## УПРАВЛІННЯ ПРОЕКТАМИ

УДК 004.051

**Запорізький національний технічний університет**
**Г.В.Табунщик, Д.С.Халіна**

# Засоби підвищення якості планування програмних проектів

*У статті виконаний аналіз методів підвищення якості планування програмних проектів. Авторами запропонований модифікований метод визначення диференційованої трудомісткості етапу розроблення, та доповнення до етапу формалізації загальносистемних характеристик. Розглянута авторська реалізація методів підвищення якості етапу планування програмних проектів.*

*В статье выполнен анализ методов повышения качества планирования программных проектов. Авторами предложен модифицированный метод определения дифференцированной трудоемкости этапа разработки, и дополнение к этапу формализации общесистемных характеристик. Рассмотрена авторская реализация методов повышения качества этапа планирования программных проектов.*

*In the article the analysis of quality improvement methods of program projects planning was carried out. Differentiated work content determination method of a development phase and supplement to system ejectives formalization stage was offered by the authors. The implementation of program projects quality improvement methods used at the design stage is considered in the article.*

**Ключові слова:** програмний проект, якість планування, диференційована трудомісткість.

**Вступ**

Підвищенні вимоги зі сторони замовників та користувачів до якості, надійності функціонування, а також до безпеки використання програмних продуктів, зумовлені швидким зростанням складності та розмірів сучасних інформаційних систем.

Для того, щоб відповідати цим вимогам необхідно здійснювати керування якістю на усіх етапах розроблення програмних продуктів: ініціації, плануванні, впроваджені, завершенні.

Основні роботи, присвячені забезпеченню якості програмного забезпечення, написані: В.В. Ліпаєвим [1, 2], В.В. Скляром [3], В.С. Харченко [4]. В області управління проектами необхідно відзначити праці С. Макконела [5, 6], Т. Демарко, Т. Листера [7], У. Ройса [8].

Серед найбільш відомих робіт, присвячених управлінню вимогами на етапі планування, слід зазначити: Д. Леффингуэлла [9], Д. Уідріга [9], Р. Росса [10], Д. Відріга [11], К. Вітнера, И. Спенса [12], К. Вінгерса [13].

В галузі тестування програмного забезпечення значимий вклад внесли: С. Канер [14,15], Д. Фолк [14,15], Г. Майєрс [16], В.В. Скляр [17], В.С. Харченко [17], К. Браун [18], Г. Кобб [18], Р. Колбертсон [18], К. Бек [19] та інші.

На кожній стадії проекту якість зростає, по-перше, як прямий наслідок зрілості процесу, по-друге, завдяки використанню проміжного продукту вищої якості, виробленого на попередній стадії. Під зрілістю процесу слід розуміти його здатність вирішувати поставлене завдання [20].

Також слід зазначити, що значення другої причини яка забезпечує нарощування якості в процесі життєвого циклу (ЖЦ) розробки програмного продукту для зрілих процесів виявляється набагато важливішим.

Складність продукту погіршує якість планування, у свою чергу нечіткість (двозначність) вимог погіршує якість реалізації, а погана структура продукту - якість самого продукту, що відображено на рис.1.

Але чим вище якість кожного процесу програмного проекту (ПП), тим вище якість планування, реалізації і власне підсумкового програмного продукту [21].

Отже, якість процесу планування гарантує якість підсумкового програмного продукту. Іншими словами, чим вище якість планування, тим вище якість розробленого в цьому процесі програмного забезпечення.

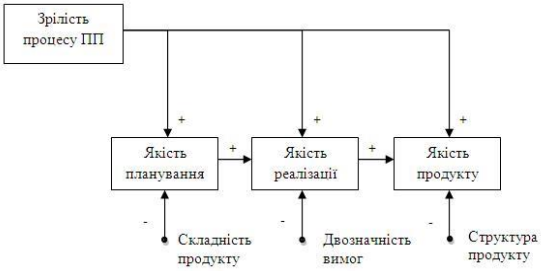


Рис.1. Концептуальна модель якості

Можна зробити наступні висновки:

- якість акумулюється і вклад, зроблений в якість на ранніх стадіях, має більший вплив на підсумковий програмний продукт, чим на завершуючих стадіях;

- вимір, оцінка якості повинні відбуватися на кожному етапі ЖЦ програмного продукту.

Тому актуальним є завдання управління якістю етапу планування ПП.

**Постановка завдання**

Під якістю планування розуміють величину відхилення реального часу на створення системи від прогнозованого значення [22].

Неякісне планування часто є причиною невдачі багатьох проектів. Адже успішність виконання всього проекту та можливі його проблеми й ризики можна передбачити вже на початкових етапах.

Часові витрати на розробку програмного забезпечення практично безпосередньо залежать від його розміру та трудомісткості розробки. При якісному плануванні розробник може оцінити тривалість проекту і ресурси, що потрібні для розробки програмної системи (ПС), знизити ризик помилки при визначенні передбачуваної вартості, ефективно управляти робочими бізнес-процесами та ресурсами компанії, аргументувати вартість і терміни перед замовником при проведенні переговорів і підписанні договору.

Для підвищення якості планування, зазвичай використовують: метод визначення диференційованої трудомісткості етапу розробки, метод формалізації загальносистемних характеристик і метод корекції плану проекту [23], а також модель COCOMO II [24].

**Метод визначення диференційованої трудомісткості етапу розробки складається з наступних кроків:**

*Крок 1.* Систему розподіляють на дерево вкладених підсистем.

*Крок 2.* В рамках кожної з підсистем визначається кількість елементів наступних типів: логічна сутність, екрани інтерфейсу користувачів, бізнес функції, звіти, інтерфейси із зовнішніми системами. При цьому кількість елементів різних типів визначається детерміновано, відповідно до градації їх складності.

*Крок 3.* Для кожного типа елементу та його складності заповнюється питома трудомісткість (людино-днів). Налаштування значень питомої трудомісткості може виконувати лише кваліфікований співробітник організації (архітектор, конструктор або керівник проекту) на основі інформації про кваліфікацію розробників, оскільки значення питомої трудомісткості мають визначальний вплив на результати оцінки.

Градація складності є оптимальною із значеннями 2 або 3, оскільки більша кількість рівнів складності викличе нелінійне збільшення трудовитрат на оцінку програмної системи (ПС), при незначному збільшенні точності. Наприклад:

1 рівень – мінімальна трудомісткість;

2 рівень - середня трудомісткість;

3 рівень – максимальна трудомісткість.

Для невеликих систем можливо і не потрібно розділення на підсистеми, тоді як для великих корпоративних ПС може потрібно розбиття на підсистеми з великим рівнем декомпозицій. Чим більше рівнів декомпозиції зроблено при оцінці, тим більше її точність.

*Крок 4.* Позначивши Nj, як загальну кількість елементів j-го типа в підсистемі, а tj – задану питому трудомісткість елементу j-го типа, отримаємо трудомісткість в підсистемі:

$$T_{підсистеми} = N_j * t_j,$$

де j=1..k, k – кількість підсистем.

І трудомісткість системи в цілому:

$$T_{системи} = \sum_{j=1}^{k} \sum_{i=1}^{n} N_{ij} * t_{ij},$$

де *i =1..n, n* – це кількість типів елементів в кожній підсистемі,

*tij* – питома трудомісткість елементу для різних підсистем.

Отримана трудомісткість системи дозволить оцінити витрати в людино-днях на розроблення програмного продукту.

Для планування з високою точністю, оцінки, отримані в результаті цього методу, коректуються на етапі формалізації впливу загальносистемних характеристик і за допомогою методу корекції плану проекту.

**Етап формалізації загальносистемних характеристик.**

Кожний проект з розроблення ПЗ має свої специфічні ознаки, які не відносяться безпосередньо до функціональних характеристик системи, що розробляється, але можуть в значній мірі вплинути на терміни та вартість розробки.

На першому кроці дані ознаки необхідно розділити на три основні групи: організаційні, функціональні й архітектурні. До першої групи можна віднести такі ознаки, як тип замовника, розмір структури замовника, переважний спосіб контакту, мову спілкування, вимоги до оформлення документації, необхідність роботи з третіми особами і тому подібне. До групи функціональних ознак можна віднести: розподілене

розміщення користувачів, кількість різних груп користувачів, міру забезпечення безпеки системи, комп'ютерну кваліфікацію користувачів, підтримку мультіязичності системи і тому подібне. Найбільш важливі з третьої групи ознак: середовище розробки, БД, архітектура системи, що використовується, міра швидкодії апаратних засобів, що використовуються для розробки, можливість багатократного використання кода, кількість робочих місць і тому подібне.

На другому кроці визначені ознаки оформляються у вигляді запитань з варіантами відповіді. Також можуть бути присутні й суб'єктивні запитання, відповіді на які не можуть бути формалізовані. Але, проте, їх присутність в анкеті є край доцільним, оскільки даний документ акумулює основні зовнішні умови та нефункціональні вимоги, і такі відповіді можуть допомогти не забути менеджерові проекту про істотні ризики.

Шаблон опитувального листа наведено в таблиці 1

*Таблиця 1 Приклад опитувального листа*

| Інтерв'юєр | | | Додаткова інформація | Питання | Варіанти відповідей |
|---|---|---|---|---|---|
| Менджер | Аналітик | Конструктор | | | |

У методі корекції плану проекту використовуються ознаки, які часто зустрічаються, і їх базові коефіцієнти впливу на кожне завдання плану проекту. Коефіцієнти впливу зручно задавати в матриці, в якій рядками є загальносистемні ознаки, а стовпцями – завдання плану проекту з розроблення програмного забезпечення.

Корекція плану проекту може бути виконана як в автоматичному режимі за допомогою розробленої програми, так і в напівавтоматичному режимі за допомогою додатка, який видасть менеджерові проекту необхідні рекомендації по кожному із завдань плану на основі наявних відповідей в запитальнику. Оскільки вплив загальних специфічних ознак на трудомісткість проекту хоча і формалізована, але все-таки залишається в деякій мірі суб'єктивною, напівавтоматичний режим часто може бути надійнішим, бо передбачає більшу міру участі експерта.

Для даного підходу характерні наступні недоліки: Використання запропонованого ділення системи на підсистеми і елементи приводить до труднощів при побудові плану проекту відповідно до існуючих моделей (MSF, RUP, Agile);

Неточність при підрахунку трудомісткості підсистеми, оскільки в рамках однієї підсистеми трудомісткість елементів j-го типа може розрізнятися;

Для визначення складності завдання проекту, використовується суб'єктивна думка експерта. Модифікований метод оцінки трудомісткості планування програмних проектів

При визначенні диференційованої трудомісткості, пропонується:

1. Здійснювати ділення системи на підсистеми і прості елементи шляхом ділення на етапи створення програмного продукту і для кожного етапу задавати його значення трудомісткості (таблиця 2)

*Таблиця 2. Приклад визначення трудомісткості для проекту заснованому на RUP*

| Назва етапу плану проекту | Трудомісткість (люд/дні) |
|---|---|
| Реалізація АС Ресторан | |
| Підсистема Бармен | 66 |
| 1.1 Логічна суть 1 | 15 |
| 1.2 Логічна суть 2 | 10 |
| 1.3 Бізнес функція 1 | 15 |
| 1.4 Звіт про наявність напоїв | 10 |
| 1.5 Екран замовлення | 16 |

2. Градацію складності задавати відносно тривалості проекту або фази. Оскільки проекти характеризуються різною тривалістю, то для визначення складності пропонується використовувати відносну величину:

Sвідн = тривалість/тривалість проекту

або для Agile, RUP-проектів:

Sвідн1 = тривалість/тривалість фази

Тоді доцільно використовувати такі значення складності:

1 рівень складності – 1 – 15%;

2 рівень складності – 15 – 20%;

3 рівень складності – 20 – 50%;

При формалізації загальносистемних характеристик рекомендується запитальник формувати на основі документа концепції системи, наприклад документа Vision (RUP, MSF, Agile, MDP)

*Таблиця 3. Питаня для формування документа Vision*

| П/п | Питання | Відповідний пункт документа Vision |
|---|---|---|
| 1 | Хто замовник? | Product Position Statement |
| 2 | Основна проблема, яка привела до вирішення створення програмної системи (суть проблеми)? | Problem Statement |
| 3 | До чого дана проблема приводить (наслідки)? | Problem Statement |
| 4 | Що покращується, якщо вирішити проблему (створити ПС)? | Problem Statement |
| 5 | Для кого призначена ПС? | Product Position Statement |
| 6 | Відмінності від аналогів? | Product Position Statement |
| 7 | Основні категорії користувачів ПС? | Stakeholder and User Descriptions: User Summary; Stakeholder Profiles |
| 8 | Основні стандарти, яким повинна відповідати ПС? | Other Product Requirements: Applicable Standards |

3. У методі корекції плану проекту, для зручності, значення коефіцієнтів впливу необхідно визначити від 1 до 5, де:
1) означає, що ознака – не впливає на завдання проекту;
2) слабкий вплив на завдання проекту;
3) помірний вплив на завдання проекту;
4) великий вплив на завдання проекту;
5) дуже великий вплив на завдання проекту.
Корекцію значень трудомісткості для кожного завдання плану проекту виробити відповідно до вираження:

$$K_{t_j} = \frac{\sum_{i=1}^{p} k_{ji}}{p},$$

де $K_{t_j}$ - сумарний коефіцієнт впливу на трудомісткість j-того завдання плану проекту;

$k_{ji}$ - коефіцієнт впливу і-тої ознаки на j-те завдання плану проекту;

p – кількість загальносистемних ознак.

Тоді значення трудомісткості буде розраховано по формулі:

$$t_{ji} = \begin{cases} 1, K_{t_j} = 1; \\ 1.25, K_{t_j} = 2; \\ 1.5, K_{t_j} = 3; \\ 1.75, K_{t_j} = 4; \\ 2, K_{t_j} = 5. \end{cases}$$

Всі ці зміни підвищать адаптивність, ефективність і точність оцінки трудомісткості ПП.

Реалізація методів підвищення якості програмних проектів

У розробленому додатку реалізовані метод визначення диференційованої трудомісткості етапу розробки, метод корекції плану проекту, а також етап формалізації загальносистемних характеристик.

На рис. 2 представлена діаграма програмних компонентів розробленої системи, де:

DiagWizard.exe – додаток, що реалізовує методи оцінки якості програмних проектів;

dataM1.xml – містить таблицю ділення системи на підсистеми і елементи з відповідними їм складністю і трудомісткістю. Даний файл створюється в результаті виконання методу визначення трудомісткості етапу розробки;

Elements.txt – файл зберігається автоматично після завершення роботи з першим методом, в ньому зберігаються елементи дедерева самого нижнього рівня, тобто ті які не мають дочірніх елементів;

dataM2.xml - містить шаблон запитальника для використання в методі формалізації загальносистемних характеристик з використанням шаблону. Зміна у файлі data.xml вплине на зміну даних при виконання програми DiagWizard.exe;

dataM2backup.xml – містить таблицю-запитальник. Даний файл створюється в результаті виконання етапу формалізації загальносистемних характеристик. У нім зберігається запитальник створений користувачем;

<имя>.xml – збережений користувачем файл із запитальником, що містить лише ті питання (характеристики), які йому необхідно буде використовувати в методі корекції плану проекту. <Ім'я> користувач задає самостійно

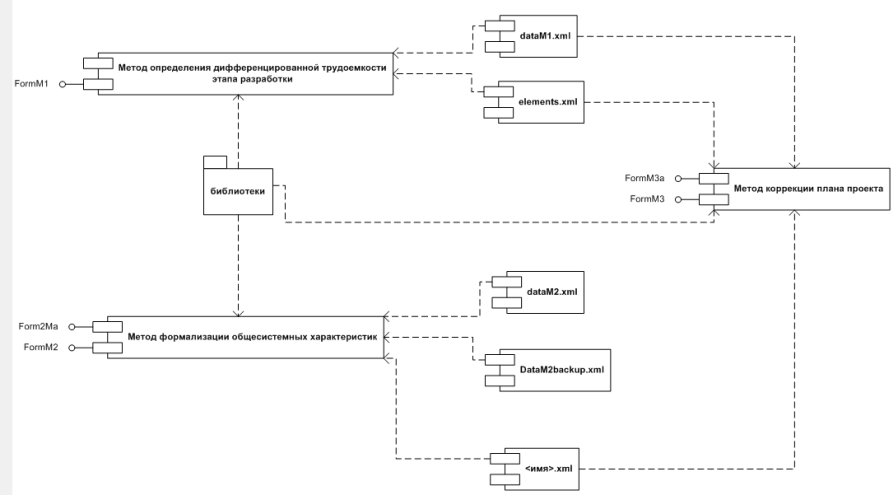


Рис. 2. Діаграма компонентів системи оцінки трудомісткості

Запропонована архітектура підсистеми дозволить інтегрувати з іншими системами управління програмними проектами (MS Project). Інтерфейс розроблених модулів зображений на рис. 3-6.

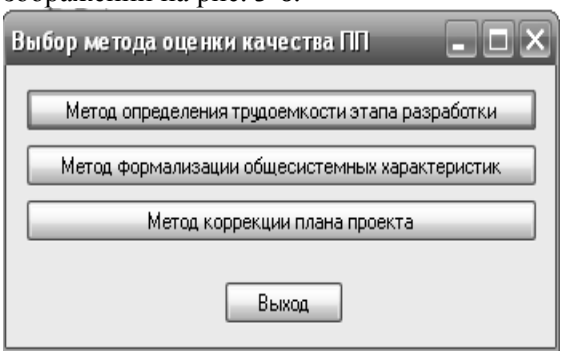


Рис. 3. Головне вікно програми

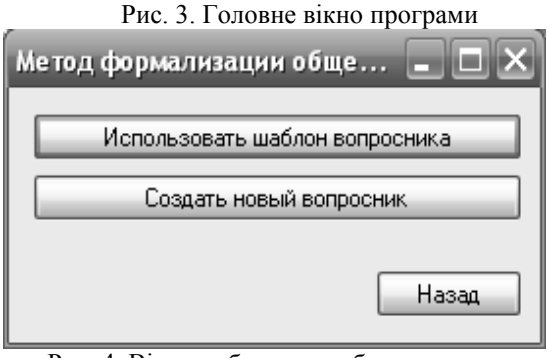


Рис. 4. Вікно вибору способу створення Запитальника

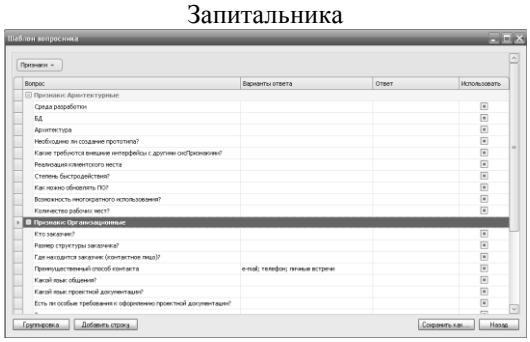

Рис. 5. Шаблон запитальника в розгорнутому вигляді

У розробленому додатку, приступаючи до методу корекції плану проекту, користувачеві необхідно ввести значення коефіцієнтів впливу ознак на завдання плану проекту. Після завершення введення, користувачеві необхідно підтвердити значення, що вводяться, натиснувши кнопку «Застосувати», після чого відкриється вікно, де для кожного елементу буде виведена підказка, в якій користувачеві буде дана інформація, як потрібно змінити значення трудомісткості елементу. Користувач зможе тут же його змінити і перерахувати всі значення трудомісткості для підсистем і системи в цілому (рис. 6).

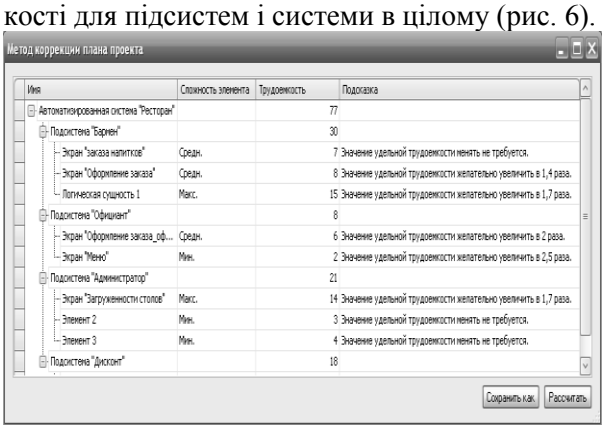


Рис. 6. Вікно результатів методу корекції плану проекту

Розроблений додаток дозволяє скоротити час, трудомісткість і ресурси процесу оцінювання програмних проектів.

**Висновок**

В роботі розглянуті методи підвищення якості планування програмного проекту: метод корекції плану проекту та формалізації загальносистемних характеристик.

Запропонований модифікований метод визначення диференційованої трудомісткості етапу розробки, що використовує поняття відносної трудомісткості, та дозволяє аналізувати різ-

ні типи проектів, які характеризуються великим розкидом тривалості проекту.

Практичною цінністю роботи є реалізація розглянутих методів, що дозволяє підвищити ефективність етапу планування.

**Список літератури**

1. Липаев В.В. Надежность программных средств [Текст] / В.В. Липаев. М. : СИНТЕГ, 2000. – 201 с.
2. Липаев В.В. Обеспечение качества программных средств. Методы и стандарты [Текст] / В.В. Липаев. М. : СИНТЕГ, 2001. – 380 с.
3. Скляр В.В. Оценка и обеспечение качества программных средств космических систем [Текст] / В.В. Скляр, В.С. Харченко, Б.М. Конорев и др. Харьков : НКАУ, Государственный центр регулирования качества, Нац. Аэрокосмический ун-т «ХАИ», 2006. – 224 с.
4. Харченко В.С. Методы моделирования и оценки качества и надежности программного обеспечения [Текст] / В.С. Харченко, В,В, Скляр, О.М. Тарасюк. Харьков : Нац. Аэрокосмический ун-т «ХАИ», 2004. – 159 с.
5. Макконелл С. Остаться в живых. Руководство для менеджера программных проектов [Текст] / С. Макконелл. Спб : Питер, 2006. – 240 с.
6. Маконелл С. Профессиональная разработка программного обеспечения [Текст] / С. Макконелл. М. : Символ, 2007. – 240 с.
7. Демарко Т. Человеческий фактор: успешные проекты и команды [Текст] / Т. Демарко, Т. Листер. Спб. : Символ-Плюс, 2005. – 134 с.
8. Ройс У. Управление проектами по созданию ПО [Текст] / У. Ройс. Лори, 1998. – 431 с.
9. Леффингуэлл Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход [Текст] / Д. Леффингуэлл, Д. Уидриг. М. : Издательский дом «Вильямс», 2002. - 448 с.
10. Ronald G. Ross Principles of the Business Rule Approach [Текст] / Ronald G. Ross. Addison-Wesley Professional, 2003. – 400 p.
11. Widrig D. Managing Software Requirements: A Use Case Approach [Текст] / D. Widrig, D. Leffingwell. 2003. – 544 p.
12. Bittner K. Use Case Modeling [Текст] / K. Bittner, I. Spence. 2002. – 368 p.
13. Вигерс К. Разработка требований к ПО [Текст] / Карл И. Вигерс. М. : «Русская редакция», 2004 – 576 с.
14. Канер С. Тестирование ПО [Текст] / С. Канер, Д. Фолк, Е. Нгуен. К. : ДиаСофт, 2000. – 544 с.
15. Канер С. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений [Текст] / С. Канер, Д. Фолк, Е. Нгуен. К. : ДиаСофт, 2001. – 544 с.
16. Майерс Г. Искусство тестирования программ [Текст] / Г. Майерс. М. : Финансы и статистика, 1982. – 176 с.
17. Харченко В.С. Верификация программного обеспечения [Текст] / В.С. Харченко, В.В. Скляр, А.А. Гордеев. Харьков: Нац. Аэрокосмический ун-т «ХАИ», 2007. – 132 с.
18. Калбертсон Р. Быстрое тестирование [Текст] / Р. Калбертсон, К. Браун, Г. Кобб. 2002. – 384 с.
19. Бек К. Экстремальное программирование. Разработка через тестирование [Текст] / К. Бек. Питер, 2003. – 224 с.
20. Гришин А.В. Метрики качества программного проекта [Текст] / А.В. Гришин, С.Н. Никонов, А.А. Ионов. Н-Н : ННГУ, 2004. – 24 с.
21. Меламед А.Я. Методы оценки трудоемкости разработки программного обеспечения корпоративных информационных систем [Текст]: автореферат дис. на соискание ученой степени кандидата технических наук / Меламед Александр Яковлевич; Московский государственный институт электроники и математики. – М. , 2006. – 21 с.
22. Архипенков С. Лекции по управлению программными проектами: [Электрон. ресурс]. – Режим доступа: http://citforum.univ.kiev.ua/SE/project/arkhipenkov_lectures/13.shtml/.
23. Калинина Л. Ю. Оценка качества программных продуктов [Текст] / Калинина Л. Ю. // Качество Инновации Образование. – 2006. - №4. – с. 14.
24. Григораш В.В. Управление качеством требований: [Электрон. ресурс]. – Режим доступа: http://www.slideshare.net/Vitaly.Grigorash/ss-2730920/.
25. ISO/IEC 9126-1:2001. Software engineering — Software product quality — Part 1-4: Quality model.
26. ДСТУ 3230-95. Управление качеством и обеспечение качества. Термины и определения.
27. Халина Д.С. Измерение качества требований к программному продукту [Текст] / Халина Д.С., Табунщик Г.В. // XVI Международный молодежный форум «Радиоэлектроника и молодежь в XXI веке». – 2010. – с.269.
28. Халина Д.С. Оценка качество программных проектов [Текст] / Халина Д.С., Табунщик Г.В. // Тиждень науки ЗНТУ. – 2010. – с. 154.
Халина Д.С. Реализация методов оценки качества программных проектов [Текст] / Халина Д.С., Табунщик Г.В. // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій. –2010 – с. 197.

**Відомості про авторів:**

**Табунщик Галина Володимирівна**, доцент Запорізького національного технічного університету, кафедра програмних засобів; кандидат технічних наук.
Наукові напрями – якість інформаційних систем, технології проектування інформаційних систем.
E-mail: tabunshchik@ieee.org

**Халіна Діана Сергіївна**, аспірант Запорізького національного технічного університету, кафедра програмних засобів. Наукові напрями – методи управління програмними проектами, управління якістю програмних проектів, метрики якості
E-mail: diaaana@i.ua

Стаття надійшла до редакції 20.10.2010 р.