УДК 004.891.3:004.3

Говорущенко Т.О.
Хмельницький національний університет

Боднар М.А.
Хмельницький національний університет

Стасенко А.С.
Хмельницький національний університет

ВИЗНАЧЕННЯ ВПЛИВУ ХАРАКТЕРИСТИК CODESTYLE НА ХАРАКТЕРИСТИКИ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У статті досліджується вплив характеристик CodeStyle на характеристики якості програмного забезпечення. Проведене дослідження дало можливість зробити висновок про те, що характеристики CodeStyle впливають на всі вісім характеристик якості ПЗ, а не тільки на зручність його використання.

Ключові слова: *програмне забезпечення (ПЗ), якість ПЗ, характеристики якості ПЗ, стандарт ISO 25010:2011, CodeStyle.*

Постановка проблеми. Ключовим фактором забезпечення ефективного застосування програмного забезпечення (ПЗ) та однією із основних вимог користувачів і зацікавлених осіб до сучасного ПЗ є досягнення високих значень показників його якості. Якість ПЗ є основним чинником для його успішного впровадження та експлуатації. Потреба в забезпеченні якості ПЗ випливає з того, що помилки та відмови ПЗ загрожують катастрофами, які призводять до людських жертв, екологічних катаклізмів, значних часових втрат та фінансових збитків.

Якість ПЗ має зовнішні та внутрішні характеристики. До зовнішніх характеристик належать властивості, які усвідомлює користувач ПЗ. До внутрішніх характеристик якості належать властивості, які цікавлять програміста (в т. ч. й властивості читабельності та зрозумілості коду, які визначаються CodeStyle). Поняття якості ПЗ включає не лише повноту та коректність реалізації потрібного функціоналу, але й простоту підтримки та модифікації коду, тому зовнішні та внутрішні характеристики якості взаємопов'язані – внутрішні характеристики впливають на зовнішні [1, с. 456-459].

Тоді актуальною задачею є визначення впливу характеристик CodeStyle на характеристики якості програмного забезпечення.

Аналіз останніх досліджень і публікацій. Різниця між внутрішніми та зовнішніми характеристиками якості дещо розмита, оскільки внутрішні характеристики впливають на зовнішні. Якщо ПЗ недостатньо зрозуміле або незручне в супроводі, в ньому важко виправляти дефекти, що, у свою чергу, впливає на такі зовнішні характеристики, як коректність та надійність. ПЗ, що «страждає» від браку гнучкості, неможливо покращити у відповідь на запити користувачів, що відбивається на його практичності [1, с. 458].

Важливим аспектом управління процесом розроблення ПЗ є «брама якості» – періодичні огляди (review), які дозволяють визначати, чи достатню якість має ПЗ на конкретному етапі розроблення [2]. Огляди коду не лише забезпечують більш високу ефективність виявлення помилок, ніж тестування, але й дозволяють виявляти типи помилок, на які тестування вказати не може. Ефективність виявлення дефектів при використанні неформальних оглядів коду складає 20-35%, а при використанні формальних інспекцій коду – 45-70% [1, с. 459–473]. На цю ефективність впливає читабельність та зрозумілість коду – чим читабельніший та зрозуміліший код, тим продуктивнішим є проведення оглядів коду (зазвичай, ця продуктивність становить 1000 рядків на день [1, с. 486]) і тим вищою є ефективність виявлення дефектів. «Програми повинні писатись для того, щоб їх читали люди, і лише потім для виконання машиною» [3].

У роботі [4] досліджено підхід до використання аналізів коду з метою покращення якості програмного забезпечення та зменшення певних класів несправностей для конкретних програмних систем. Дослідження [5] досліджує вплив швидкості перегляду на ефективність дефектоспроможності та якість програмного забезпечення. Рекомендована швидкість огляду у 200 рядків коду на годину була визначена у [5] як базова для окремих оглядів, в разі високої читабельності та зрозумілості коду.

Робота [6] спрямована на виявлення архітектурної тактики в коді, а також на візуалізацію взаємовпливу читабельності коду та якості ПЗ. У роботі [7] доведено, що якість програмного забезпечення також залежить від організації коду, тобто його дизайну; проаналізовано, як на якість ПЗ впливають шаблони проектування (patterns), антипаттерни (antipatterns), а також «код із запашком» (code smell) – код з ознаками проблем. У роботі [8] пропонується метод застосування стилів програмування на основі онтологій, що підвищує ефективність і автоматизує відповідні процеси.

Постановка завдання. Проведений аналіз досліджень та публікацій показав, що читабельність та зрозумілість коду, а також його зовнішній вигляд впливають на розуміння коду та на якість ПЗ. Зазначені властивості значною мірою визначаються стилем коду (CodeStyle), який має вплив на всі процеси життєвого циклу. Але на сьогодні не досліджено, які саме характеристики CodeStyle впливають на якість ПЗ, а також не досліджено, на які саме характеристики якості ПЗ впливає CodeStyle, чому й присвячено подальше дослідження.

Виклад основного матеріалу дослідження. *Характеристики якості програмного забезпечення.* Згідно зі стандартом [9] якість ПЗ визначається за 8-ма загальними характеристиками (характеристика – це набір властивостей ПЗ, за допомогою яких описується та оцінюється його якість): функційна придатність (Functional Suitability), ефективність (Performance Efficiency), сумісність (Compatibility), зручність використання (Usability), надійність (Reliability), захищеність (Security), супроводжуваність (Maintainability), можливість переносу (Portability). Характеристики якості ПЗ можуть бути уточнені на основі комплексних показників (підхарактеристик), які ґрунтуються на властивості задовольняти заявлені або виникаючі потреби. Згідно зі стандартом [9] на сьогодні характеристики якості залежать від 31 підхарактеристики – рис. 1. Підхарактеристика якості ПЗ виражається середнім зваженим арифметичним показником з урахуванням значень атрибутів, що оцінюють цю підхарактеристику та підлягають точному опису та вимірюванню, а також коефіцієнтів їхньої вагомості. Атрибути якості ПЗ визначені та описані в стандарті ISO 25023 [10]. Аналіз стандарту [10] дав можливість визначити залежність підхарактеристик якості від 203 атрибутів, але від 138 різних атрибутів.

Моделі характеристик якості програмного забезпечення. Якість ПЗ (Q), відповідно до стандарту [9], є функцією від восьми основних характеристик якості ($QCH = \{qch_1, ..., qch_8\}$). Множину характеристик якості ПЗ запишемо у вигляді $QCH = \{Fs, Pe, Ub, Rb, Cb, Scr, Mb, Pb\}$, де: Fs – функційна придатність, Pe – ефективність, Ub – зручність використання, Rb – надійність, Cb – сумісність, Scr – захищеність, Mb – супроводжуваність, Pb – можливість переносу. Вказані характеристики якості можуть приймати значення з певного діапазону. Тоді якість ПЗ є функцією від цих характеристик:

$$Q = f(Fs, Pe, Ub, Rb, Cb, Scr, Mb, Pb). \quad (1)$$

Кожна з вищевказаних характеристик якості є функцією від декількох підхарактеристик якості:

$$Fs = f_1(qsch_1, qsch_2, qsch_3) = f_1(FCom, FCor, FAppr), \quad (2)$$

де $FCom$ – функційна повнота, $FCor$ – функційна коректність, $FAppr$ – функційна доцільність;

$$Pe = f_2(qsch_4, qsch_5, qsch_6) = f_2(Tb, Ru, Cc), \quad (3)$$

де Tb – поведінка у часі, Ru – поведінка ресурсів, Cc – ємність (місткість);

$$Ub = f_3(qsch_7, ..., qsch_{12}) = f_3(Ar, Lb, Ob, Uep, Uia, Ab), \quad (4)$$

де Ar – розпізнавання доцільності, Lb – можливість вивчення, Ob – керованість, Uep – захист від помилок користувача, Uia – естетичність інтерфейсу користувача, Ab – доступність;

$$Rb = f_4(qsch_{13}, ..., qsch_{16}) = f_4(Mat, Avb, Ft, Rcv), \quad (5)$$

де Mat – зрілість, Avb – наявність (доступність), Ft – відмовостійкість, Rcv – відновлюваність;

$$Cb = f_5(qsch_{17}, qsch_{18}) = f_5(Ce, Ib), \quad (6)$$

де Ce – співіснування, Ib – взаємодія;

$$Scr = f_6(qsch_{19}, ..., qsch_{23}) = f_6(Conf, Int, Nr, Acb, Auth), \quad (7)$$

де $Conf$ – конфіденційність, Int – цілісність, Nr – невідхилюваність, Acb – підзвітність, $Auth$ – ідентичність;

$$Mb = f_7(qsch_{24}, ..., qsch_{28}) = f_7(Mod, Rub, Anb, Mdfb, Tsb), \quad (8)$$

де Mod – модульність, Rub – повторне використання, Anb – аналізованість, $Mdfb$ – модифікованість, Tsb – тестованість;

$$Pb = f_8(qsch_{29}, qsch_{30}, qsch_{31}) = f_8(Adb, Inb, Rpb), \quad (9)$$

де Adb – адаптованість, Inb – можливість інсталяції, Rpb – можливість заміни.

Таким чином, множина основних підхарактеристик якості ПЗ має вигляд:

$$QSCH = \{qsch_1, \ldots, qsch_{31}\} = \begin{Bmatrix} FCom, FCor, FAppr, Tb, Ru, Cc, Ar, Lb, Ob, Uep, Uia, \\ Ab, Mat, Avb, Ft, Rcv, Ce, Ib, Conf, Int, Nr, Acb, Auth, \\ Mod, Rub, Anb, Mdfb, Tsb, Adb, Inb, Rpb \end{Bmatrix}$$

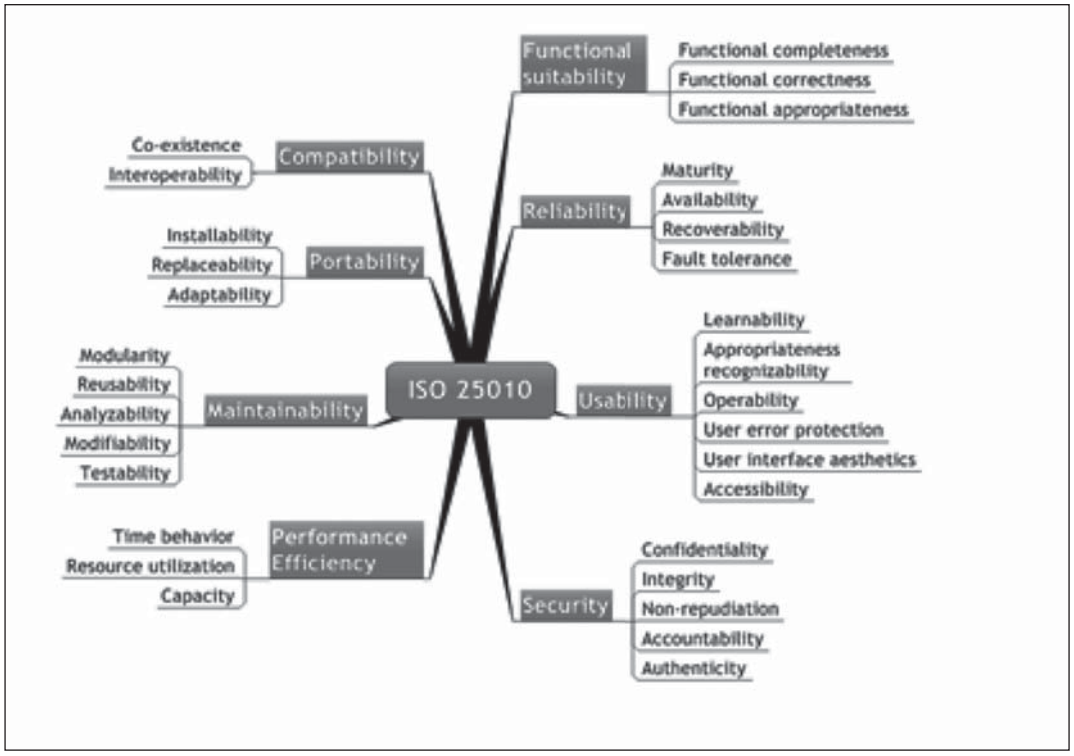

Рис. 1. Якість ПЗ за стандартом ISO/IEC 25010:2011 [9]

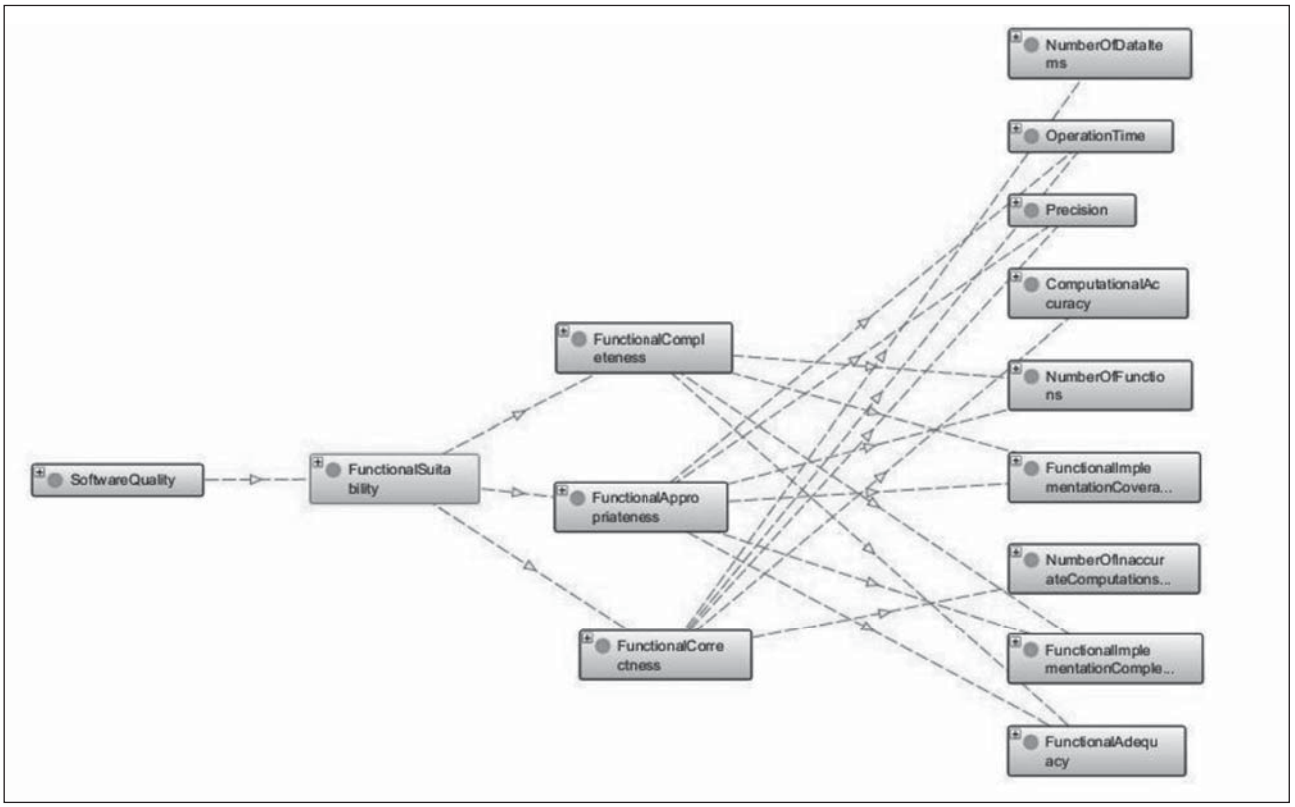

Рис. 2. Складова онтології для Функційної придатності

Водночас кожна підхарактеристика якості ПЗ є функцією певних атрибутів якості ПЗ, описаних у стандарті [10]. Множину атрибутів якості ПЗ представимо як: $QMS = \{qms_1, ..., qms_{138}\}$.

У [11] побудовано онтологію предметної галузі «Інженерія програмного забезпечення» (частина «Якість ПЗ»), яку утворюють складові частини для: Функційної придатності (рис. 2), Сумісності, Ефективності, Можливості переносу, Зручності використання, Надійності, Захищеності, Супроводжуваності.

Якість ПЗ може бути виражена не тільки функцією від підхарактеристик якості, але й середнім зваженим арифметичним показником з урахуванням значень атрибутів та коефіцієнтів їхньої вагомості. Отже, на основі формул (1)-(9), представимо якість програмного забезпечення у наступному теоретико-множинному вигляді:

$$\begin{gathered} Q = f(Fs, Pe, Ub, Rb, Cb, Scr, Mb, Pb) = \\ = f(f_1(FCom, FCor, FAppr), \\ f_2(Tb, Ru, Cc),\ f_3(Ar, Lb, Ob, Uep, Uia, Ab), \\ f_4(Mat, Avb, Ft, Rcv),\ f_5(Ce, Ib), \\ f_6(Conf, Int, Nr, Acb, Auth), \\ f_7(Mod, Rub, Anb, Mdfb, Tsb), \\ f_8(Adb, Inb, Rpb)) = \phi(w_i^m \cdot qms_i) \end{gathered} \quad (10)$$

де w_i^m – ваговий коефіцієнт i-го атрибуту якості ПЗ, $i = \overline{1, 138}$, $qms_i \in QMS$.

Аналіз характеристик CodeStyle. Взагалі, CodeStyle – це зовнішній вигляд коду. Стиль коду може бути надто детальним (наприклад, керівництво щодо стилю Java Script від Google) або більш загальним (як рекомендації щодо стилю ядра jQuery) [12]. Основною перевагою керівництв щодо стилю є те, що, знаючи, як повинен виглядати правильний код, набагато легше розпізнати неправильний код та потенційні помилки до того, як вони проявляться. CodeStyle дає рекомендації щодо певного візуального оформлення елементів програми з метою збільшення інформативності коду для людини. CodeStyle використовується для швидкого сприйняття коду, для запобігання помилок в коді, для швидкого написання коду, а також для зменшення часу та зусиль, витрачених на розроблення ПЗ.

Розглянемо, які ж характеристики декларує CodeStyle: 1) консистентність (однаковість форматування) коду – *cc*; 2) гарний синтаксис – *gs*; 3) перевірка типів – *tv*; 4) коментування коду – *ccom*; 5) зрозуміле та правильне іменування ідентифікаторів змінних та функцій (наприклад, запис типу змінної в її ідентифікаторі – за угорською нотацією) – *ccn*; 6) мінімізація (оптимізація) розміру коду (наприклад, за допомогою використання лаконічних конструкцій коду) – *mcs*; 7) мінімізація вкладених операторів розгалуження та циклів – *mno*; 8) мінімізація циклічних залежностей між пакетами – *mcd*; 9) висока зв'язність всередині методів – *hch*; 10) одноманіття коду (подібний код легше сприймається) – *mngc*; 11) «розрядка» коду (за допомогою пробілів) – *dcc*; 12) застосування правила вертикалі (частини одного запиту розташовуються на одному відступі, на одній вертикалі) – *arv*; 13) використання операторних дужок «початок-кінець» – *uob*; 14) коректна розстановка дужок (некоректно поставлені дужки призводять до логічних помилок) – *cpb*; 15) відсутність дублювання коду – *ndc*; 16) розбиття (структурування) коду на відносно незалежні блоки – *scb*; 17) документування коду (документація = читабельність, простота підтримки та використання) – *docc*; 18) групування (узагальнення сусідніх елементів коду) та організація коду – *goc*; 19) мінімізація навантаження на пам'ять та зір при читанні програми – *mmvl*; 20) лаконічні та зрозумілі мітки елементів керування – *lucl*; 21) послідовність коду – *csq*; 22) зрозумілість (ясність, спрощення сприйняття) програмного коду – *clc*; 23) гарантування легкості супроводу – *gem*; 24) спрощення процесу внесення подальших змін (легкість модифікації) – *em*; 25) покриття коду – $c\,\mathrm{cov}$; 26) очевидність (візуалізація) помилок – *eo*; 27) попередження помилок – *ep*; 28) безаварійне розгортання ПЗ – *tfsd*; 29) забезпечення стійкості програми – *pss*; 30) забезпечення можливості підтримки проекту декількома розробниками – *psd*.

Визначення впливу характеристик CodeStyle на характеристики якості програмного забезпечення. Проаналізувавши характеристики CodeStyle, можна зробити висновок про вплив певної характеристики CodeStyle на той чи інший атрибут якості. Побудуємо множину кортежів

$$CSCH = \{< csch_1, \{qcsm_k \mid k = 1..138\} >, ..., < csch_{30}, .\{qcsm_k \mid k = 1..138\} >\},$$

де $csch_i$ – i-та характеристика CodeStyle (з вищеперерахованих), $i = \overline{1, 30}$, $\{qcsm_k \mid k = 1..138\}$ – підмножина атрибутів якості, на які впливає i-та характеристика CodeStyle (тобто $qcsm_k \in QMS$), ця підмножина може бути порожньою множиною – в разі, якщо i-та характеристика CodeStyle не впливає на жоден атрибут якості.

Тоді:

$$CSCH = \{ < cc,\{Fusb,Usbio,Cmptd,Nac,Neum,Facs\} > ,$$
$$< gs,\{Fusb,Usbio,Cmptd,Nac,Neum,Facs\} > ,$$
$$< tv,\{Nofl,Noft,Ftr,Flr,Nbr,Niore,Ntre,Mote,Nio,Ndfe,De\} > ,$$
$$< ccom,\{Fusb,Usbio,Cmptd,Nac,Neum,Facs,Notut,Efud,Cmptud\} > ,$$
$$< ccn,\{Nofl,Noft,Ftr,Flr,Nbr,Fusb,Usbio,Cmptd,Nac,Neum,Facs\} > ,$$
$$< mcs,\{Ps,Nolcd\} > ,$$
$$< mno,\{Nofl,Noft,Ftr,Flr,Nbr\} > ,$$
$$< mcd,\{Nofl,Noft,Ftr,Flr,Nbr,Ntre,Mote,Ndfe,De\} > ,$$
$$< hch,\{Nofl,Noft,Ftr,Flr,Nbr,Ntre,Mote,Ndfe,De\} > ,$$
$$< mngc,\{Fusb,Usbio,Cmptd,Nac,Neum,Facs\} > ,$$
$$< dcc,\{Fusb,Usbio,Cmptd,Nac,Neum,Facs\} > ,$$
$$< arv,\{Fusb,Usbio,Cmptd,Nac,Neum,Facs\} > ,$$
$$< uob,\{Nofl,Noft,Ftr,Flr,Nbr,Fusb,Usbio,Cmptd,Nac,Neum,Facs\} > ,$$
$$< cpb,\{Nofl,Noft,Ftr,Flr,Nbr\} > ,$$
$$< ndc,\{Ps,Nolcd\} > ,$$
$$< scb,\{Nof,Fusb,Usbio,Cmptd,Nac,Neum,Facs,Ndfe,De\} > ,$$
$$< docc,\{Fusb,Usbio,Cmptd,Nac,Neum,Facs,Notut,Efud,Cmptud,Nuec,$$
$$Neusc,Nace,Disu,Tlrb,Nio,Ea,Nso,Nis,Eoi\} > ,$$
$$< goc,\{Nof,Ndfe,De\} > ,$$
$$< mmvl,\{Tlrb\} > ,$$
$$< lucl,\{Ps,Nolcd,Fusb,Usbio,Cmptd,Nac,Neum,Facs,Tlrb\} > ,$$
$$< csq,\{Fusb,Usbio,Cmptd,Nac,Neum,Facs,Tlrb\} > ,$$
$$< clc,\{Fusb,Usbio,Cmptd,Nac,Neum,Facs,Tlrb,Ntbm,Ntam\} > ,$$
$$< gem,\{Disu,Tlrb,Ntbm,Ntam,Ea,Nso,Nis,Eoi\} > ,$$
$$< em,\{Disu,Tlrb,Ntbm,Ntam,Ea,Nso,Nis,Eoi\} > ,$$
$$< ccov,\{Fic\} > ,$$
$$< eo,\{Nofl,Noft,Ftr,Flr,Nbr,Nuec,Neusc,Nace,Nio\} > ,$$
$$< ep,\{Nofl,Noft,Ftr,Flr,Nbr,Nuec,Neusc,Nace,Disu,Nio\} > ,$$
$$< tfsd,\{Nbr,Disu,Ea,Nso,Nis,Eoi\} > ,$$
$$< pss,\{Disu,Ea,Nso,Nis,Eoi\} > ,$$
$$< psd,\{Tlrb\} > \}$$

де *Fic* – покриття функційної реалізації, *Ps* – розмір програмного продукту, *Nofl* – кількість збоїв, *Noft* – кількість відмов, *Ftr* – усунення відмов, *Flr* – усунення збоїв, *Nbr* – кількість аварій, *Niore* – кількість помилок введення-виведення, *Ntre* – кількість помилок передачі даних, *Mote* – середня поява помилок передачі даних, *Nolcd* – безпосередня кількість рядків коду, *Nof* – кількість функцій, *Fusb* – зрозумілість функцій, *Usbio* – зрозумілість входів та виходів, *Cmptd* – повнота опису, *Nac* – кількість спроб налаштування, *Neum* – кількість легко зрозумілих повідомлень, *Facs* – фізична доступність, *Notut* – кількість тьюторіалів, *Efud* – ефективність документації користувача, *Cmptud* – повнота документації користувача, *Nuec* – кількість помилок користувача або змін, *Neusc* – кількість помилок, які користувач успішно виправляє, *Nace* – кількість спроб корекції помилок, *Disu* – ступінь підвищення задоволеності користувача, *Tlrb* – пристосованість, *Ntbm* – кількість проблем до модифікації, *Ntam* – кількість проблем після модифікації, *Nio* – кількість недозволених операцій, *Ndfe* – кількість форматів даних для обміну, *De* – обмін даними, *Ea* – адаптованість до навколишнього середовища, *Nso* – кількість операцій налаштування, *Nis* – кількість етапів інсталяції, *Eoi* – простота інсталяції.

Аналіз причинно-наслідкових зв'язків у розробленій в [11] онтології предметної галузі «Інженерія програмного забезпечення» (частина «Якість ПЗ») та вищенаведеної моделі якості ПЗ (формула (10)) дала можливість на основі отриманих атрибутів визначити підхарактеристики та характеристики, на які впливають характеристики CodeStyle: підхарактеристики «функційна повнота», «функційна доцільність» характеристики «Функційна придатність»; підхарактеристики «зрілість», «відмовостійкість», «відновлюваність» характеристики «Надійність»; підхарактеристика «поведінка ресурсів» характеристики «Ефективність»; підхарактеристики «розпізнавання доцільності», «можливість вивчення», «керованість», «захист від помилок користувача», «естетичність інтерфейсу користувача» характеристики «Зручність використання»; підхарактеристики «модульність», «повторне використання», «аналізованість», «модифікованість» характеристики «Супроводжуваність»; підхарактеристики «конфіденційність», «цілісність» характеристики «Захищеність»; підхарактеристики «співіснування», «взаємодія» характеристики «Сумісність»; підхарактеристики «адаптованість», «можливість інсталяції», «можливість заміни» характеристики «Можливість переносу».

Висновки. Внутрішня якість ПЗ (як написано код) не повинна поступатись зовнішній якості (який функціонал реалізовано). Внутрішня якість забезпечує: зрозумілість (що допомагає легко виконати фіксацію помилок і легше включатись в роботу новим членам команди, легко підтримувати код); зменшення роботи в перспективі (кількість помилок зменшується, кількість часу на розбір коду скорочується, полегшується підтримка коду); адаптованість (вимоги замовників в сучасному світі змінюються часто і реагувати на зміни потрібно швидко).

Основною метою прийняття та використання CodeStyle є спрощення сприйняття програмного коду людиною, мінімізація навантаження на пам'ять та зір при читанні програми, досягнення такого стану, коли програміст достатньої кваліфікації міг би дати висновки про функцію, яку виконує конкретна ділянка коду, а в ідеалі – також визначити його коректність, вивчивши тільки цю ділянку коду, або мінімально вивчивши інші частини програми. Іншими словами, сенс коду, написаного з використанням CodeStyle, повинен

буди зрозумілим із самого коду без необхідності вивчати його контекст.

Проведений аналіз досліджень та публікацій показав, що читабельність та зрозумілість коду, а також його зовнішній вигляд, які значною мірою визначає CodeStyle, впливають на якість ПЗ. У статті проведено визначення впливу характеристик CodeStyle на характеристики якості ПЗ.

Проведене дослідження показало, що кожна характеристика CodeStyle впливає щонайменше на один атрибут якості. Крім цього, проведене дослідження дало можливість зробити висновок про те, що характеристики CodeStyle впливають на 35 (зі 138) атрибутів, на 22 (з 31) характеристики та на всі 8 характеристик якості ПЗ, а не тільки на зручність використання, як могло б здатись на перший погляд.

Список літератури:

1. Макконнелл С. Совершенный код. Мастер-класс. Москва, 2013. 896 с.
2. IEEE 1028-2008. IEEE Standard for Software Reviews and Audits. Geneva, 2008. 53 p.
3. Абельсон Х., Сассман Дж. Структура и интерпретация компьютерных программ. Москва, 2010. 608 с.
4. Kienle H. M., Kraft J., Nolte T. System-specific static code analyses: a case study in the complex embedded systems domain. Software Quality Journal. 2012. Vol. 2. Issue 2. Pp. 337–367.
5. Kemerer Ch. F., Paulk M. C. The Impact of Design and Code Reviews on Software Quality: An Empirical Study Based on PSP Data. IEEE Transactions on Software Engineering. 2009. Vol. 35. Issue 4. Pp. 534–550.
6. Lian XL., Fakhry A., Zhang L., Cleland-Huang J. Leveraging Traceability to Reveal the Tapestry of Quality Concerns in Source Code. Software and Systems Traceability: Proceedings of 2015 IEEE/ACM 8th International Symposium. (Florence, May 17, 2015). Florence (Italy), 2015. Pp. 50–56.
7. Khomh F. SQUAD: Software Quality Understanding through the Analysis of Design. Reverse Engineering: Proceedings of 16th Working Conference. (Lille, October 13-16, 2009). Lille (France), 2009. Pp. 303–306.
8. Sidorova N. N. Ontology-Drived Method Using Programming Styles. Інженерія програмного забезпечення. 2015. № 2. С. 19–28.
9. ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models. Geneva, 2011. 34 p.
10. ISO 25023:2016.Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Measurement of system and software product quality. Geneva, 2016. 45 p.
11. T. Hovorushchenko, O. Pomorova. Ontological Approach to the Assessment of Information Sufficiency for Software Quality Determination. CEUR-WS. 2016. Vol.1614. Pp. 332–348.
12. Zakas N. C. Why Coding Style Matters. URL: https://coding.smashingmagazine.com/2012/10/why-coding-style-matters/ (дата звернення: 25.01.2018).

ОПРЕДЕЛЕНИЕ ВЛИЯНИЯ ХАРАКТЕРИСТИК CODESTYLE НА ХАРАКТЕРИСТИКИ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В статье исследуется влияние характеристик CodeStyle на характеристики качества программного обеспечения. Проведенное исследование позволило сделать вывод о том, что характеристики CodeStyle влияют на все восемь характеристик качества ПО, а не только на удобство его использования.

Ключевые слова: *программное обеспечение (ПО), качество ПО, характеристики качества ПО, стандарт ISO 25010:2011, CodeStyle.*

IDENTIFYING THE EFFECT OF CODESTYLE CHARACTERISTICS ON SOFTWARE QUALITY CHARACTERISTICS

The article examines the impact of CodeStyle characteristics on software quality characteristics. The conducted research provide the conclusion that CodeStyle characteristics effect on all eight software quality characteristics, and not just of its usability.

Key words: *software, software quality, characteristics of software quality, standard ISO 25010:2011, CodeStyle.*