УДК 004.415.5

О. Поморова, Д. Іванчишин
Хмельницький національний університет,
кафедра системного програмування

ДОСЛІДЖЕННЯ ЗАСОБІВ ОЦІНКИ ЯКОСТІ НА РІЗНИХ ЕТАПАХ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Досліджено додатки для оцінки якості програмного забезпечення на різних етапах життєвого циклу. Проаналізовано використання компонентів штучного інтелекту у таких додатках.

Ключові слова: якість, життєвий цикл, програмне забезпечення.

The paper presents the results of research tools for assessing of software quality at various stages of the life cycle. The use of intelligent components in application for the evaluation of software quality was researched.

Key words: quality, life cycle, software.

Вступ

Потреба в оцінці якості програмного забезпечення (ПЗ) [1,2] виникає з двох головних причин: в критично важливих сферах відмова ПЗ призводить не лише до матеріальних втрат, а й до шкоди для людей; застосування якісного ПЗ вимагає менше коштів на етапі супроводу, враховуючи використання, підтримку та внесення змін для нагальних потреб. Якщо друга причина призводить до значних фінансових збитків, а також збільшення навантаження на команду розробників, що є значними наслідками, але не критичними, то недоліки в програмному забезпеченні, що наносять шкоду людському здоров'ю, є недопустимими.

Сьогодні існують спеціалізовані додатки, котрі на різних етапах життєвого циклу допомагають аналізувати та оцінювати якість ПЗ, однак вони лише частково виконують завдання забезпечення якості програмного забезпечення. Наприклад, у лютому 2007 р. дванадцять винищувачів F-22 виконували переліт з військової бази США на Гавайях в Японію. У момент перетину часової границі на усіх літаках через програмну помилку відмовили бортові комп'ютери [3]. У 2009 р. широкого розголосу набула інформація про можливість перехоплення радіообмінів безпілотного літального апарата ВПС США MQ-1. Допущені недоліки в проектуванні програмної системи апарата призвели до відсутності шифрування потоку його даних, що надало можливість до безперешкодного доступу до конфіденційної критично важливої інформації [4]. П'ятого грудня 2010 р. у Тихий океан впали три супутники російської навігаційної системи ГЛОНАСС. Фінансові збитки оцінюються в 138 млн. доларів. У результаті розслідування причиною аварії була визнана помилка у програмі, що призвела до заправки ракети неправильною кількістю палива [5].

Згадані приклади свідчать про недоліки сучасного програмного забезпечення та необхідність аналізу виникнення подібних проблем. Незважаючи на широке розповсюдження та використання додатків для оцінки якості програмного забезпечення, якість сучасного ПЗ не відповідає вимогам часу.

Постановка задачі

Необхідно дослідити основні функціональні особливості додатків для аналізу та підвищення якості ПЗ. Визначити переваги, недоліки та етапи життєвого циклу ПЗ, на яких використання засобів оцінки якості є доцільним. Прослідкувати ефективність застосування компонентів штучного інтелекту, як одного зі шляхів підвищення якості програмного забезпечення.

Основна частина

Якість програмного забезпечення – це сукупність характеристик ПЗ, що належать до його здатності задовольняти встановлені та прогнозовані потреби. Виділяють внутрішню та зовнішню якість ПЗ. Під час розробки програмного продукту формується внутрішня якість програмного продукту. Зовнішня якість ПЗ проявляється під час його взаємодії з користувачем і визначає якість у використані. Стандарт ISO 9126 визначає шість основних характеристик, що відображають внутрішню та зовнішню якість ПЗ:

- надійність (Reliability);
- функціональність (Fnctionality);
- зручність використання (Usability);
- ефективність (Efficiency);
- зручність супроводу (Maintainability);
- портативність (Portability).

Забезпечення якості ПЗ – це сукупність заходів, які охоплюють всі технічні етапи розробки, випуску та експлуатації програмного продукту, виконані на різних стадіях життєвого циклу для забезпечення його якості.

Результати оцінки якості програмного забезпечення слугують засобом для визначення програмних продуктів, що найкраще відповідають вимогам користувачів та показником у виборі їх для застосування.

Задача оцінки якості ПЗ полягає у перевірці повноти його відповідності встановленим вимогам, а також у визначенні ступеня, в якому програмне забезпечення має необхідну комбінацію властивостей. До методів оцінки якості ПЗ належать: тестування, формальний аналіз, аналіз коду, властивостей ПЗ, архітектури проекту, розроблення, виміри (визначення метрик, профілювання), моделювання, використання моделей для оцінки властивостей ПЗ, вивчення документів з метою пошуку проблемних місць і перевірки відповідності стандартам, стилям, прийнятим правилам та угодам.

Сьогодні тестування є практично єдиним способом перевірки якості ПЗ, що застосовується до вихідного коду – кінцевого продукту розроблення програмного забезпечення. Тестування не замінюється жодним з вказаних методів контролю якості або їх поєднанням. Зміст тестування характеризується, як спостереження за функціонуванням програмних продуктів в специфічних умовах з метою визначення ступеня їх відповідності до вимог [6]. Визначення вказує на два головні аспекти:

– тестування не змінює програмне забезпечення, а відповідно не впливає на ті метрики якості, котрі залежать тільки від самого ПЗ;

– тестування є методом контролю якості ПЗ, а саме тих його характеристик, які проявляються під час функціонування ПЗ.

Під час виконання проекту необхідно від початку враховувати стандарти і вимоги, згідно з якими буде виконуватись тестування продукту. Які засоби використовуватимуться для пошуку та документування дефектів. Це дозволяє скоротити витрати та час на розробку продукту, а також гарантувати високий рівень якості.

Сьогодні існує декілька загальновідомих моделей життєвого циклу програмного забезпечення: каскадна, ітераційна, спіральна, прототипування, v-подібна та швидкої розробки додатків RAD. Найрозповсюдженішою є спіральна модель [7], яка зображена на рисунку.

Вона має деякі переваги і складається з витків, кожен з яких відповідає створенню фрагменту чи версії ПЗ. Під час проходження витків уточнюються цілі та характеристики проекту, визначається його якість і плануються подальші роботи наступного витка спіралі. Отже, поглиблюються і послідовно конкретизуються деталі проекту, у результаті вибирається обґрунтований варіант, який доводиться до реалізації. Кожен виток має чотири сектори:

- визначення цілей;
- оцінка і вирішення ризиків;
- розробка та тестування;
- планування.

Спіральна модель життєвого циклу ПЗ

Використовуючи такий підхід, тестування перестає бути етапом, що виконується після створення повного коду програмного продукту. Робота над тестами починається від першого етапу виявлення вимог до майбутнього проекту і тісно пов'язана з поточними задачами. На початку кожного нового витка визначається мета тестування та методи її досягнення, а наприкінці – наскільки ця мета була досягнута. Виявлення та усунення найвпливовіших ризиків на ранніх етапах розробки значно підвищує її ефективність. Яскравим прикладом неякісно проведених початкових етапів розробки програмного продукту є історія американського програміста Джея Редкліфа. Хворий на діабет, він виявив критичну вразливість своєї інсулінової помпи. Вона не виключала можливості приймання сигналів не тільки з давача цукру, а й з будь-яких бездротових пристроїв. Це дало змогу Джею за допомогою дешевого USB-передавача, надсилати помпі керуючі сигнали і регулювати рівень інсуліну, що потрапляв у його кров. Зазначимо, що програмне забезпечення інсулінової помпи працювало цілком коректно і повністю виконувало свої функції в визначених межах. До того ж воно містило недолік, від якого залежало людське життя. Наслідком до цього стала помилка, допущена не під час написаня вихідного коду, а саме під час формування вимог до наявного ПЗ.

Отже, оцінку або хоча б прогноз якості програмного забезпечення потрібно робити вже на початкових етапах розробки.

Існує велика кількість програмних додатків для аналізу та підвищення якості ПЗ. Умовно їх можна поділити на п'ять груп: аналіз та оцінка якості вихідного коду, середовища розробки з можливістю тестування якості ПЗ, аналіз функціональних особливостей ПЗ, аналіз, верифікація та управління вимогами, керування розробленням проекту. Додатки, що належать до цих груп, наведені в табл. 1.

У першій групі подані додатки, що працюють безпосередньо з вихідним кодом програмного продукту. Відповідно, для аналізу та оцінки якості ПЗ вони використовують повний перелік сучасних методик, що об'єднують статичне, динамічне, регресійне, модульне тестування.

У другій групі наведені додатки, що містять певний функціонал з першої групи, але також пропонують широкі можливості для розробки ПЗ.

До третьої групи входять додатки, що виконують аналіз певних функційних особливостей ПЗ, наприклад, використання стеку пам'яті та ін.

У четвертій групі наведені додатки, що застосовуються для роботи з вимогами до ПЗ, а також можуть використовуватись для тестування та верифікації вимог.

Таблиця 1

Групи додатків для аналізу та підвищення якості ПЗ

Група	Перелік ПЗ
Аналіз та оцінка якості вихідного коду	LDRA Testbed, TBrun, TBvision, TBsecure, TBevolve, TBsafe, TBmisra, Cantata++, AdaTEST 95, QA·C, QA·C MISRA, VectorCAST/Ada, VectorCAST/C, VectorCAST/C++, VerOCode, CodePeer, C Verifier, GNATCoverage, SPARK Toolset, TESSY, C++ Test, MALPAS, AQtime, BoundsChecker, Bullseye Coverage, CMT++, Code Coverage, CodeCheck, CodeWizard, CTA++,CTC++, devAdvantage, Diversy Analyzer, GlowCode, Insure++, Leak Check, OSPC, Panorama, McCabe TQ, Prevent, Purity, TCAT C/C++, Test Coverage, Testers Desk
Середовища розробки з можливістю тестування якості ПЗ	Perfect Developer , Perfect Developer Critical System Edition, Concerto, SCADE Suite, STOOD
Аналіз функціональних особливостей ПЗ	VerOStack, VerOLink, GNATstack
Аналіз, верифікація та управління вимогами	TBreq, Telelogic DOORS
Керування розробкою проекту	Telelogic Rhapsody, Aba CM Enablement

У п’ятій групі наведені пакети для керування розробленням проекту, що надають можливості з керування та узгодження робіт над проектом.

Отже, найбільше розповсюдженими сьогодні є продукти, котрі призначені для аналізу та оцінки якості вихідного коду. Додатків у цій групі виявилось більше ніж у чотирьох інших разом узятих. Це вказує на те, що питання оцінки якості програмного забезпечення на ранніх стадіях життєвого циклу не набуло достатньо високого рівня розвитку і вимагає детальнішого вивчення.

Програмне забезпечення вважається критичним, якщо воно контролює або виконує моніторинг небезпечного або критично-важливого апаратного або програмного забезпечення. Програмне забезпечення, яке надає інформацію, необхідну для прийняття критичного (safety-related) рішення також підпадає під категорію критичного програмного забезпечення [8,9]. Таке ПЗ повинно відповідати підвищеним стандартам надійності та безпеки, зокрема, RTCA DO-278, (ESARR) 4 / 6, CAP 670 / SW01, IEC 61508, DO-178B Level A та ін. Для перевірки відповідності критичне ПЗ проходить обов’язкову сертифікацію. Результати аналізу та дослідження характеристик засобів аналізу якості та середовищ розробки ПЗ наведені в табл. 2. У табл. 2 подано додатки, що можуть використовуватись для роботи з програмним забезпеченням критичного застосування, проілюстровані їх основні функціональні можливості та зазначені основні переваги та недоліки.

Одним із засобів підвищення ефективності процесу оцінки якості програмних засобів є використання інтелектуальних методів. Штучний інтелект (artificial intelligence)[10] — ШІ (AI) – це властивість автоматичних систем брати на себе окремі функції інтелекту людини, наприклад, вибирати і приймати оптимальні рішення на основі раніше отриманого досвіду та раціонального аналізу зовнішнього впливу. На початкових етапах життєвого циклу виконується опрацювання великих обсягів неточної, неповної та наближеної інформації. Це саме той набір задач, для розв’язання яких найкраще підходять методи штучного інтелекту. Здатність систем штучного інтелекту до самонавчання та розв’язання нових задач на основі результатів самонавчання, дозволяє підвищити якість аналізу програмного забезпечення.

Інтелектуальна система – це технічна чи програмна система, що здатна розв’язувати задачі, традиційно зараховані до творчих і належать до конкретної предметної галузі, знання про яку зберігаються в пам’яті такої системи. Системи, що використовують штучний інтелект, здатні:

- приймати гнучкі рішення;
- вирішувати неоднозначні або суперечливі ситуації ;
- визначати важливість та пріоритети між декількома задачами;
- знаходити відмінності в схожих ситуаціях;
- знаходити подібності в різних ситуаціях.

Коли користувач зустрічає в описі програмного продукту вказівки на наявність штучного інтелекту, він сподівається, що система здатна реалізовувати саме такі функції.

Деякі розробники сьогодні пропонують ПЗ оцінки якості, що за їх уявленням містить інтелектуальні компоненти. З розглянутих у табл. 2 до таких належать: Cantata++, TBevolve та VectorCAST/Ada/C/C++.

Таблиця 2

Додатки для аналізу якості програмного забезпечення критичного застосування

Розробник, продукт	Мова	Платформа	Переваги	Недоліки
Розробник: LDRA. Базовий продукт: LDRA Testbed – засіб для аналізу якості та тестування ПЗ, містить статичний та динамічний аналіз, перевірку стандартів та структури програмного коду, аналіз покриття, відслідковування змінних, переходів та рекурсій, звіти по метриках. Розширення: TBrun – автоматичне модульне тестування; TBreq – відслідковування вимог на протязі ЖЦ ПЗ; TBvision – відповідність стандартам та пошук недоліків коду ПЗ;TBevolve – аналіз змін внесених в ПЗ;	Ada 83, Ada 95, C, C++, Java	Windows 7/Vista/ XP/ 2000/NT Unix (Sun Solaris & HP-UX) Linux	набір засобів та розширень для аналізу якості програмного забезпечення будь-якого рівня складності з підтримкою усіх ключових платформ.	вартість 19800€ за першу копію
IPL. Cantata++ – засіб для аналізу якості та тестування ПЗ, що включає статичний та динамічний аналіз тестування білого та чорного ящика, позитивне та негативне тестування, регресійне тестування. AdaTEST 95 – аналіз якості Ada додатків, підтримує стат. та динамічний та аналіз	C, C++	Windows 9x, NT, 2000,XP Solaris, Linux AIX, Unix	вичерпні функціональні можливості	вартість 7500$
	Ada 83, Ada 95		функціонально потужній засіб для тестування Ada проектів	–
Veroce]. VerOCode – аналіз виконання додатків на наявність помилок. VerOStack – аналіз використання стеку пам’яті додатком VerOLink – аналіз інтеграції об’єктних файлів	-	PC/ Windows NT [or higher]	підвищення якості ПЗ при комплексному використанні набору додатків	обмеж. функціо-нальність окремих додатків
Programming Research. QA·C – аналіз С коду на відповідність стандартам та помилки QA·C MISRA – аналіз С коду на відповідність стандарту MISRA VectorCAST/Ada/C++ – засіб для аналізу якості та тестування ПЗ на мовах Ada та C++;VectorCAST/Cover – аналіз покр. коду	C	Windows (x86), Solaris (sparc) Linux (x86), IBM AIX (sparc)	підвищення якості коду на ранніх стадіях розробки	робота лише з С кодом
	Ada, C++		зручн. інтер-фейс широка функціонал.	Окр. пак. для Ada, C++, вар. 6850$
Розробник: Adacore CodePeer – аналіз якості вихідного коду GNATCoverage – аналіз покриття коду GNATStack – аналіз використання стеку пам’яті додатком SPARK Toolset – аналіз якості коду для мови SPARK	Ada, C, C++	Linux x86-64 SPARC Windows x86	з GNAT PRO засіб для розробки та контролю якості ПЗ	прив’язка до GNAT PRO Ada/C/C++
	SPARK		тестування SPARK коду	пр. до SP. PRO
Echer Technologies. Perfect Developer Critical System Edition – комплексний засіб для розробки, аналізу якості та тестування ПЗ C Verifier – аналіз якості С коду	C++, C#, Java	Windows NT or Linux OS	мод. та оцінка якості ПЗ	-
	C		матем. дов. відсутності помилок	обмежена підтримка мов

Розробник Cantata++ компанія IPL в офіційних документах додає до назви свого продукту термін «intelligent testing», а в описі вказує «intelligent unit and integration testing», тобто декларується певна інтелектуальність при модульному та інтеграційному тестуванні. Детальний аналіз продукту та запити розробнику виявили, що інтелектуальність – це частина бренду компанії IPL, що використовується для Cantata++. Під інтелектуальністю розробник розуміє використання його продукту, як «інтелектуального шляху» до тестування ПЗ. Тобто, реалізація певного інтелектуального компоненту у вказаному додатку відсутня.

Для додатка TBevolve компанія LDRA Software Technology вказує на використання Intelligent Difference Analysis (IDA) – інтелектуального аналізу різниці. Під ним розуміється особлива методика відслідковування на рівні вихідного коду додавання, видалення та змін, що були внесені в функціональність програмного засобу. Однак розробник використовує звичайний алгоритмічний метод, що не може бути віднесений до інтелектуальних.

Додаток VectoCAST/Ada/C/C++ за інформацією розробника Programming Research містить певні «intelligent stubs», тобто певні інтелектуальні «заплатки», що використовують при структурному програмуванні для заміни виклику підпрограм. В якості інтелектуальності своєї системи розробник розуміє алгоритм використання «заплаток», який базується на чіткій послідовності визначених дій. Властивості чи компоненти, що притаманні інтелектуальним системам, у цьому випадку також відсутні.

Висновки

В результаті дослідження засобів оцінки якості програмного забезпечення на різних етапах життєвого циклу, виявлено, що сьогодні найбільше поширені засоби для оцінки якості вихідного коду ПЗ. Це справедливо і для додатків, що можуть використовуватись з програмним забезпеченням критичного застосування.

Якісні показники програмного забезпечення закладаються на ранніх етапах розробки, тому необхідним та доцільним є розвиток продуктів для аналізу якості на цих етапах життєвого циклу.

Дослідження щодо використання інтелектуальних компонентів у додатках для оцінки якості програмного забезпечення показало, що в усіх розглянутих додатках, функції чи компоненти, притаманні інтелектуальним системам, виявились відсутні. Актуальність проблеми підвищення якості сучасного ПЗ вказує на потребу впровадження інтелектуальних компонентів для підвищення ефективності відомих або розроблення нових методів оцінки якості.

1. 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. 2. 1061-1998 IEEE Standard for Software Quality Metrics Methodology. 3. "Lockheed's F-22 Raptor Gets Zapped by International Date Line: Raptors arrive at Kadena" Air Force, 26 February 2007, <http://www.af.mil/news/story.asp?storyID= 123041567> 4. Siobhan Gorman et al., "Insurgents Hack U.S. Drones," The Wall Street Journal, 17 December 2009, sec. A. p. 1., <http://online.wsj.com/article/SB126102247889095011.html> 5. "Russia's $2 billion project to rival America's GPS suffers setback" Weir, Fred, December 6, 2010, <http://www.csmonitor.com/World/Europe/2010/1206/Russia-s-2-billion-project-to-rival-America-s-GPS-suffers-setback> 6. Кулямин В.В., Петренко О.Л. ИСП РАН. Место тестирования среди методов оценки качества ПО. <http://software-testing.ru/library/5-testing/117-2008-10-13-19-25-13> 7. Boehm, Barry W., "A Spiral Model of Software Development and Enhancement," IEEE Computer, May 1988. 8. NASA/CR-2003-212806. Certification Processes for Safety-Critical and Mission-Critical Aerospace Spftware. 9. Липаев В.В. Обеспечение качества программных средств. Методы и стандарты. – М.: Синтег, 2001. 10. Luger, George; Stubblefield, William (2004). Artificial Intelligence: Structures and Strategies for Complex Problem Solving (5th ed.). The Benjamin/Cummings Publishing Company, Inc.. ISBN 0-8053-4780-1, pp. 27–55.