УДК 004.4:657.474.51

В.О. Гороховатський, В.Ю. Дубницький, А.М. Кобилін, В.О. Лукін, О.В. Москаленко

Харківський інститут банківської справи Університету банківської справи Національного банку України (Київ), Харків

ВИЗНАЧЕННЯ ТРУДОМІСТКОСТІ ПРИ РОЗРОБЛЕННІ ПРОГРАМНИХ КОМПЛЕКСІВ

Розглянуто теоретичні підходи до нормування трудомісткості виготовлення програмного забезпечення та практику нормування у комп'ютерних фірмах; на базі проведенного аналізу існуючих у науковій літературі підходів та рекомендацій практиків розроблено пропозиції щодо нормування роботи програмістів.

Ключові слова: *трудомісткість, програмне забезпечення, нематеріальні активи, трудомісткість виробництва програмного забезпечення.*

Вступ

На сучасному етапі комп'ютеризації висока якість програмного забезпечення недосяжна без контролю процесу його розроблення [1]. Необхідність контролю створення програмного забезпечення, прогнозу його вартості, необхідність дотримання термінів і якості результатів зумовила перехід до промислової техноло-

гії створення програмного забезпечення, що одержала назву «інженерія програмного забезпечення». Проектування програмного забезпечення сучасних інформаційно-керуючих систем – трудомістка і тривала робота, що потребує участі висококваліфікованих фахівців. Всі етапи цієї роботи регламентовані вітчизняною галузевою та міжнародною нормативною базою [2]. Обсяг вихідного коду програмного забезпечення сучасних інформаційно-керуючих та складних технічних систем зростає у часі експоненційно та становить мільйони рядків програмного коду [2]. Подібне збільшення обсягів програмного забезпечення супроводжується нелінійним збільшенням складності, що, у свою чергу, впливає на собівартість програмного забезпечення. Після завершення процесу розробки програмного забезпечення настає етап поступової його модифікації, що дозволяє враховувати нові інформаційні вимоги. В силу специфіки методу створення програм, прийнятого в даний час, процес супроводження також є досить дорогим, тривалим і важкопередбачуваним. У більшості компаній по створенню програмного забезпечення інформаційних систем більше 50% штату зайнято супроводом програм, що входять до складу існуючих систем.

Необхідність врахування всіх вищезазначених складових підтверджує складність та багатофакторність визначення собівартості створеного власними силами програмного забезпечення.

Постановка завдання. Метою роботи є розробка пропозицій щодо визначення трудомісткості програмного забезпечення. Дослідження спрямоване на підвищення рівня нормування трудомісткості, планування та обліку витрат на виготовлення програмного забезпечення.

Аналіз літератури. Витрати на оплату праці є найбільш вагомим елементом загальних витрат, які включають у собівартість програмного забезпечення. Слід відзначити, що визначення витрат на оплату праці програмістів є досить складним завданням.

Оцінка трудомісткості програмного забезпечення є однією з ключових фінансових проблем, які виникають в процесі створення та розвитку новітніх інформаційних технологій. Програмне забезпечення як результат виробництва має ряд специфічних властивостей, на які звертають увагу дослідники, наприклад [3, 5]: серед них слід відзначити наступні особливості:

– створення програмного забезпечення – це інтелектуальний процес, тому він на сьогодні не підлягає точному вартісному та часовому оцінюванню;

– програмний продукт залежить від його призначення; створення програмного забезпечення для економічних та фінансових обчислень дещо відрізняється від програм для веб-дизайну, розробки сайтів, створення моделей об'єктів та ін.;

– важливе значення в оцінці трудомісткості мають окремі етапи розроблення програми, особливо етап відлагодження, який надто важко спрогнозувати у часі;

– затрати включають використання платного (ліцензійного) програмного забезпечення та стандартних програмних засобів: зв'язок, Інтернет, аутсорсинг (оплата послуг сторонніх організацій);

– необхідність обов'язкового врахування експертних оцінок затрат за окремими статтями розробки;

– конкурентоспроможність програмного продукту визначається не «абсолютною цінністю» для споживача, а порівняльною корисністю його характеристик (технічних, економічних, дизайнерських, особливості інтерфейсу та ін. параметрів) із відповідними характеристиками програмного забезпечення конкурента. Затрати зростають зі зростанням показника конкурентоспроможності;

– при створенні програмного забезпечення з'являються додаткові ризики: зв'язок роботи розробленого програмного забезпечення з іншим програмним забезпеченням, яке вже існує або вимагає додаткового розроблення для його використання;

– дуже часто проект створюваного програмного забезпечення є унікальним в рамках одного підприємства, навіть якщо використовується готова платформа, тому важко у повній мірі використовувати дані, отримані при виконанні схожих проектів [3 – 10].

Оцінка трудомісткості, на погляд розробників, відноситься до імовірнісних тверджень. Це значить, що для неї існує деякий розподіл ймовірності, який може бути досить широким, якщо невизначеність усіх факторів дуже висока, або достатньо вузьким, якщо невизначеність незначна. Використання власного досвіду або досвіду колег, отриманого в схожих проектах, – це найбільш прагматичний підхід, що дає можливість отримати достатньо реалістичні оцінки трудомісткості та строків реалізації програмного продукту швидко і без значних затрат.

Дослідники, виходячи з процесу проектування, поділяють існуючі моделі оцінювання трудомісткості програмного забезпечення на дві групи [5]: алгоритмічні, що засновані на підрахунку кількісних характеристик програми у вигляді числа операторів або функціональних точок, та неалгоритмічні, що використовують визначені схеми або принципи. Розглянемо спочатку неалгоритмічні моделі. До найбільш відомих з них відносяться експертні оцінки.

Експертні оцінки в першу чергу застосовуються для проектів, що вирішують інноваційні завдання або засновані на новітніх технологіях та процесах. Зважаючи на інноваційність значної частини створюваного програмного забезпечення, такі оцінки набули значного поширення у практиці розроблення програм [6]. Часто експертами виступають безпосередньо замовники та розробники, що мають певний досвід. На основі оцінок окремих експертів у відповідності до існуючих методик формується інтегрована консенсусна оцінка.

До загальних недоліків методів експертного оцінювання відносять моменти, пов'язані з використанням складного і часом непрогнозованого «людського» фактору. Важко оцінити ступінь повноти і достовірності інформації, поданої експертами. Немає повної впевненості, що експерти виявили дійсно всі основні проблеми і правильно визначили взаємозв'язки між ними. Відсутність у явному виді аналітичного обґрунтування виявлених проблем, складність у перевірці компетенції тієї або іншої особи через неформалізованість критеріїв об'єктивності представлених даних теж є актуальними.

Метод оцінки по аналогії [8] є загальноприйнятим для будь-якого оцінювання і вважається різновидом експертної оцінки. Він використовує у якості оцінки емпіричні дані про вже завершені проекти схожого типу.

Алгоритмічні моделі базуються на математичних моделях і постійно удосконалюються з метою підвищення їх точності оцінювання.

Найчастіше реалізованими і добре документованими моделями є модель Путнем (ступенева, аналітична) і модель COCOMO (ступенева, емпірична) [4].

Модель Путнем (SLIM) [7] є найбільш поширеною моделлю аналітичної групи. Вона створена для проектів об'ємом більше 70 000 рядків коду. Модель ґрунтується на твердженні, що витрати на розробку програмного забезпечення розподіляються згідно кривим Нордена-Рейлі, які є графіками функції, що представляє розподіл робочої сили у часі [11]. При цьому необхідно звернути увагу на те, що спочатку дослідження Нордена базувалися не на теоретичній основі, а на спостереженнях за проектами, в основному не пов'язаними з програмним забезпеченням (машинобудування, будівництво). Тому немає наукового підтвердження тому факту, що програмні проекти вимагають такого ж розподілу робочої сили, навпаки, часто кількість людино-годин, необхідних за проектом, може різко змінитися, зробивши оцінку непридатною до використання [12].

У 1991 році була представлена альтернативна реалізація моделі, виконана на замовлення Quantitative Software Management (QSM) Inc. і застосована в комплексі SLIM Estimate для оцінки вартості програмного забезпечення [13]. Обмеженням цієї моделі є те, що вона може бути використана, якщо передбачувані витрати на створення програмного забезпечення більше 20 людино-місяців.

Найпопулярнішою серед алгоритмічних моделей є сімейство моделей COCOMO (Constructive Cost Model), створене у 1981 р. [7]. Модель використовує просту формулу регресії з параметрами, визначеними з даних, зібраних по ряду проектів. COCOMO являє собою три моделі у відповідності до фаз життєвого циклу програмного забезпечення: базова (Basic) застосовується на етапі напрацювання специфікацій, розширена (Intermediate) – після визначення конкретних вимог до програмного забезпечення, поглиблена (Advanced) застосовується після закінчення проектування програмного забезпечення. При цьому витрати праці на проект визначають у людино-місяцях і залежать від розміру коду програми та від її складності, що визначається різновидом програми:

– відносно проста, однорідна команда розробників;

– проект середньої складності, вимоги можуть змінюватися у процесі розроблення;

– складний (вбудований) проект, що повинен бути реалізованим у жорстких рамках заданих вимог.

Відомі модифікації COCOMO у вигляді сімейства моделей COCOMO II [7], що створені у 1999 р. Основними відмінностями COCOMO II від COCOMO є використання для оцінювання складності вхідних даних у вигляді функціональних точок, оцінювання елементів повторного використання та інтеграції програмних продуктів, об'єктно-орієнтовані підходи до оцінки компонентів програмного забезпечення та ін. Функціональні точки – це одиниці вимірювання проекту, вони залишаються незмінними незалежно від програмістів та мов програмування.

Модель Путнем надзвичайно чутлива до значення технологічних факторів, тому точне визначення їх значення є дуже важливим для правильної оцінки на основі SLIM. Перевагою моделі Путнем перед COCOMO 1.1 або COCOMO 2.0, є невелика кількість параметрів, необхідних для оцінки [12]. Ефективне застосування алгоритмічних моделей оцінки вартості програмного забезпечення та заснованих на них засобів оцінки воліють їх спільне використання з неалгоритмічними методами оцінки. Завдяки широким можливостям експорту даних і візуалізації, використання автоматизованих засобів оцінки вартості програмного забезпечення дає можливість формувати власні бази характеристик реалізованих проектів, а також створювати звіти, що ілюструють процес розробки проекту, що значно знижує трудовитрати, пов'язані з підготовкою звітності [12].

У цілому вважається, що для невеликих проектів застосування високих рівнів оцінювання COCOMO неефективне, а базовий рівень дає недостатню точність. Для комерційних застосувань метод COCOMO дає, як правило, завищені оцінки, тому його застосовують більше в проектах, що відносяться до програмного забезпечення інженерних розрахунків.

У різний час та в різних країнах централізовано розроблялися норми часу та нормативи трудомісткості для нормування роботи програмістів. Так, у Республіці Білорусь для розробки програмного забезпечення «в якості одиниці вимірювання обсягу програмного забезпечення використовувався рядок вихідно-

го коду (LOC)» [14, п.7], а для визначення трудомісткості визначення супроводження програмних продуктів пропонувалося три варіанти визначення трудомісткості супроводження програмних продуктів.

Перший варіант містить порядок розробки організаціями нормативів і норм часу для розрахунку трудомісткості виконання робіт з супроводження програмних продуктів і проекти збільшених нормативів і норм часу на виконання робіт з супроводження програмних продуктів, розрахованих на основі індивідуальних і групових експертних оцінок спеціалістів низки провідних організацій, які здійснюють розробку та супроводження програмних продуктів.

Другий варіант ґрунтується на визначенні базової трудомісткості супроводження програмних продуктів з застосуванням корегуючих (поправочних) коефіцієнтів, які відображають вплив на її величину інших характеристик програмних продуктів, які супроводжуються.

За третім варіантом трудомісткість супроводження програмних продуктів визначається за його питомою вагою в загальній трудомісткості на розробку програмних продуктів [15, п.4].

У Росії згідно з ОСТ 4.071.030 нормативи трудомісткості встановлюються: на розробку програм за групами складності програм і в залежності від мови програмування; на впровадження проектних рішень АСУП в залежності від: ступеню новизни АСУП, які створюються; складності задач; періодичності рішення задач [16, п.1.12, 1.13].

Існують й інші підходи щодо встановлення нормативів трудомісткості, але всі вони ґрунтуються на базових нормах, які встановлюються на основі дослідно-статистичних або експертних норм, які коректуються на ті або інші показники складності програм. Проте використовувати централізовано розроблені норми у практичній діяльності неможливо. Причина в тому, що такі норми не враховують два фактори: поточний стан культури програмування та поточну кваліфікацію програмістів.

На складність нормування роботи програмістів вказують як практики, так і науковці. В роботі [17] зазначено, що: «Створення відповідних норм і нормативів витрат праці – дуже складний процес насамперед через те, що програмування – творча праця, а також тому, що в програмуванні однієї задачі можуть брати участь фахівці різних професій і категорій: наукові працівники, математики, інженери-технологи, економісти, техніки, оператори. Крім того, продуктивність праці залежить від багатьох факторів, урахувати які важко, а саме:

- логічна складність розроблюваної задачі;
- кваліфікація програміста;
- мова програмування;
- математичне забезпечення ЕОМ;
- система команд та адресність ЕОМ тощо».

Програмісти-практики рідко використовують норми часу для нормування робіт зі складання програм. Як правило, обґрунтування наступне[18]: «…довжина програми має дуже слабке відношення до обсягу роботи… У нас в Красноярську проводять конкурси програмістів. Збираються 20-40 кращих програмістів міста і розв'язують одну задачу. Задача одна, час однаковий (один людино-день, тому що конкурс проходить за один день), а от довжина програм істотно різна. От цифри: розкид довжин від 17 до 129 операторів, довірчий інтервал довжин від 30 до 90 операторів, тобто у границях довірчого інтервалу довжина «плаває» в три рази».

Таку ж точку зору мають і інші практики [19]: «Мій досвід і досвід інших спеціалістів говорить про те, що формальних процедур нормування праці програмістів практично не існує. Розробка програмного забезпечення, на мою думку, найскладніший вид людської діяльності. Формулу витрат праці вивести неможливо, тут придатні лише експертні оцінки, спочатку грубі, потім більш покращені. Поступово ми приходимо до розуміння того, скільки праці та часу спеціалістів витратимо».

В роботі [20] зазначено, що: «формального нормування праці програмістів немає. Ми не рахуємо строки коду, функції, класи і т.ін. При підготовці плану виконання проекту відповідальний виконавець проекту з начальником відділу та іншими фахівцями дають експертну оцінку (інакше кажучи, визначають навмання) тривалість розробки підзавдань проекту теми програмістами, які включені в групу проекту. Тобто нормування праці програмістів на практиці проводиться в основному на базі експертних оцінок. Але в даному випадку це експерти, які оцінюють трудомісткість роботи виходячи з конкретної ситуації.

Отримані результати

Розрахунок трудомісткості робіт з підготовки програмного забезпечення дійсно є досить складною проблемою. Справа в тому, що діяльність програміста є суто творчою роботою. Як відомо, в трудовому процесі всі трудові операції поділяються на три види: творчі, логічні (переробні) та технічні.

Технічні операції (наприклад, розмноження документів, арифметичні підрахунки, розноска даних в картотеки, ведення діловодства) характерні найбільшою частотою повторення, прості по виконанню, не вимагають високої кваліфікації працівника. Вони не вимагають від працівників спеціальній підготовки, тому легко нормуються.

Логічні (переробні) операції виконуються за розробленим алгоритмом, тобто у певній наперед заданій послідовності. Вони регламентуються відповідними інструктивними або нормативними документами і тому складніше ніж технічні, але простіше творчих. Во-

ни вже вимагають від працівників спеціальної підготовки, проте їх також досить легко нормувати.

Найбільш складними із них є творчі операції. Вони вимагають достатньо високої кваліфікації працівників, тому що включають такі розумові дії, як порівняння, аналіз, абстрагування, конкретизація, висновки, розрахунки, прогнози, прийняття рішень тощо. Творчі операції носять досить складний характер і тому важко піддаються зовнішньому контролю та нормуванню; враховувати їх можна лише непрямими методами, оцінюючи складність праці, а також кінцеві результати цієї діяльності. Операції з підготовки програмного забезпечення мають найвищий рівень складності, тому повністю відноситься до творчих операцій. Саме тому неможна точно пронормувати роботу таких спеціалістів, хоча для нормування пропонуються багато різних методів щільно до методів, які ґрунтуються на теорії нечіткої логіки [2]. Проте, по перше, такі методи також не дають змогу точно пронормувати роботу спеціалістів, а, по друге, – вони досить складні у використанні; не всі спеціалісти, які займаються нормуванням праці, знають, що це таке та як їх можна використовувати. Краще використовувати метод експертних оцінок, який має багато варіантів. Для використання пропонується наступна модель нормування праці: планові норми часу встановлювати на базі інтервального методу, тобто спочатку встановлюється імовірний інтервал трудоемкости (із залученням кваліфікованих експертів, керівництва та виконавців), а потім проводиться коректування його величини з допомогою поправочних коєффіциентов.

Трудомісткість робіт з підготовки програмного забезпечення розрізняється у залежності від того, є або ні у нього аналоги.

Трудомісткість робіт з підготовки програмного забезпечення, яке не має аналогів та не пов'язано з раніше отриманими результатами, або є логічним продовженням раніше виконаних робіт (Т) може бути обчислена за формулою, яку пропонує використовувати Методика визначення вартості наукових робіт [21, ф. (2.5)]. Це дозволяє визначити нижню границю оцінки трудомісткості роботи з розробленя програмного забезпечення таким чином:

$$T = N_н(1 + K_н + K_{скл} + K_і + K_{терм} + K_{кор}); \qquad (1)$$

де $N_н$ – базовий норматив трудомісткості проведення робіт, люд./дн. Він визначається експертним методом. Замість базового нормативу трудомісткості проведення роботи $N_н$ можна використовувати норматив трудомісткості аналогічної роботи (якщо він відомий).

Для урахування індивідуальних особливостей роботи можуть бути використані поправочні коефіцієнти:

$K_н$ – коефіцієнт новизни, який характеризує ступінь наближення нової інформації відносно вже відомих результатів та ступінь наукової розробленості проблеми, наявність інформаційного наробітку і практичного досвіду виконання подібних робіт;

$K_{скл}$ – коефіцієнт складності. Він враховує доступність інформації про об'єкт дослідження; про стан науково-технічного наробітку; досконаліть інформаційного забезпечення та обслуговування; необхідність залучення спеціалістів нового профілю, яких немає в організації; умови, які потрібні для виконання теоретичного дослідження;

$K_і$ – коефіцієнт інформаційної місткості, який враховує багатовекторність теоретичної роботи, кількість досліджуваних проблем, кількість джерел інформації, які використовують при проведенні теоретичної роботи;

$K_{терм}$ – коефіцієнт терміновості, який враховує терміновість виконання теоретичної роботи;

$K_{кор}$ – коефіцієнт значимості результатів теоретичної роботи, який враховує корисність та прогресивність теоретичної роботи. Цей коефіцієнт враховує зростання трудомісткості теоретичної роботи, у разі необхідності оформлення патентного захисту можливих об'єктів інтелектуальної власності; розроблення проекту технічного завдання на наступну науково-дослідну роботу, у разі необхідності здійснення подальших досліджень.

Всі коефіцієнти визначаються експертним методом. Особливості проведення робіт з виготовлення програмного забезпечення у кожному випадку обумовлюють свій набір значень коефіцієнтів. Якщо за якоюсь причиною немає можливості провести експерте оцінювання трудомісткості робіт (відсутність експертів належної кваліфікації, принципово нове завдання, яке не має аналогів тощо) всі коефіцієнти ($K_н$, $K_{скл}$, $K_і$, $K_{терм}$, $K_{кор}$) треба прийняти рівними 0.

Можливі значення цих коефіцієнтів, визначених експертним шляхом,наведено у табл. 1.

У процесі проведення розрахунків для визначення трудомісткості роботи можливі такі варіанти:

1. Для виконання розрахунків використовують середні значення коефіцієнтів.

2. Для виконання розрахунків використовують значення коефіцієнтів, узгоджені з замовником.

3. Для виконання попередніх розрахунків використовують інтервальні значення всіх, або частини коефіцієнтів в залежності від поінформованості замовника та виконавця про умови виконання роботи. В останньому випадку необхідно використовувати технології обчислень, пов’язаних з інтервальними обрахунками.

Після розрахунку трудомісткості робіт з виготовлення програмного забезпечення загальні витрати на оплату праці ($S_{опл.пр.}$) можна визначити за формулою (2), але планування витрат не закінчується розрахунком загальної трудомісткості та витрат на оплату праці.

Таблиця 1

Інтервали можливих значень поправочних коефіцієнтів

№	Назва	Позначення	Межі нтервалу		
			Найменше значення	Середнє значення	Найбільше значення
1	Коефіцієнт новизни, який характеризує ступінь близькості нової інформації до вже відомих результатів та ступінь наукової розробленості проблеми, наявність інформаційного наробітку і практичного досвіду виконання подібних робіт: частина роботи спирається на раніше отримані результати; робота принципово нова	$К_н$	 0,1 0,15	 0,15 0,20	 0,20 0,25
2	Коефіцієнт складності, враховує необхідність залучення спеціалістів нового профілю, яких немає в організації; умови, які потрібні для виконання дослідження	$К_{скл}$	0	0,3	0,5
3	Коефіцієнт інформаційної місткості, який враховує багатовекторність теоретичної роботи, кількість досліджуваних проблем, кількість джерел інформації, які використовують при проведенні роботи	$К_і$	0	0,15	0,3
4	Коефіцієнт терміновості, який враховує терміновість виконання роботи: планова планова, визначена як невідкладна; позапланова	$К_{терм}$	 0 0,1 0,2	 0 0,15 0,4	 0 0,25 0,5
5	Коефіцієнт корисності та значимості результатів теоретичної роботи, який враховує корисність та прогресивність роботи. Цей коефіцієнт враховує зростання трудомісткості роботи в разі необхідності оформлення патентного захисту можливих об'єктів інтелектуальної власності; розроблення проекту технічного завдання на наступну науково-дослідну роботу, у разі необхідності подальших досліджень:	$К_{кор}$	0	0,15	0,25

Керівник підрозділу повинен орієнтуватися на календарний план розроблення програмного забезпечення, який є складовою технічного завдання, тоді:

$$S_{опл.пр.} = \frac{T}{D} \times З_{ср}, \qquad (2)$$

де Т – загальна трудомісткість роботи, люд./дн. (або люд./год); D – середня кількість робочих днів (годин) у місяцях, протягом яких буде виконуватися робота, дн. (або год.); $З_{ср.}$ – середньомісячна заробітна плата (основна та додаткова) колективу робітників, які безпосередньо задіяні для виконання робіт, грн.

В залежності від загальній трудомісткості і терміну виконання робіт визначається планова кількість робітників, коефіцієнти їх участі в проекті та термін їх участі в проекті. Для цього можна використовувати формулу (3):

$$П = T / t, \qquad (3)$$

де П – планова чисельність робітників;

t – термін виконання роботи.

Висновки

1. Аналіз вітчизняної та зарубіжної літератури, а також діючих нормативних документів задля визначення трудомісткості та оцінювання собівартості програмного забезпечення показує, що традиційні підходи, що засновані на підрахунку кількісних характеристик програми та застосуванні визначених розробником та замовником схем або принципів, не є універсальними і не враховують всіх чинників та особливостей досліджуваної проблеми.

2. Практика показує, що процес розроблення програмного забезпечення, як один з найбільш інтелектуальних видів сучасної людської діяльності, досить складно піддається нормуванню з використанням традиційних формальних моделей, що при оцінюванні працезатрат часто вимагає додаткового застосування експертних чи інтервальних підходів.

3. Більшої універсальності та об'єктивності оцінювання можна досягти з використанням комбінованих підходів, що поєднують традиційні методи проведення розрахунків трудомісткості з використанням експертно оцінюваних коефіцієнтів та технології обчислень, пов'язаних з інтервальними обрахунками.

4. Перспективою досліджень є розроблення методики оцінювання в двох можливих варіантах – інтервальному та точковому. Інтервальний варіант може бути ефективно використано на стадії попере-

дніх розрахунків (на етапі планування границь для значення можливої собівартості програмного забезпечення), точковий – на завершальній стадії, коли розраховується фактична собівартість програмного забезпечення з урахуванням всіх витрат.

Робота виконана в рамках виконання науково-дослідної роботи «Визначення собівартості власного розроблення програмного забезпечення (№ держреєстрації 0113U002194)».

Список літератури

1. Мандрикова Л.В. Методы оценки стоимости и затрат на создание программного продукта, основанные на нечеткой логике / Л.В. Мандрикова, Ю.С. Манжос, П.А. Лучшев // Радіоелектронні і комп'ютерні системи. – 2008. – № 2 (29). – С. 115-118.

2. Інформаційні технології. Процеси життевого циклу програмного забезпечення: ДСТУ 3918-1999 (ISO/IEC 12207-1995). – К.: Дердстандарт України, 1995. – 57 с.

3. Сидоров Н.А. Методы и средства оценки стоимости программного обеспечения / Н.А.Сидоров, Д.В. Баценко, Ю.Н. Василенко, Ю.В. Щебетин, Л.Н. Иванова // Проблеми системного підходу в економіці. – НАУ. – 2004. – № 7. – С. 113-118.

4. Boehm B.W. The COCOMO 2.0 Software Cost Estimation Model / B.W. Boehm. – American Programmer, 2000. – 586 p.

5. Boehm B.W. Software engineering economics / B.W. Boehm. – Prentice–Hall, 1981. – 320 p.

6. Coates J. Technological Forecasting and Social Change / J. Coates. – Elsevier Science Inc, 1999. – 235 p.

7. Johnson Kim. Software cost estimation – Metrics and models / Johnson Kim. – University of Calgary, 2001. – 115 p.

8. Shepperd M. Estimating software project effort using analogy / M. Shepperd, C. Schofield // IEEE Trans Software Eng. – 1997. – P. 736-743.

9. Software engineering: IFPUG 4.1 Unadjusted functional size measurement method: Counting practices manual. – ISO/IEC. – 2003. – 430 p.

10. Yang Y. Reducing Local Calibration Bias in COCOMO II 2004 Calibration / Y. Yang, B. Clark // 19th International Forum on COCOMO and Software Cost Modeling, October 26-29, 2004. [Електронний ресурс]. − Режим доступу: http://sunset.usc.edu/cse/pub/event/2004/COCOMO/files/WedAM/Wed_AM_05.ppt.

11. David L. Norden-Raleigh Analysis: A Useful Tool for EVM in Development projects / L. David. – The Measurable News. – 2002. – 24 p.

12. Модели, методы и средства оценки стоимости програмного обеспечения / Д.В. Баценко, Н.А. Сидоров, Ю.Н. Василенко, Ю.В. Щебетин. [Електронний ресурс]. – Режим доступу до ресурсу: http://masters.donntu.edu.ua/2010/fknt/zhukova/library/article01.htm.

13. Основы программной инженерии [Електронний ресурс]. − Режим доступу до ресурсу: http://swebok.sorlik.ru/index.html.

14. Постановление Министерства труда и социальной защиты Республики Беларусь от 27 июня 2007 г. № 91. «Об утверждении укрупненных норм затрат труда на разработку программного обеспечения» [Електронний ресурс]. − Режим доступу до ресурсу: http://www.busel.org/texts/cat3ka/id5swucdd.htm.

15. Постановление Министерства труда и социальной защиты Республики Беларусь от 23.09.2011 N 92 «Об утверждении Методических рекомендаций по определению трудоемкости сопровождения программных продуктов». [Електронний ресурс]. − Режим доступу до ресурсу: http://pravo.levonevsky.org/bazaby11/republic00/text237.htm.

16. Отраслевой стандарт автоматизированная система управления предприятием. Создание системы. Нормативы трудоёмкости. ОСТ 4.071.030. [Електронний ресурс]. − Режим доступу до ресурсу: http://www.internet-law.ru/law/gosts/normatives.htm.

17. Лазарєва С.Ф. Економіка та організація інформаційного бізнесу. [Електронний ресурс]. − Режим доступу до ресурсу: http://studentam.kiev.ua/content/view/776/81/.

18. Бабий А. Об измерении труда программистов. [Електронний ресурс]. − Режим доступу до ресурсу: http://www.alex.krsk.ru/198_/1986/1986_07.htm.

19. Как работают питерские программисты. Интервью с генеральным директором компании АстроСофт П.В.Васильевым. [Електронний ресурс]. − Режим доступу до ресурсу: http://www.ci.ru/inform20_05/p_14.htm.

20. Как работают питерские программисты. Интервью с руководителями ведущих организаций С-Петербурга, производящих программное обеспечение на заказ [Електронний ресурс]. − Режим доступу до ресурсу: http://www.ci.ru/inform17_05/p_08.htm.

21. Про затвердження Методики визначення вартості наукових робіт. Наказ Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків чорнобильської катастрофи від 22 липня 2008 року N 545.

Надійшла до редколегії 27.02.2014

Рецензент: д-р екон. наук, проф. Т.В. Давидюк, Харківський інститут банківської справи Університету банківської справи НБУ, Харків.

ОПРЕДЕЛЕНИЕ ТРУДОЕМКОСТИ ПРИ РАЗРАБОТКЕ ПРОГРАММНЫХ КОМПЛЕКСОВ

В.А. Гороховатский, В.Ю. Дубницкий, А.М. Кобылин, В.А. Лукин, Е.В. Москаленко

Рассмотрены теоретические подходы к нормированию трудоемкости создания программного обеспечения и практика нормирования в компьютерных фирмах; на базе проведенного анализу существующих в научной литературе подходов и рекомендаций практиков разработаны предложения по нормированию работы программистов.

Ключевые слова: *трудоемкость, программное обеспечение, нематериальные активы, трудоемкость производства программного обеспечения.*

LABOR INTENSIVENESS DETERMINATION IN DEVELOPMENT OF SOFTWARE PACKAGES

V.A. Gorokhovatskiy, V.Yu. Dubnitskiy, A.M. Kobylin, V.A. Lukin, E.V. Moskalenko

Theoretical approaches discussed to labor quota setting in development of software packages as well as quota setting practice in IT companies; on the basis of analysis of approaches existing in literature and of practitioners' recommendations some proposals are presented on programmer labor quota setting.

Keywords*: labor intensiveness, software, intangible assets, labor intensiveness of software development.*