12. Platt, John; Cristianini, N.; and Shawe-Taylor, J. (2000). "Large margin DAGs for multiclass classification". In Solla, Sara A.; Leen, Todd K.; and Müller, Klaus-Robert; eds. Advances in Neural Information Processing Systems. MIT Press. pp. 547–553.
13. Press, William H.; Teukolsky, Saul A.; Vetterling, William T.; Flannery, B. P. (2007). "Section 16.5. Support Vector Machines". Numerical Recipes: The Art of Scientific Computing (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.
14. Aizerman, Mark A.; Braverman, Emmanuel M.; and Rozonoer, Lev I. (1964). "Theoretical foundations of the potential function method in pattern recognition learning". Automation and Remote Control 25: 821–837.
15. Hsu, Chih-Wei; Chang, Chih-Chung; and Lin, Chih-Jen (2003). A Practical Guide to Support Vector Classification. Department of Computer Science and Information Engineering, National Taiwan University.

Рецензія/Peer review : 18.10.2013 р. Надрукована/Printed :24.11.2013 р.
Рецензент: Шалапко Ю.І., д.т.н., проф.

УДК 004.891.3: 004.3

Т.О. ГОВОРУЩЕНКО, А.В. КРАСІЙ
Хмельницький національний університет

ВИЗНАЧЕННЯ ХАРАКТЕРИСТИК ТА ВИБІР МОДЕЛІ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ АНАЛІЗУ СПЕЦИФІКАЦІЙ

Дана робота доводить важливість та можливість прогнозування характеристик ПЗ на ранніх етапах життєвого циклу, а також показує можливість вибору прийнятної моделі життєвого циклу на основі аналізу специфікацій.

У роботі описано всі основні характеристики ПЗ, досліджено формат специфікації вимог до програмного забезпечення та визначено, яку інформацію для визначення (прогнозування) характеристик можна отримати зі специфікації вимог до ПЗ.

Автори провели аналіз існуючих автоматизованих засобів оцінювання характеристик ПЗ та визначили їх неспроможність надавати прогнозовані кількісні значення характеристик ПЗ на основі аналізу специфікацій.

У статті наведено приклад вибору прийнятної моделі життєвого циклу програмного проекту на основі аналізу специфікацій, який доводить можливість прийняття таких рішень на ранніх етапах життєвого циклу на основі ретельного аналізу специфікацій.

Ключові слова: програмне забезпечення (ПЗ), програмний проект, характеристики ПЗ, специфікація вимог до ПЗ, модель життєвого циклу ПЗ.

T. O. HOVORUSHCHENKO, A. V. KRASIY
Khmelnytskiy National University

THE DEFINING OF CHARACTERISTICS AND THE SELECTING OF SOFTWARE LIFE CYCLE MODEL BASED ON THE SPECIFICATIONS ANALYSIS

Abstract - This work proves the importance and possibility of software characteristics prognosis at the early life cycle stages, and proves the possibility of the selecting an appropriate software life cycle model on the basis of the specifications analysis.

In this paper, all main software characteristics are described, format of software requirements specification is analysed and information to defining (prognosis) characteristics ,that can be obtained from the software requirements specification, is defined.

The authors conducted the analysis of known automated tools for software characteristics evaluation and identified their inability to provide the prognosis quantitative values of software characteristics on the basis of software specifications analysis.

The example of selecting acceptable software project life cycle model based on an specifications analysis is in the article. This example proves the possibility of such decision-making at the early life cycle stages on the basis of thorough analysis of software requirements specifications.

Keywords: software, software project, software characteristics, software requirements specification (SRS), software life cycle model.

Вступ

Розроблення програмного забезпечення (ПЗ) – це діяльність, яка вимагає детального вивчення предметної області та повного розуміння цілей розроблюваного продукту [1].

Специфікація вимог до програмного забезпечення (Software Requirements Specification – SRS) – це основа для побудови ПЗ. Вона включає множину функціональних і нефункціональних (додаткових) вимог. Функціональні вимоги описують всі взаємодії користувача з програмним забезпеченням, нефункціональні – накладають обмеження на проект чи реалізацію [2].

Програмний проект – це комплекс взаємозв'язаних заходів, спрямованих на досягнення поставлених задач з чітко визначеними цілями протягом заданого періоду часу та при встановленому бюджеті [3].

Процес розроблення ПЗ тісно пов'язаний з процесом аналізу та оцінювання значущих характеристик ПЗ. До характеристик ПЗ належать: вартість ПЗ, тривалість життєвого циклу ПЗ, модель життєвого циклу ПЗ, ефективність ПЗ, простота або складність ПЗ (найважливіша характеристика ПЗ з точки зору розробника), зручність використання ПЗ, кросплатформеність ПЗ, захист ПЗ, повнота реалізації вимог, обсяг файлів ПЗ, вимоги до системного програмного забезпечення та технічних засобів, обсяг потрібної оперативної та дискової пам'яті, а також безумовно надійність та якість ПЗ (найважливіші характеристики з точки зору користувача).

Наразі все помітнішою стає криза у галузі розроблення ПЗ – великі проекти виконуються з відставанням від графіка або з перевищенням кошторису витрат, розроблений продукт не має необхідних функціональних можливостей, продуктивність його низька, якість програмного забезпечення не влаштовує споживачів. При наявності ряду методів та засобів, залученні кращих фахівців для розроблення технологій та стандартів

проектування та розроблення ПЗ, процес розроблення ПЗ, як і раніше, залежить від знань та досвіду розробників.

Сучасне ПЗ характеризується: складністю опису; наявністю сукупності тісно взаємодіючих компонентів з локальними завданнями та цілями; відсутністю повних аналогів, а відтак і відсутністю можливості використання будь-яких типових проектних рішень; необхідністю інтеграції наявного і розроблюваного ПЗ; функціонуванням у неоднорідному середовищі на різних апаратних платформах; відокремленістю та різнорідністю окремих груп розробників із різним рівнем кваліфікації; великими термінами виконання проекту [4].

За наближеними оцінками Standish Group International, витрати на розроблення ПЗ складають близько 275 мільярдів доларів на рік, але лише 72% програмних проектів досягають етапу впровадження і всього 26% програмних проектів завершуються успіхом, тобто лише 71,5 мільярд доларів витрачається на успішні проекти, а решта 200 мільярдів витрачаються на провальні або незавершені проекти (рис. 1). За статистикою, 18% програмних проектів ніколи не завершуються; 53% проектів по розробленню ПЗ завершуються з перевитратами на 56% і перевищенням термінів на 84%; і лише 29% проектів вкладаються в терміни та бюджет.

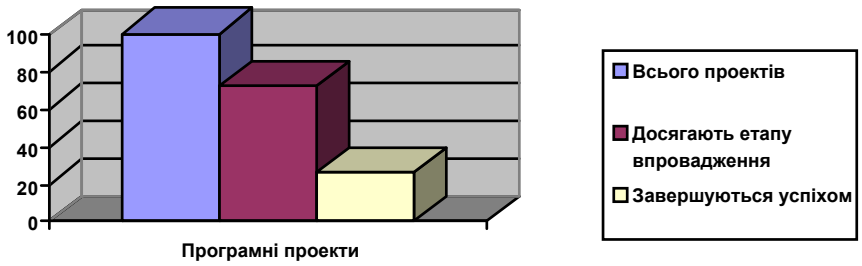

Рис. 1. Статистика щодо впровадження програмних проектів згідно оцінок Standish Group International

Програмні проекти часто зазнають невдач через помилки на ранніх етапах життєвого циклу ПЗ, а саме [5]: 1) неадекватне формулювання вимог; 2) невдале проектування або неефективне планування; 3) невірне розуміння або недостатній аналіз специфікації та проекту; 4) нереалістичні проектні плани щодо вартості проекту, щодо тривалості проекту, щодо тривалості життєвого циклу ПЗ в цілому, щодо ефективності розроблюваного за проектом ПЗ, щодо простоти та зручності використання розроблюваного ПЗ, щодо кросплатформеності розроблюваного ПЗ; 5) невірно обрана модель життєвого циклу.

Помилки формулювання вимог та проектування складають 25–55% всіх помилок, причому чим більший обсяг ПЗ, тим більше помилок вноситься саме на ранніх етапах [5], рис. 2. Слід врахувати й факт, що вартість виправлення помилки проектування в 2–4 рази вища вартості виправлення помилки конструювання. Залежність вартості виправлення помилок від етапу ЖЦ ПЗ наведено у [6].

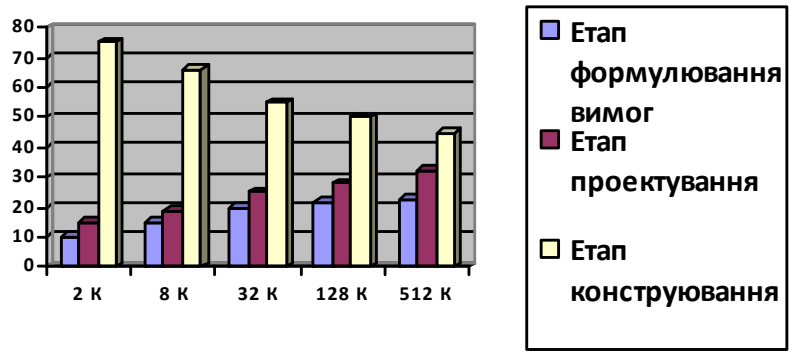

Рис. 2. Розподіл помилок, припущених на різних етапах життєвого циклу

Критичний вплив на програмні проекти та на успішність їх завершення здійснюють питання, пов'язані із аналізом специфікації. Отже, аналіз специфікації з метою одержання прогнозованих оцінок вартості, тривалості, ефективності, простоти, зручності використання, кросплатформеності, якості, надійності ПЗ, з метою прогнозування життєвого циклу ПЗ та вибору прийнятної моделі життєвого циклу ПЗ є *актуальним* на початку етапу проектування.

Основні характеристики ПЗ

Вартість ПЗ може бути обчислена за формулою:

$$ВАРТІСТЬ = LOC_{очікувана} \times ВАРТІСТЬ_{рядка}, \quad (1)$$

де $LOC_{очікувана}$ – очікувана кількість рядків коду, яку можна отримати зі специфікації.

Вартість рядка є константою і не змінюється від реалізації до реалізації; вимірюється у гривнях або доларах США. За статистикою *максимальною очікуваною вартістю розроблення ПЗ* в доларах США вважають кількість рядків сирцевого коду, поділену навпіл, тобто вартість рядка приймається рівною 0,5 доларів США. Тоді формула (1) матиме вигляд:

$$ВАРТІСТЬ_i = 0{,}5 \times LOC_{оч_i}, \quad (2)$$

Тривалість життєвого циклу ПЗ – послідовність фаз проекту, що задається виходячи з потреб управління проектом. Згідно моделі, запропонованої Боемом [7], тривалість проекту зростає експоненційно разом

з докладеними до проекту зусиллями, однак в даному випадку значення експоненти менше 1 і складає близько 0.35. Тоді *прогнозована оцінка тривалості проекту за моделлю Боема* обчислюється за формулою:

$$Тривалість = c \times Трудовитрати^{d} = c \times (a \times KLOC^{b})^{d} = c \times a^{d} \times KLOC^{b \times d}, \quad (3)$$

де c, d – коефіцієнти COCOMO [7]: для органічного (самостійного) програмного проекту $c = 2{,}5; d = 0{,}38$; для вбудованих програмних проектів (інтеграція апаратного та програмного забезпечення) $c = 2{,}5; d = 0{,}32$; для проміжних програмних проектів (не органічних, але й не жорстко вбудованих) $c = 2{,}5; d = 0{,}35$;

$KLOC$ – очікувана кількість тисяч рядків коду;

a, b – коефіцієнти COCOMO [7]: для органічного програмного проекту $a = 2{,}4; b = 1{,}05$; для вбудованих програмних проектів $a = 3{,}6; b = 1{,}20$; для проміжних програмних проектів $a = 3{,}0; b = 1{,}12$.

Модель життєвого циклу ПЗ – це структура, що складається із процесів, робіт та задач, які включають в себе розроблення, експлуатацію і супровід програмного продукту; охоплює життя системи від визначення вимог до неї до припинення її використання. Модель життєвого циклу ПЗ залежить від специфіки, масштабу і складності проекту та особливостей умов, за яких ПЗ створюється та функціонує. У модель життєвого циклу обов'язково включаються процеси реалізації робіт і завдань, що забезпечують створення проміжного продукту і перехід до наступного процесу моделі. При виборі схеми моделі ЖЦ для конкретної предметної галузі вирішуються питання включення важливих для створюваного продукту видів робіт або не включення несуттєвих робіт. Наразі найбільш відомими та використовуваними є такі моделі життєвого циклу ПЗ: каскадна, спіральна, інкрементна та модель еволюційного прототипування (RAD).

Ефективність ПЗ – відношення рівня послуг, що надаються програмним продуктом користувачу за заданих умов, до обсягу використовуваних ресурсів. Для оцінювання ефективності ПЗ використовують показники рентабельності, функційно-вартісний аналіз та сукупну вартість володіння. Під методами обчислення рентабельності розуміють всі підходи, пов'язані з виразом очікуваного ефекту у грошовій формі, а також із використанням співвідношення грошових витрат та результатів в якості критерію для відбору програмних проектів [8]. Ефективність ПЗ буває двох типів: часова та ефективність пам'яті. Особливо на ефективність ПЗ впливає вибір структури та представлення даних. Але й вибір алгоритмів, використовуваних у різних програмних модулях, а також особливості їх реалізації (включаючи вибір мови програмування) можуть істотно впливати на ефективність ПЗ. При цьому постійно доводиться розв'язувати задачу суперечливості між часовою ефективністю та ефективністю пам'яті. Тому дуже важливо, щоб у специфікації було чітко вказано кількісне співвідношення між показниками цих примітивів або хоча б задано кількісні межі для одного з цих показників.

Простота або *складність ПЗ* – це найважливіша характеристика ПЗ з точки зору розробника. *Програмна складність* характеризується довжиною програми або обсягом пам'яті ЕОМ, необхідної для розміщення ПЗ [9]. *Структурна складність програм* визначається кількістю взаємодіючих компонентів, кількістю зв'язків між компонентами та складністю їх взаємодії [9]. *Складність програмних модулів* характеризується конструктивною складністю створення оформленої компоненти програми і оцінюється з позиції складності внутрішньої структури та перетворення змінних в кожному модулі, а також інтегрально за деякими зовнішніми статичними характеристиками модулів [9]. Складність програми для систем реального часу переважно визначається допустимим часом відгуку, а для інформаційних систем – кількістю типів оброблюваних змінних [9]. Існує велика кількість метрик складності ПЗ [1], але найбільш популярною є *метрика Мак-Клура* – метрика, спрямована на оцінювання архітектури системи; міра складності, заснована на кількості можливих шляхів виконання програми, кількості керуючих конструкцій і змінних [1]. Вона обчислюється в 3 етапи:

1) для кожної керуючої змінної i обчислюється значення її функції складності :

$$C(i) = (D(i) * J(i)) / n, \quad (4)$$

де $D(i)$ – величина, яка вимірює сферу дії змінної (для локальної змінної $D(i) = 0$, а для глобальної змінної – $D(i) = 1$);

$J(i)$ – міра складності взаємодії модулів через цю змінну (скільки разів модулі взаємодіяли з використанням цієї змінної);

n – кількість модулів;

2) для всіх модулів визначаються функції складності:

$$M(P) = fp * X(P) + gp * Y(P), \quad (5)$$

де fp, gp – відповідно кількість модулів, безпосередньо передуючих і безпосередньо слідуючих за модулем P;

$X(P)$ – складність звертання до модуля P (скільки разів відбувається звертання до модуля P);

$Y(P)$ – складність управління викликом з модулю P інших модулів (скільки разів модуль P викликає інші модулі);

3) загальна складність MP програмної системи:

$$MP = \sum_{P} M(P). \quad (6)$$

Зручність використання ПЗ (usability) – це перевірка інтерфейсу на ефективність людино-машинної взаємодії; сукупність характеристик програмного продукту, які: 1) дозволяють мінімізувати зусилля користувачів з підготовки початкових даних, застосування програмного продукту і оцінки отриманих результатів; 2) дозволяють викликати позитивні емоції певного користувача. Зручність використання повинна бути частиною

проекту і проходити тестування в процесі проектування та розроблення. Вона повинна мати операційне визначення (щоб її можна було виміряти) та забезпечувати тестування. Зручність використання ПЗ – це показник його якості, який визначає кількість зусиль, необхідних для вивчення принципів роботи з ПЗ з використанням пропонованого інтерфейсу. Отже, зручність використання визначає ступінь простоти доступу користувача до функцій системи, наданих посередництвом інтерфейсу. Міжнародна організація стандартизації (ISO) дає наступне визначення [10]: "Зручність використання – це ефективність, рентабельність та задоволення, з яким користувачі можуть виконати ті чи інші задачі в заданому середовищі". Тестування на зручність використання проводиться для того, щоб оцінити якість роботи програмного продукту і вияснити, наскільки він ефективний, рентабельний та чи задоволені ним користувачі. Методи оцінки зручності використання: 1) кількісні – методи оцінки функцій, які передбачають підрахунок дій, визначення повноти виконання задачі, підрахунок часу, помилок та звернень по допомогу; 2) якісні – суб'єктивні методи, які передбачають збирання усних та письмових повідомлень користувачів про їх сприйняття, думки, судження, переваги, а також ступінь задоволеності системою. Бут (Booth) виявив 4 фактори, що складають зручність використання [10]: корисність, ефективність, простота вивчення та відношення користувача. Шекель (Shackel) теж розбиває зручність використання на 4 схожих категорії [10]: простота вивчення, ефективність, гнучкість та відношення користувача. На зручність використання ІК впливають фактори [10], представлені на рис. 3. Легкість навчання показує, чи швидко користувач вчиться використовувати систему. Ефективність навчання показує, як швидко користувач працює після навчання. Запам'ятовування навчання показує, чи легко запам'ятовується все, чому користувач навчився. Частота помилок показує частоту появи помилок під час роботи користувача. Загальна задоволеність показує, чи є загальне враження від роботи із системою позитивним.

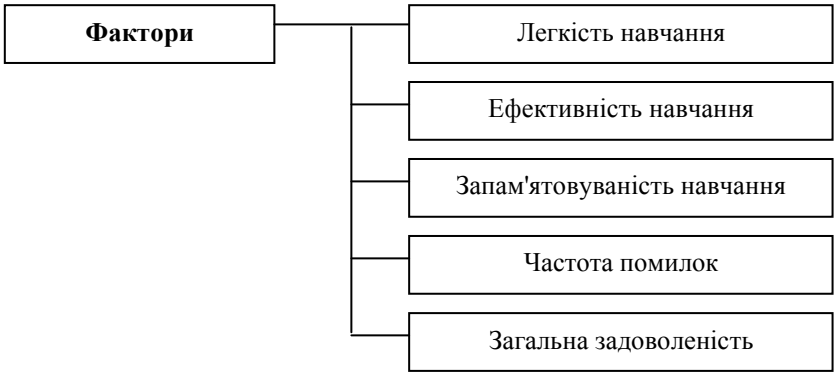

Рис. 3 . Фактори, які впливають на зручність використання ІК

Кросплатформеність ПЗ – це здатність програмного забезпечення працювати більш, ніж на одній апаратній платформі та/або операційній системі.

Якість ПЗ – здатність програмного продукту підтвердити свою специфікацію за умови, що специфікація орієнтована на характеристики, які бажає отримати користувач. Якість ПЗ – це характеристика ПЗ, яка відображає ступінь його відповідності вимогам [1]. Основним стандартом якості в області інженерії ПЗ є стандарт [11], який визначає номенклатуру, атрибути і метрики вимог якості ПЗ. Взагалі, якість ПЗ за стандартом [11] залежить від восьми основних характеристик – функційної придатності, ефективності, сумісності, зручності використання, надійності, безпеки, супроводжуваності, можливості переносу. У роботі [1] були визначені основні *метрики якості ПЗ*, а також описані методи опрацювання значень метрик з метою одержання прогнозованого кількісного значення якості ПЗ на ранніх етапах життєвого циклу.

Надійність ПЗ – здатність програмного продукту безвідмовно виконувати певні функції при заданих умовах протягом заданого періоду часу з досить великою ймовірністю; властивість програми виконувати задані функції в заданих умовах роботи і на заданій ЕОМ. Надійність ПЗ рекомендується характеризувати рівнем завершеності (відсутності помилок), стійкістю до помилок та можливістю перезапуску. Надійність ПЗ – сукупність властивостей, яка характеризує здатність програмного засобу зберігати заданий рівень придатності в заданих умовах протягом заданого інтервалу часу. *Показники надійності ПЗ* [12]: 1) критерій тривалості напрацювання на відмову – вимірюється час роботоздатного стану системи між двома послідовними відмовами або початком нормального функціонування системи після них; 2) тривалість відновлення – враховує можливість багаторазових відмов та відновлень; 3) коефіцієнт готовності – відображає ймовірність мати відновлювану систему в роботоздатному стані в довільний момент часу; значення коефіцієнту готовності відповідає долі часу корисної роботи системи на достатньо великому інтервалі, який містить відмови і відновлення; 4) напрацювання на ситуацію відмови, або стійкість – значення тривалості між втратами роботоздатності ПЗ незалежно від того, наскільки швидко відбулось відновлення. Визначення кількісних значень показників надійності ПЗ забезпечують аналітичні та емпіричні моделі надійності ПЗ – математичні моделі, побудовані для оцінювання залежності надійності ПЗ від деяких певних параметрів [12].

Аналіз формату специфікацій

При розробленні ПЗ софтверні організації зобов'язані керуватись стандартами як щодо процесів розроблення, так і щодо процесів оцінювання та забезпечення якості. Саме стандарти забезпечують зв'язок між розробниками та користувачами та повинні містити у собі попередній досвід розробників.

Існує велика кількість специфікацій і стандартів по їх написанню. До основних специфікацій належать: SFS (Software Functional Specification), SRS (Software Requirements Specification). *Функціональна специфікація*

(SFS) [6] описує необхідні характеристики системи (функціональність). Документація описує необхідні для користувача системи вхідні і вихідні параметри. Функціональна специфікація складається з трьох частин: 1) описи зовнішнього інформаційного середовища; 2) визначення функцій, визначених на множині станів інформаційного середовища; 3) опис небажаних (виняткових) ситуацій, які можуть виникнути при виконанні програм. Дана специфікація, призначена для визначення основних характеристик проекту ПЗ, є корисною лише при визначенні складності та вартості ПЗ, оскільки описує лише функціональні вимоги, без зазначення якості, надійності і т. і.

Іншою, більш повнішою з точки зору визначення основних характеристик ПЗ, є *специфікація вимог до ПЗ (SRS)*. Вона описує як повинна вести себе системи в певній ситуації, які функції повинна виконувати.

Згідно [2], специфікація вимог до програмного забезпечення повинна складатись з таких розділів:

1. Вступ
    1.1. Цілі
    1.2. Опис термінів
    1.3. Очікувана аудиторія « Reading Suggestions »
    1.4. Масштаб проекту
    1.5. Посилання на джерела
2. Загальний опис
    2.1. Бачення продукту
    2.2. Функціональність
    2.3. Класи та характеристики користувачів
    2.4. Середовище функціонування
    2.5. Рамки , обмеження , правила і стандарти
    2.6. Документація для користувачів
    2.7. Допущення і залежності
3. Функціональність системи
    3.1. Функціональний блок X
        3.1.1. Опис і пріоритет
        3.1.2. Причинно-наслідкові зв'язки , алгоритми
        3.1.3. Функціональні вимоги
    3.2. Вимоги до зовнішніх інтерфейсів
        3.2.1. Інтерфейси користувача (UX)
        3.2.2. Програмні інтерфейси
        3.2.3. Інтерфейси обладнання
        3.2.4. Інтерфейси зв'язку і комунікації
    3.3. Нефункціональні вимоги
        3.3.1. Вимоги до продуктивності
        3.3.2. Вимоги до схоронності (даних)
        3.3.3. Критерії якості програмного забезпечення
        3.3.4. Вимоги до безпеки системи
4. Інші вимоги

Для одержання достовірної інформації зі специфікації потрібно точно знати, що вона є повною та містити несуперечливі відомості. Для визначення повноти та несуперечливості специфікації використовуватимемо CASE. Аналітик – це чи не єдиний наразі конкурентоздатний російський CASE-засіб функційного моделювання, який в змозі визначити повноту та несуперечливість специфікації.

Визначимо *кількісну інформацію*, яку можна одержати із кожного пункту запропонованої стандартної структури SRS: 1.2 – термін виконання, кваліфікація виконавця(ів), кількість виконавців; 1.3 – розмір цільової аудиторії; 1.4 – кількість компонентів системи, розмір кожного компонента, повторюваність; 2.4 – вартість використовуваних операційних систем, баз даних, компіляторів; 2.5 – вартість специфічних засобів розробки; 2.6 – розмір і вартість документації для користувачів; 2.7 – залежність від інших програмних засобів(вартість, складність, надійність); 3.1.1 – перелік всіх функційних вимог, вартість, складність, надійність окремо взятої вимоги; 3.1.2 – складність та надійність окремо взятого алгоритму; 3.1.3 – вартість помилки, реакція на помилку; 3.2.1 – кількість, вартість інтерфейсів користувача; 3.2.2 – кількість, вартість, надійність, складність програмних інтерфейсів; 3.2.3 – кількість, вартість, надійність, складність апаратних інтерфейсів; 3.2.2 – кількість, вартість, надійність, складність інтерфейсів зв’язку і комунікацій; 3.3.1 – перелік всіх нефункційних вимог, вартість, складність, надійність окремо взятої вимоги;

Якісна інформація: 1.3 – профілювання(цільова аудиторія); 2.3 – вміння користувачів працювати із аналогічними продуктами; 2.4 – складність операційних систем, баз даних, компіляторів; 3.1.1 – залежність між вимогами, суперечливість вимог; 3.3.3 – якість програмного забезпечення; 3.3.4 – якість безпеки програмного забезпечення;

Це неповний перелік усіх кількісних і якісних показників інформації, які можна одержати при аналізі специфікації. Із переліку видно, що деякі критерії тісно взаємозв'язані і деколи дублюються, отже, до оцінки певного показника потрібно підходити комплексно. Всю якісну інформацію потрібно перетворити у кількісну за допомогою залучення експертів, оскільки необхідні кількісні значення характеристик ПЗ можна отримати лише в результаті опрацювання саме кількісної інформації специфікації.

Автоматизовані засоби оцінювання характеристик ПЗ

Розглянемо існуючі на сьогодні автоматизовані засоби оцінювання характеристик ПЗ.

CaliberRM – це корпоративна система управління вимогами на етапі конструювання програмного забезпечення. Дана система розроблена з метою підвищення якості створюваних продуктів і призначена для

покращення взаємодії між учасниками проекту, спрощення аналізу впливів і процесу передачі інформації в сфері управління змінами вихідних вимог. Можливості CaliberRM: 1) комплексна система управління вимогами; 2) налаштування процесів управління вимогами; 3) сортування вимог і призначення пріоритетів; 4) управління вимогами. Недоліки: 1) робота із вимогами на етапі їх формулювання, а не з готовими вимогами на початку етапу проектування; 2) відсутні прив'язки до стандартів; 3) неможливість визначення характеристик програмного проекту на основі вимог.

IBM Rational DOORS – це додаток для управління вимогами, який дозволяє оптимізувати обмін інформацією про вимоги, перевірку їх виконання та спільну роботу з управління ними в масштабах організації та всього ланцюжка поставок.. Rational DOORS дозволяє збирати, трасувати, аналізувати і керувати змінами інформації і показувати відповідність вимог нормативам і стандартам. Недоліками є робота із вимогами на етапі їх формулювання, а не на основі готових вимог, а також неможливість визначення характеристик програмного проекту на основі вимог.

Очевидно, що розглянуті засоби не прийнятні для кількісного оцінювання характеристик ПЗ на основі аналізу специфікацій.

Вибір прийнятної моделі життєвого циклу ПЗ на основі аналізу специфікацій

Враховуючи завдання, що покладаються на модель життєвого циклу (ЖЦ), необхідно зробити правильний вибір процесів, їхніх завдань та дій для побудови моделі ЖЦ ПЗ, яка задовольнятиме концептуальну ідею проектованого ПЗ з урахуванням його складності та масштабу робіт.

У [3] описано алгоритм вибору прийнятної моделі життєвого циклу розроблення ПЗ для проекту: 1) аналіз таких категорій проекту: вимоги, тип проекту та ризики, колектив розробників, колектив користувачів; 2) відповіді на питання кожної категорії; 3) розташування категорій та питань за ступенем важливості відносно проекту, для якого обирається модель ЖЦ.

Очевидно, що для будь-якого програмного проекту визначальними категоріями при виборі моделі ЖЦ є саме вимоги, а також тип проекту та ризики, а колектив розробників та колектив користувачів є другорядними категоріями. Проаналізувати та дати відповіді на питання визначальних категорій дає можливість саме специфікація вимог до ПЗ.

Категорія вимог складається з питань відносно вимог, які висуває користувач до проекту, тобто відносно властивостей розроблюваного ПЗ. Категорія «тип проекту та ризики» є категорією, визначення якої виконується на етапі планування.

Для прикладу розглянемо проект Olive Insurance Client-Management System [13]. Експерти проаналізували специфікацію вимог до цього проекту і надали наступні відповіді на питання кожної категорії (таблиці 1, 2).

Таблиця 1

Вибір моделі ЖЦ на основі характеристик вимог

Питання	Каскадна	Спіральна	RAD	Інкрементна
Чи є вимоги добре відомими або такими, що легко визначаються?	Так	Ні	Так	Ні
Чи можуть вимоги наперед визначатись у циклі?	Так	Ні	Так	Так
Чи часто змінюватимуться вимоги в циклі?	Ні	Так	Ні	Ні
Чи потрібно демонструвати вимоги з метою визначення?	Ні	Так	Так	Ні
Чи потрібна перевірка концепції для демонстрації можливостей?	Ні	Так	Так	Ні
Чи будуть вимоги відображати складність програмної системи?	Ні	Так	Ні	Так
Чи володіє вимога функціональними властивостями на ранньому етапі?	Ні	Так	Так	Так

Таблиця 2

Вибір моделі ЖЦ на основі характеристик типу та ризиків проекту

Питання	Каскадна	Спіральна	RAD	Інкрементна
Чи буде проект ідентифікувати новий напрямок продукту для організації?	Ні	Так	Ні	Так
Чи буде проект мати тип системної інтеграції?	Ні	Так	Так	Так
Чи буде проект розширенням існуючої програмної системи?	Ні	Ні	Так	Так
Чи буде фінансування проекту стабільним протягом всього ЖЦ?	Так	Ні	Так	Ні
Чи очікується тривала експлуатація програмної системи в організації?	Так	Так	Ні	Так
Чи потрібен високий ступінь надійності?	Ні	Так	Ні	Так
Чи буде програмна система змінюватись із застосуванням непередбачених методів на етапі супроводу?	Ні	Так	Ні	Так
Чи є графік обмеженим?	Ні	Так	Так	Так
Чи є «прозорими» інтерфейсні модулі?	Так	Ні	Ні	Так
Чи доступні повторно використовувані компоненти?	Ні	Так	Так	Ні
Чи є достатніми ресурси (час, фінанси, інструменти, персонал)?	Ні	Так	Ні	Ні

В результаті опрацювання даних таблиць очевидним є вибір RAD-моделі ЖЦ ПЗ.

Висновки

Автори довели, що аналіз специфікації, особливо її частини, пов'язаної з вимогами, надає різнопланову кількісну та якісну інформацію (як пряму, так і непряму) для подальшого розрахунку вартості програмного проекту, тривалості проекту, тривалості життєвого циклу розроблюваного ПЗ, ефективності проекту та розроблюваного за проектом ПЗ, а також для вибору прийнятної моделі життєвого циклу ПЗ та формування висновків щодо простоти та зручності використання ПЗ, щодо його якості та надійності. Крім того, експерти після опрацювання специфікації можуть надати додаткову кількісну інформацію та висновки щодо кількісних оцінок якісної інформації, корисні для визначення перелічених вище характеристик ПЗ.

Зрозуміло, що необхідну інформацію для визначення зазначених характеристик ПЗ можна отримати лише, якщо специфікація буде достатньо інформативною. *Тоді однією із комплексних задач, які потрібно вирішити в ході даного дослідження буде*: 1) визначення критеріїв інформативності специфікації для визначення необхідних характеристик ПЗ; 2) розподіл доступних специфікацій на інформативні та неінформативні за визначеними критеріями; 3) визначення інформативності специфікацій, виконаних за наявними стандартами, щодо визначення необхідних характеристик ПЗ.

Наступною комплексною задачею, яку потрібно вирішити для визначення характеристик ПЗ на основі аналізу специфікацій, є: 1) визначення чіткості одержаної зі специфікації кількісної інформації; 2) критерії кількісного оцінювання експертами якісної інформації специфікації.

Ще однією, найважливішою, задачею дослідження є: 1) опрацювання одержаної зі специфікації та від експертів кількісної інформації з метою одержання прогнозованих значень таких характеристик ПЗ, як вартість проекту, тривалість проекту, тривалість життєвого циклу ПЗ, ефективність ПЗ (з точки зору простоти та зручності використання, якості та надійності ПЗ), а також з метою вибору прийнятної моделі життєвого циклу ПЗ; 2) обчислення на основі одержаних прогнозованих значень характеристик ПЗ інтегративного показника проекту для подальшого порівняння та вибору проектів.

В результаті проведеного дослідження було визначено необхідність раннього прогнозування важливих характеристик розроблюваного ПЗ на основі аналізу специфікацій. Аналіз формату специфікацій ПЗ дав змогу визначити *основні задачі для подальших досліджень*: 1) обґрунтоване визначення інформативності специфікацій; 2) визначення кількісної інформації, доступної у специфікації, та ступеня її чіткості; 3) визначення якісної інформації, доступної у специфікації, та критеріїв для її кількісного оцінювання експертами; 4) опрацювання одержаної кількісної інформації з метою одержання прогнозованих оцінок основних характеристик ПЗ; 5) обчислення на основі одержаних прогнозованих оцінок інтегративного показника проекту для порівняння та вибору проектів.

Для забезпечення можливості автоматизованого вибору прийнятної моделі ЖЦ ПЗ на основі аналізу специфікацій необхідно вирішити *наступні задачі*: 1) підвищити точність та однозначність відповідей – для цього визначити кількісні критерії для відповідей на запропоновані у [3] питання; 2) визначити кількісну інформацію, доступну у специфікації, яка необхідна при виборі прийнятної моделі ЖЦ ПЗ; 3) визначити доступну у специфікації якісну інформацію, яка необхідна при виборі прийнятної моделі ЖЦ ПЗ, а також критерії для її кількісного оцінювання експертами (кількісну експертну інформацію); 4) опрацювати одержану зі специфікації кількісну та кількісну експертну інформацію з метою одержання відповідей на запитання визначальних категорій проекту, і, відповідно, з метою вибору прийнятної моделі ЖЦ ПЗ – обрати метод чи компонент, який дасть можливість опрацювати зазначену інформацію із врахуванням її природи та взаємозв'язків та надати висновки щодо прийнятної моделі ЖЦ ПЗ.

Запропонований підхід дасть змогу прийняти мотивоване та обґрунтоване рішення щодо вибору проекту та його реалізації на основі різних важливих характеристик ПЗ, моделі життєвого циклу для розроблюваного програмного забезпечення на основі аналізу специфікацій вже на початку етапу проектування. На вирішення поставлених задач будуть спрямовані подальші зусилля авторів.

Література

1. Мищенко В.О., Поморова О.В., Говорущенко Т.А. CASE-оценка критических программных систем. В 3-х томах. Том 1. Качество / Под ред. Харченко В.С. – Харьков: НАУ «ХАИ», 2012. – 201 с.

2. IEEE 830-1998. Recommended Practice for Software Requirements Specifications - New York: IEEE, 1998

3. Роберт Т. Фатрелл, Дональд Ф. Шафер, Линда И. Шафер. Управление программными проектами: достижение оптимального качества при минимуме затрат.: Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 1136 с.

4. С.Макконнелл. Совершенный код. Мастер-класс - М.: Издательство "Русская редакция", 2013 - 896 с.

5. Поморова О.В., Говорущенко Т.О. Сучасні проблеми оцінювання якості програмного забезпечення // Радіоелектронні і комп'ютерні системи – Харків: НАУ "ХАІ", 2013 – № 5, с.319-327

6. IEEE 1031-2011: IEEE Guide for the Functional Specification // [Electronic resource] - Access mode: https://standards.ieee.org/findstds/standard/1031-2011.html

7. Boehm B.W. Software Engineering; R&D trends and defense needs. – In: Research. Directions in Software Technology (P.Wegner, ed.). – Cambridge, MA: MIT Press, 1979. – 543 pp.

8. Скрипкин К. Г. Экономическая эффективность информационных систем - М.: Радио и связь, 2003 - 156 с.

9. Ковалевская Е.В. Материалы к курсу "Метрология, качество и сертификация ПО" - М.: Московский государственный университет экономики, статистики и информатики, 2002. - 38 с. // [Електронний ресурс] - Режим доступу: http://bookinist.net/books/bookid-333504.html

10. Поморова О.В., Говорущенко Т.О. Проектування інтерфейсів користувача: Навчальний посібник -

Хмельницький: ХНУ, 2011 - 206 с.

11. ISO/IEC 25010:2011. Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models

12. В.А.Благодатских, В.А.Волнин, К.Ф.Поскакалов. Стандартизация разработки программных средств: Учебное пособие. - М.: Финансы и статистика, 2005. - 288 с.

13. Software Requirement Specifications [Full Version] // [Electronic resource] - Access mode: http://findebookee.com/s/software-requirement-specifications

References

1. Mishchenko V.O., Pomorova O.V., Hovorushchenko T.A. CASE-otsenka kriticheskih programmnih sistem. Tom 1. Kachestvo / Pod red. Kharchenko V.S. - Kharkov: NAU "KhAI", 2012 - 201 s. [in Russian]

2. IEEE 830-1998. Recommended Practice for Software Requirements Specifications - New York: IEEE, 1998

3. R.T.Futrell, D.F.Shafer, L.I.Shafer. Quality Software Project Management - New York: Prentice Hall PTR, 2003.

4. Steve McConell. Code Complete - New York: Microsoft Press, 2013

5. Pomorova O.V., Hovorushchenko T.O. Suchasni problemi otsinuvannya yakosti programnogo zabezpechennya // Radioelektronni i komp'uterni sistemi - Kharkiv: NAU "KhAI", 2013 - № 5, c.319-327 [in Ukrainian]

6. IEEE 1031-2011: IEEE Guide for the Functional Specification // [Electronic resource] - Access mode: https://standards.ieee.org/findstds/standard/1031-2011.html

7. Boehm B.W. Software Engineering; R&D trends and defense needs. – In: Research. Directions in Software Technology (P.Wegner, ed.). – Cambridge, MA: MIT Press, 1979. – 543 pp.

8. Skripkin K.G. Ekonomicheskaya effektivnost informatsionnih sistem - M.: Radio i svyaz, 2003 - 156 s.

9. Kovalevskaya E.V. Materiali k kursu "Metrologiya, kachestvo i sertifikatsiya PO" - M: MGUESI, 2002 - 38 s. // [Electronic resource] - Access mode: http://bookinist.net/books/bookid-333504.html [in Russian]

10. Pomorova O.V., Hovorushchenko T.O. Proektuvannya interfeisiv koristuvacha: Navchalniy posibnik - Khmelnitskiy,: KhNU, 2011 - 206 s. [in Ukrainian]

11. ISO/IEC 25010:2011. Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models

12. V.A.Blagodatskih, V.A.Volnin, K.F.Poskakalov. Standartizatsiya razrabotki programmnih sredstv: Uchebnoe posobie - M.: Finansi i statistika, 2005 - 288 s. [in Russian]

13. Software Requirement Specifications [Full Version] // [Electronic resource] - Access mode: http://findebookee.com/s/software-requirement-specifications

Рецензія/Peer review : 18.10.2013 р. Надрукована/Printed :24.11.2013 р.
Рецензент: Шалапко Ю. І., д.т.н., професор кафедри ІМ, ХНУ,

УДК 004.3:681.518

Д.Е. ИВАНОВ
Институт прикладной математики и механики НАН Украины, г.Донецк

МЕТОД ПОСТРОЕНИЯ ТЕСТОВ ЦИФРОВЫХ УСТРОЙСТВ С ПОДТВЕРЖДЕНИЕМ СОСТОЯНИЙ НА ОСНОВЕ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

В статье описывается метод построения тестов для цифровых последовательностных устройств на логическом уровне описания. Он основан на двухуровневом подходе, а особенностью является реализация нижнего уровня метода. Целью метода данного уровня является построение теста одной выбранной неисправности, а реализация состоит из двух фаз. Первая фаза выполняет структурное распространение влияния неисправности внутри одного такта модельного времени, при этом формируются ограничения на исходное состояние устройства. Подтверждение данного состояния выполняется с помощью разработанного генетического алгоритма. Применение комбинированного подхода построения тестов должно увеличить эффективность такой процедуры для сложных современных СБИС.

Ключевые слова: цифровое устройство, генерация тестов, генетический алгоритм, достижение состояния.

D.E. IVANOV
Institute of Applied Mathematics and Mechanics, National Academy of Sciences of Ukraine, Donetsk

TEST GENERATION METHOD FOR DIGITAL DEVICES WITH STATE SUBSTANTIATION BASING ON GENETIC ALGORITHM

Abstract – The aim of this paper is developing a test generation method for sequential digital devices at the logical level of description. Its main feature is the use of two phases for the construction of test of one selected fault.

The first phase of method performs structural fault propagation in one clock cycle of model time, which well developed for this time. At end of this phase are formed the restrictions to the initial state of the fault and fault-free devices. Substantiation of this state is performed by using the new developed genetic algorithm.

The use of such combined approach of test generation increases the effectiveness of this procedure for modern complex VLSI, which contains both data processing parts and controllers.

Keywords: digital device, test generation, genetic algorithm, state substantiation.

Введение

Согласно Международной Дорожной Карте [1] число транзисторов в SOC-системах потребительского уровня увеличится в 17 раз к 2024 году. С другой стороны, длина входных последовательностей для тестирования