 2009. - Режим доступа: http://www.viva64.com/ru/a/0045/
15. *Питер Брукс* Метрики для управления ИТ-услугами/ Пер.с англ.- М.:Альпина Бизнес Букс, 2008. - 283 с.
16. ISO 9000:2000 *Quality management systems - Fundamentals and Vocabulary* [Электронный ресурс] – 2000. - Режим доступа: http://www.iso.org/iso/catalogue_detail?csnumber=29280
17. ГОСТ 28195-89 Оценка качества программных средств [Электронный ресурс] – 2000. - Режим доступа: http://vsegost.com/Catalog/11/11212.shtml
18. Научно-технический вестник СПбГИТМО (ТУ). Выпуск 10 / Ред. *Ю.А. Гатчин*. - СПб.: СПбГИТМО(ТУ), 2003. - 209 с.

Відомості про авторів:

Дишлевий Олексій Петрович – старший викладач кафедри інженерії програмного забезпечення Національного авіаційного університету. Наукові інтереси: емпірична інженерія програмного забезпечення.
e-mail: oleksiy.dyshlevyy@livenau.net

Драпушко Юлія Володимирівна – студентка Інституту комп'ютерних інформаційних технологій Національного авіаційного університету. Наукові інтереси: емпірична інженерія програмного забезпечення.
e-mail: YuliyaV.Drapushko@livenau.net