УДК 004.9:412

Ю.В. Жмаєва, Л.Е. Чала
ХНУРЕ, м. Харків, Україна, larysa.chala@nure.ua

ГІБРИДНИЙ МЕТОД ОЦІНЮВАННЯ СКЛАДНОСТІ ІТ ПРОЕКТІВ

Досліджено сучасні методи оцінювання складності проектування програмних продуктів. Розглянуто процедури визначення меж, запитів та транзакцій взаємодії додатків. Запропонований метод дозволяє корегувати результати, отримані за допомогою методу функціональних точок, з використанням алгоритмів корекції оцінок складності проектування. Визначено перспективи використання методу.

СКЛАДНІСТЬ ПРОГРАМНИХ ДОДАТКІВ, МЕТОД ФУНКЦІОНАЛЬНИХ ТОЧОК, КОРЕКЦІЯ ОЦІНОК

Ю.В. Жмаева, Л.Э. Чалая. Гибридный метод оценивания сложности IT проектов. Исследованы современные методы оценки сложности проектирования программных продуктов. Рассмотрены процедуры определения границ, запросов и транзакций взаимодействия приложений. Предложенный метод позволяет корректировать результаты, полученные с помощью метода функциональных точек, с использованием алгоритмов коррекции оценок сложности проектирования. Определены перспективы использования метода.

СЛОЖНОСТЬ ПРОГРАММНЫХ ПРИЛОЖЕНИЙ, МЕТОД ФУНКЦИОНАЛЬНЫХ ТОЧЕК, КОРРЕКЦИЯ ОЦЕНОК

Y.V. Zhmaeva, L.E. Chala. A hybrid method for estimating the complexity of IT projects. The modern methods of evaluating the complexity of designing software products are investigated. Considered procedures for determining the boundaries, queries and transactions of interaction of applications. The proposed method allows you to adjust the results obtained using the method of functional points, using algorithms for correcting the estimates of the complexity of the design. The possibilities of using the method are determined.

COMPLEXITY OF SOFTWARE APPLICATIONS, METHOD OF FUNCTIONAL POINTS, CORRECTION OF EVALUATING

Вступ

Визначення складності розробки програмного забезпечення є одним з найбільш критичних аспектів в управлінні ІТ проектами [1]. Точне визначення розміру програмного забезпечення допомагає менеджерам проектів ефективно планувати і виконувати поставлені завдання. Дотримання правил, що диктуються існуючими методиками оцінок складності ІТ проектів, є надзвичайно важливим. Розробникам програмного забезпечення доводиться виробляти значні обсяги вихідного продукту в умовах постійного розширення вимог користувача, внаслідок чого метод вимірювання розміру програми повинен бути максимально уніфікований. Поширеним методом вирішення цієї проблеми є функціональний точковий аналіз, який дозволяє здійснювати функціональну декомпозицію процесу проектування програмного забезпечення [2]. Основою даного підходу є класичний метод оцінювання програмного продукту з використанням ієрархічної структури робіт.

Застосування базового методу функціональних точок (МФТ) дозволяє вимірювати масштабність програмного забезпечення шляхом кількісної оцінки його функціональності, що надається користувачеві насамперед на момент реалізації логічного дизайну. Однак, безпосереднє використання МФТ має ряд недоліків, які не тільки вимагають додаткових витрат на проектування, але і можуть негативно позначитися на взаємодію проектувальників та замовників.

У даній роботі досліджуються сучасні методи оцінювання процесу проектування програмних продуктів, а також пропонується модифікований гібридний підхід до оцінки складності ІТ проектів на основі удосконалення методу функціональних точок за рахунок застосування механізмів ризик-менеджменту та прогнозування зусиль, пов'язаних з розробкою програмних систем.

1. Базовий метод функціональних точок

Згідно з базовим методом функціональних точок, трудомісткість програмного продукту розраховується з урахуванням функціональності системи, що розробляється, яка в свою чергу визначається на основі логічних груп взаємопов'язаних даних, підтримуваних і використовуваних додатком, а також простих процесів, пов'язаних з виведенням і введенням інформації. Безсумнівною перевагою методу є те, що можливість його застосування не залежить від технологічної платформи, на якій буде розроблятися продукт.

Для використання МФТ слід виконати послідовність наступних кроків: визначення типу виконуваної оцінки; оцінку області та меж розробки програмного продукту; підрахунок функціональних точок, пов'язаних з даними; підрахунок функціональних точок, пов'язаних з транзакціями; визначення сумарної кількості невирівняних функціональних точок (UFP); визначення значення фактору вирівнювання (FAV); розрахунок

кількості вирівняних функціональних точок (AFP – Adjusted Functional Point). Метод функціональних точок зазвичай передбачає оцінки наступних типів:

– «проект розробки» – проект, який є новим для розробника;

– «проект розвитку» – проект, розробка якого тривала певний час (оцінюється проект доопрацювання в функціональних точках: видалення, додавання, зміна та доопрацювання функціоналу);

– «продукт» – проект, якому надається оцінка незалежною стороною (оцінюються обсяги вже існуючого і розробленого продуктів).

Для оцінювання меж розробки програмного продукту мають бути визначені:

– внутрішні логічні файли (ILFs — Internal Logical File), що виділяються користувачем як логічно пов'язані групи даних або блоки керуючої інформації, які підтримуються всередині продукту;

– файли зовнішнього інтерфейсу (EIFs), що виділяються користувачем як логічно пов'язані групи даних або блоки керуючої інформації, на які посилається продукт, але які підтримуються поза продукту.

Прикладами логічних даних (інформаційних об'єктів) можуть бути клієнт, рахунок, тарифний план, послуга. Для підрахунку функціональних точок, пов'язаних з даними, спочатку визначається складність даних за такими показниками:

– DET (Data Element Type), тобто повторюване унікальне поле даних, наприклад, ім'я клієнта — 1 DET; адреса клієнта (індекс, країна, область, район, місто, вулиця, будинок, корпус, квартира) — 9 DET's;

– RET (Record Element Type) – логічна група даних, наприклад, адреса, паспорт, телефонний номер.

Оцінка кількості невирівняних функціональних точок залежить від складності даних, яка визначається згідно з матрицею складності (таблиця 1).

Таблиця 1

Матриця складності даних

Тип	1-19 DET	20-50 DET	51+ DET
1 RET	Low	Low	Average
2-5 RET	Low	Average	High
6+ RET	Average	High	High

Оцінка даних в невирівняних функціональних точках розраховується для внутрішніх логічних файлів (ILFs) і для зовнішніх інтерфейсних файлів (EIFs) в залежності від їх складності.

Крім того здійснюється підрахунок функціональних точок, пов'язаних з транзакціями. Транзакція – це елементарний неподільний замкнутий процес, який пре і переводить продукт з одного консистентного стану в інший. У МФТ розрізняються кілька типів транзакцій, представлених в таблиці 2.

Таблиця 2

Функції та типи транзакцій

Функція	Тип транзакції		
	EI	EO	EQ
Зміна поведінки системи	Основна	Додаткова	–
Підтримка одного або кількох файлів інтерфейсу ILF	Основна	Додаткова	–
Надання інформації клієнту/ користувачу	Додаткова	Основна	Основна

Оскільки більшість комп'ютерних систем взаємодіють з іншими системами, то мають бути визначені межі поділу між розроблюваним проектом або додатком і зовнішніми додатками або доменом користувача. Після встановлення таких меж, компоненти розроблюваного програмного забезпечення можуть бути класифіковані, ранжовані і підраховані.

2. Визначення меж, запитів та транзакцій взаємодії додатків

Межі можуть бути встановлені на ранньому етапі життєвого циклу програмного забезпечення. Якщо ж команда працює з додатком повторно, наприклад, додавання нового функціоналу до релізу, то межа проекту повинна бути аналогічною, навіть ідентичною попередній. Якщо додаток новий, межі інших додатків, які будуть взаємодіяти з розроблюваної системою, повинні бути переглянуті та перераховані [3].

Існує ряд технічних особливостей, які дозволяють точно визначити межі додатків. Межа для вебдодатку, яка наведена на рисунку 1, визначена аналогічним чином як для традиційних додатків. Для традиційних додатків кордон не малюється тільки навколо призначеного для користувача інтерфейсу або групи екранів, а навколо всього програми. Часто такі додатки є просто розширеннями для існуючих десктопних додатків.

Межі для клієнт-серверного додатку, приклад якого схематично наведено на рисунку 2, повинні бути орієнтовані як на клієнта, так і на сервер. Це пов'язано з тим, що ні клієнт, ні сервер не підтримують інтерфейс, тобто жоден компонент не представляє собою сам додаток.

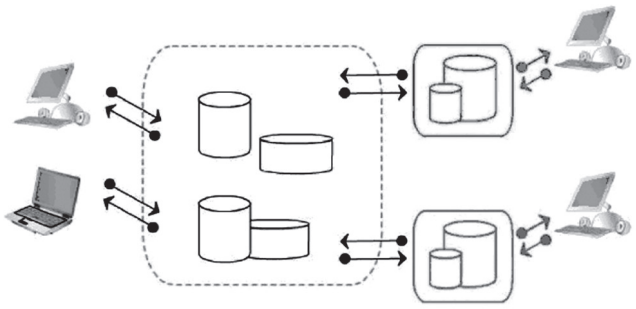

Рис. 1. Визначення меж веб-додатку

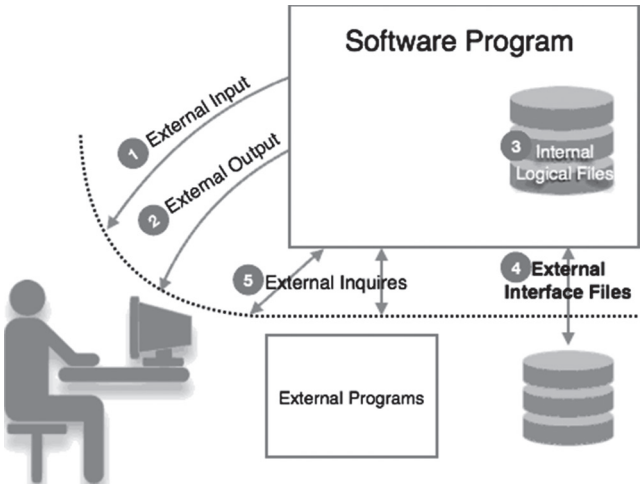

Рис. 2. Визначення меж клієнт-серверного додатку

Зовнішні вхідні транзакції EI (External Inputs) – це елементарний процес, в якому дані перетинають межу ззовні всередину. Ці дані можуть надходити з екрана введення даних або іншої програми. Вони можуть використовуватися для підтримки одного або декількох внутрішніх логічних файлів. Дані транзакції EI можуть бути пов'язані з керуючою або діловою інформацією. Для даних, пов'язаних з керуючою інформацією, не потрібно оновлювати внутрішній логічний файл. На рис. 3 наведено приклад простої транзакції EI, яка оновлює два файли ILF [4].

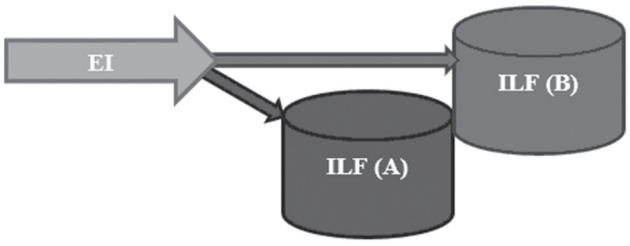

Рис. 3. Оновлення внутрішніх логічних файлів за допомогою зовнішніх вхідних транзакцій

Якщо зовнішні вхідні транзакції додають, змінюють і видаляють інформацію у внутрішньому логічному файлі, то така взаємодія представляється за допомогою трьох зовнішніх входів. Зовнішнім входам (особливо змінам і видаленням) може передувати зовнішній запит.

Як і всі компоненти, зовнішні вхідні транзакції можуть бути оцінені. Рейтинг вхідних транзакцій заснований на підрахунку кількості неповторюваних унікальних полів даних (DET) і пов'язаних з ним типів файлів (FTR). У таблиці 3 наведено матрицю рівнів складності транзакцій (низький, середній або високий) і відповідних балів (3, 4 або 6).

Таблиця 3

Матриця складності зовнішніх вхідних транзакцій

Типи файлів (FTR)	Рівень складності (DET)		
	1-4	5-15	Більше 15
0 – 1	Низький (3)	Низький (3)	Середній (4)
2	Низький (3)	Середній (4)	Високий (6)
Більше 2	Середній (4)	Високий (6)	Високий (6)

Зовнішні вхідні транзакції можуть бути бізнес-даними, даними управління і даними на основі правил. Прикладами бізнес-даних можуть бути ім'я клієнта, адреса, телефон тощо. Прикладами керуючих даних є дані, які викликають транзакцію або змінюють поведінку програми. Кожен прапорець представляє елемент даних. Крім того, кнопки для сортування співробітників і визначення тимчасового формату являють собою один елемент даних. Керуюча інформація змінює стан (поведінку) додатків, а також визначає, як і коли дані будуть оброблятися [5].

Наведемо приклади унікальних наборів даних, які дозволяють відрізняти зовнішні вхідні транзакції від інших транзакцій: поля введення даних; обчислені значення; повідомлення про помилки; повідомлення про підтвердження; рекурсивні поля, які враховуються як один DET; клавіші дії (командні кнопки); функціональні клавіші, які враховуються як один DET.

Унікальний FTR (File Types Referenced) також допомагає відрізняти оцінювану зовнішню вхідну транзакцію від інших транзакцій. FTR повинен бути або внутрішнім логічним файлом, або файлом зовнішнього інтерфейсу. Кожен внутрішній логічний файл, підтримуваний зовнішньою вхідною транзакцією, вважається FTR. Наприклад, зовнішній вхід може оновлювати внутрішній логічний файл, але також повинен посилатися на «файл безпеки», щоб переконатися, що у користувача є відповідні рівні безпеки. Це приклад двох FTR.

Існує ряд технічних особливостей, які дозволяють точно визначити кількість зовнішніх вхідних транзакцій: кнопки, де кожен набір перемикачів вважається одним DET, так як за один раз можна вибрати тільки одне значення перемикача; списки, що випадають можуть бути зовнішнім запитом, але результатом запиту має бути унікальне поле даних для зовнішнього введення; чекбокси – кожен прапорець, який може бути перевірений одночасно, є

унікальним DET; кнопки, які можуть вважатися унікальним полем даних для пов'язаної транзакції. Наступні типи документації можуть використовуватися для підрахунку зовнішніх вхідних транзакцій: документація з описом бізнес цілей; діаграма потоку даних; специфікація.

Зовнішні вихідні транзакції EO (External Outputs) – це елементарний процес, в якому похідні дані проходять через межу зсередини назовні. Крім того, вони можуть служити для поновлення файлів ILF. Вихідні дані створюють звіти або вихідні файли, що відправляються в інші додатки. Ці звіти і файли створюються з одного або декількох внутрішніх логічних файлів і файлу зовнішнього інтерфейсу.

Похідні дані – це дані, які обробляються за межами прямого пошуку і редагування інформації з внутрішніх логічних файлів або файлів зовнішнього інтерфейсу. Похідні дані зазвичай є результатом алгоритмів або розрахунків. Вони виникають, коли один або кілька елементів даних об'єднані з формулою для генерації або отримання додаткового елемента даних. Ці похідні дані не відображаються ні в одному FTR (внутрішній логічний файл або файл зовнішнього інтерфейсу).

Алгоритм визначається як механічна процедура для виконання заданого обчислення або покрокового вирішення проблеми. Розрахунок визначається як рівняння, яке має один або кілька операторів. Оператор є математичною функцією (додавання, віднімання, множення і ділення). Транзакції між додатками слід позначати інтерфейсами. Розроблюваний додаток може мати тільки зовнішній висновок або зовнішній запит даних по відношенню до додатка.

Як і всі компоненти, транзакції EO можуть бути оцінені. Рейтинг їх оцінювання заснований на кількості елементів даних (DET) і пов'язаних з ними типів файлів (FTR). Оцінка заснована на загальній кількості унікальних (комбінованих унікальних вхідних і зовнішніх сторін) елементів даних (DET) і типів файлів (FTR). У таблиці 4 наведено матрицю рівнів складності транзакцій (низький, середній або високий) і відповідних балів (4, 5 або 7).

Таблиця 4

Матриця складності зовнішніх вихідних транзакцій

Типи файлів (FTR)	Рівень складності (DET)		
	1-5	6-19	Більше 19
Менше 2	Низький (4)	Низький (4)	Середній (5)
2-3	Низький (4)	Середній (5)	Високий (7)
Більше 3	Середній (5)	Високий (7)	Високий (7)

Зовнішні вихідні транзакції містять інформацію, яка виводить дані через межу зсередини програми в зовнішній рівень взаємодії з користувачем. Деяка плутанина може виникнути через те, що EO має вхідну сторону. Вхідна сторона EO – це критерії пошуку та параметри. Мета інформації, що надходить ззовні додатку (вхідна сторона), полягає не в тому, щоб підтримувати файли ILF. Наприклад, на відміну від інших компонентів, EO майже завжди містить бізнес-дані. Базові дані про правила і «вихідні дані» на основі управління майже завжди вважаються зовнішніми запитами через те, що правила і типи керування не виводяться.

Попереджуючі повідомлення також вважаються транзакціями EO. Вони відрізняються від повідомлень про помилку. Даний вид повідомлень є елементарним процесом, в той час як повідомлення про помилку (або повідомлення підтвердження) є частиною елементарного процесу. Попереджуючі повідомлення є результатом обробки бізнес-логіки (наприклад, торговий додаток може повідомити брокера про те, що клієнт, який намагається розмістити замовлення, не має достатніх коштів на своєму рахунку).

Елементарний процес, пов'язаний із зовнішнім виходом, може оновлювати внутрішній логічний файл або файл зовнішнього інтерфейсу. Наприклад, елементарний процес, який проводить перевірку нарахування заробітної плати, може включати оновлення файлу, щоб встановити прапорець і вказати, що була проведена перевірка заробітної плати. Підтримується процес зміни даних (додавання, зміна та видалення) за допомогою елементарного процесу (через зовнішній вхід).

Зовнішні запити EQ (External Inquiries) – це елементарний процес з вхідними та вихідними компонентами, які призводять до вилучення даних з одного або декількох внутрішніх логічних файлів і файлів зовнішнього інтерфейсу. Процес введення не оновлює внутрішні логічні файли, а вихідна сторона не містить похідних даних. На рисунку 4 представлений приклад зовнішнього запиту з двома внутрішніми логічними файлами.

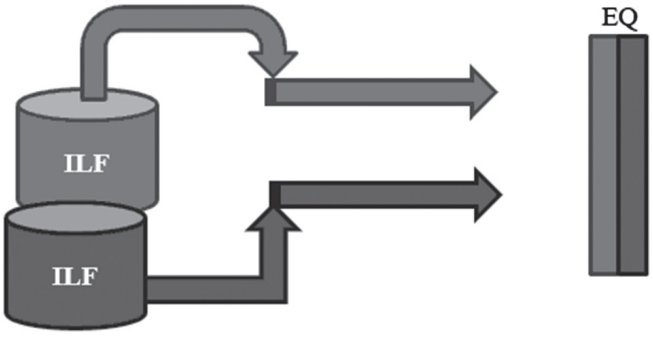

Рис. 4. Формування зовнішнього запиту

Як і всі компоненти, зовнішні запити EQ можуть бути оцінені. В цілому, EQ оцінюється (низький,

середній або високий), як EO, але запитам присвоюються значення типу EI. Оцінка заснована на загальній кількості унікальних елементів даних (DET) і типів файлів (FTR). Якщо один і той же FTR або DET використовується як на вхідний, так і на вихідній стороні, він підраховується тільки один раз. Матриця складності зовнішніх запитів наведена в таблиці 5.

Таблиця 5

Матриця складності зовнішніх запитів

Типи файлів (FTR)	Рівень складності (DET)		
	1-5	6-19	Більше 19
Менше 2	Низький (3)	Низький (3)	Середній (4)
2-3	Низький (3)	Середній (4)	Високий (6)
Більше 3	Середній (4)	Високий (6)	Високий (6)

Запити EQ можуть містити бізнес-дані, дані управління і дані, засновані на правилах. Прикладами бізнес-даних є імена клієнтів, адреси, номер телефону, список, що випадає (наприклад, список клієнтів по імені). Приклади зовнішніх запитів: натискання миші; значення пошуку; командні кнопки; скролінг; значення, що надійшли з внутрішнього логічного файлу або файлу зовнішнього інтерфейсу; зміна кольору або шрифту на екрані; рекурсивні поля.

Внутрішні логічні файли (ILF) – це ідентифікована користувачем група логічно пов'язаних даних, яка знаходиться на межі додатку і підтримується через зовнішні входи. Внутрішній логічний файл має деяку логічну структуру і зберігається в файлі. Хоч це і не обов'язково, але зазвичай файл ILF має як мінімум один зовнішній вивід або один зовнішній запит. Тобто, принаймі один зовнішній висновок або зовнішній запит повинні включати ILF як FTR. Інакше кажучи, інформація зберігається в ILF, тому її можна використовувати пізніше. Відзначимо, що транзакції EO або запити EQ можуть приходити з іншої програми.

Як і всі компоненти, файл ILF може бути оцінений. Рейтинг такої оцінки заснований на кількості елементів даних (DET) і типах записів (RET). У таблиці 6 зазначені рівні складності (низький, середній або високий) і відповідні бали (7, 10 або 15).

Таблиця 6

Матриця складності внутрішніх логічних файлів

Групи даних (RET)	Рівень складності (DET)		
	1-19	20-50	Більше 51
1	Низький (7)	Низький (7)	Середній (10)
2-5	Низький (7)	Середній (10)	Високий (15)
Більше 6	Середній (10)	Високий (15)	Високий (15)

Файли ILF можуть містити бізнес-дані, дані управління і дані, засновані на правилах. Тип даних, що містяться в ILF, являє собою той же тип даних, який містить і підтримує EI. Зазвичай керуючі дані мають тільки одне входження в ILF. Скажімо, файл керуючих даних може містити тільки параметри або налаштування стану. Наприклад, частина бортової автомобільної системи містить тільки поточну інформацію (тиск масла, температуру двигуна тощо). Цей конкретний процес бортової системи дбає не про історичні дані, а тільки про поточні екземпляри. Коли статус змінюється, файл оновлюється поточною інформацією, а історична інформація відсутня. Бортова система може відстежувати історичні зміни в файлах діагностики, але це буде абсолютно окремий процес. Цей процес не використовується, щоб підтримувати роботу автомобіля, а тільки щоб допомогти механіку зрозуміти, що відбувається з двигуном.

Ідея застосування RET (реального часу) полягає в кількісному визначенні складних відносин між даними, які підтримуються в одному FTR. Прикладом систем RET є, зокрема, система телефонної комутації, де використовуються три типи даних: бізнес-дані, дані правил і дані управління. В цьому випадку, бізнес-дані – це фактичний виклик, дані правил – це те, як виклик повинен бути направлений через мережу, а дані управління – це те, як комутатори взаємодіють один з одним. Подібно керуючим файлам, в звичайних системах реального часу буде тільки одне входження у внутрішній логічний файл. Логічні групи даних – одна з найскладніших концепцій в аналізі функціональних точок. Більшість логічних груп залежать від відносин між батьками і дочірніми елементами.

Файли зовнішнього інтерфейсу (EIF) – це ідентифікована користувачем група логічно пов'язаних даних, яка використовується тільки для довідкових цілей. Дані цих файлів повністю знаходяться за межами додатку і підтримуються іншими додатками зовнішніх входів. Файл зовнішнього інтерфейсу є внутрішнім логічним файлом для іншої програми. Кожен EIF повинен мати принаймні один зовнішній файл, одну транзакцію, зовнішній вхід, зовнішній вихід або зовнішній запит.

Як і всі компоненти, складність EIF може бути оцінена. Оцінка заснована на кількості елементів даних (DET) і типах записів (RET). У таблиці 7 зазначені рівні складності EIF (низький, середній або високий) і відповідні бали (5, 7 або 10).

Таблиця 7

Матриця складності зовнішніх інтерфейсів

Групи даних (RET)	Рівень складності (DET)		
	1-19	20-50	1-19
1	Низький (5)	Низький (5)	Середній (7)
2-5	Низький (5)	Середній (7)	Високий (10)
Більше 6	Середній (7)	Високий (10)	Високий (10)

3. Визначення кількості невирівняних та вирівняних функціональних точок

Загальний обсяг продукту в невирівняних функціональних точках (UFP) визначається шляхом підсумовування для всіх інформаційних об'єктів і елементарних операцій:

$$MFP = \sum_{i=1}^{ILF} MFP_i + \sum_{i=1}^{EI} MFP_i + \sum_{i=1}^{EO} MFP_i + \sum_{i=1}^{EQ} MFP_i \,. \quad (1)$$

Коефіцієнт вирівнювання (VAF) враховує 14 загальних характеристиках системи, які оцінюють загальну функціональність програми. Ступені впливу при цьому варіюються від нуля до п'яти: 0 – вплив відсутній; 1 – випадковий вплив; 2 – помірний вплив; 3 – середній вплив; 4 – значний вплив; 5 – сильний вплив.

Коефіцієнт вирівнювання обчислюється наступним чином:

$$VAF = 0.65 + 0.01 \sum_{i=1}^{14} C_i \,, \quad (2)$$

де C_i – ступінь впливу на кожну системну характеристику продукту; i – системні характеристики продукту.

Основним недоліком використання коефіцієнта вирівнювання є частковий суб'єктивізм, так як різні фахівці можуть по-різному оцінити той чи інший параметр. Для уніфікації процесу оцінки слід користуватися докладним описом кожного з факторів і відповідної оцінки.

До основних системних характеристик програмного продукту слід віднести: обмін даними; розподілена обробка даних; продуктивність; обмеження по апаратних ресурсах; ступінь транзакційного навантаження; інтенсивність взаємодії з користувачем; ергономічність; ступінь ефективності роботи кінцевих користувачів; інтенсивність зміни даних користувачами; складність обробки даних; можливість повторного використання; зручність інсталяції; зручність адміністрування; кількість портів (процесорів), що можуть бути залучені до обробки даних; гнучкість. Для прикладу у таблиці 8 наведено опис однієї з цих характеристик (ступеня впливу обміну даними на розроблювану програмну систему).

Подальша оцінка вирівняних функціональних точках (AFP) залежить від типу оцінки. Наприклад, початкова оцінка кількості вирівняних функціональних точок (AFP) для програмного додатка, що враховує тільки нову функціональність, яка реалізується в продукті, визначається так:

$$AFP = UFP * VAF \,. \quad (3)$$

Проект, який знаходиться в стадії розробки оцінюється в *DFP* (Development Functional Point) за такою формулою:

$$DFP = (UFP + CFP) * VAF \,, \quad (4)$$

де CFP – функціональні точки, підраховані для додаткової функціональності, яка буде потрібна при установці продукту (наприклад, при міграції даних).

Таблиця 8

Матриця складності зовнішніх інтерфейсів

Оцінка	Опис для визначення ступеня впливу
0	Продукт є автономним додатком
1	Додаток не є пакетним, але має віддалений ввод даних або віддалений друк
2	Додаток є пакетним
3	Додаток передбачає онлайн-збір даних для пакетного процесу або системи запитів
4	Логіка програмного додатку винесена на рівень інтерфейсів, що підтримують лише один тип протоколу звязку
5	Продукт обмінюється даними больш ніж по одному телекомунікаційному протоколу

Проект доопрацювання і вдосконалення продукту оцінюється за такою формулою:

$$EFP = (ADD + CHGA + CFP) * \\ *VAFA + (DEL * VAFB) \,, \quad (5)$$

де ADD – функціональні точки для доданої функціональності; $CHGA$ – функціональні точки для змінених функцій, розраховані після модифікації; $VAFA$ – величина фактора вирівнювання, розрахованого після завершення проекту; DEL – обсяг вилученої функціональності; $VAFB$ – величина фактора вирівнювання, розрахованого до початку проекту.

Сумарний вплив процедури вирівнювання знаходиться в межах ± 35% відносно обсягу, розрахованого в *UFP*.

4. Корекція оцінки складності проекту з урахуванням ризиків

Наступним кроком запропонованого методу є визначення допустимого порогу ризиків, з якими розробник буде готовий взяти проект в розробку. Для кожного розробника існує граничне значення, в рамках якого менеджмент-команда зможе продуктивно управляти ризиками. Коригування МФТ-оцінки передбачає ідентифікацію позитивних і негативних ризиків на кожному з етапів проекту: аналіз і опрацювання вимог, проектування і дизайн, розробка, тестування і підтримка. Далі, кожен ризик аналізується за параметрами ймовірності його виникнення та його впливу на проект. Це дозволяє визначити пріоритет для обчислення діапазонів допустимих значень по кожному з етапів розробки програмного забезпечення.

Для того щоб почати пом'якшувати або усувати ризики потрібно прояснити які саме з них небезпечні. Це можна зрозуміти шляхом розрахунку

пріоритету ризиків. Щоб розрахувати пріоритет ризику, потрібно мати два основних параметри — коефіцієнт впливу ризику (Impact) і ймовірність виникнення ризику (Probability). Розрахунок пріоритету ризику здійснюється за такою формулою:

$$S = I * P, \quad (6)$$

де I – коефіцієнт впливу ризику; P – ймовірність виникнення ризику.

Коефіцієнти впливу ризику і ймовірності виникнення ризику можуть набувати значення в діапазоні [0.01; 1]. Отже, пріоритет ризику може приймати ті ж значення. Пріоритетність ризиків визначається наступними діапазонами:

– [0.01; 0.05] — низький пріоритет;
– [0.06; 0.17] — середній пріоритет;
– [0.18; 1.00] — високий пріоритет.

Найчастіше, розробка програмного забезпечення складається з декількох етапів, які займають ту, чи іншу частину робочого часу. Умовно, розробку програмного засобу можна розділити на наступні фази: аналіз і опрацювання вимог (20% часу реалізації проекту); проектування та дизайн (25% часу реалізації проекту); розробка (40% часу реалізації проекту); тестування (10% часу реалізації проекту); підтримка (5% часу реалізації проекту).

Згідно з методикою PERT та методом Монте-Карло, тривалість кожної операції має межі, які виходять з статистичного розподілу. PERT використовує 3 оцінки розрахунку часу для кожного завдання або проекту. Це означає, що тривалість кожного завдання має межу від оптимістичного (найкращого) до песимістичного (найгіршого), і середній показник можна розрахувати для кожної операції. Для вираження тривалості операції в PERT використовується апроксимація бета-розподілу. Згідно з цим можна зробити висновок, що різниця між оптимістичною і найбільш вірогідною оцінкою в 1,75 разів менше ніж між найбільш імовірною і песимістичною. Дана ситуація пов'язана з тим, що в другій відрізок закладається ймовірність виникнення ризиків. У зазначеній різниці враховуються ризики з усіма стратегіями, крім стратегії усунення, так як превентивні заходи по обробці даного ризику повинні бути впроваджені в проект до його старту. Цей фактор доцільно брати до уваги для визначення меж діапазонів допустимих значень по кожній фазі на етапі корекції МФТ-оцінок складності проекту.

Для попередження виникнення максимально негативного сценарію проектування, слід визначити допустимий рівень ризиків, з яким ІТ — компанія зможе взяти проект в роботу, і скорегувати фінальну оцінку відповідним чином. Для цього пропонується використання принципу ALARP (As Low As Reasonably Practicable, що означає мінімальний практично прийнятний ризик. Цей принцип дозволяє визначити поріг допустимості ризику шляхом зниження ризику аж до того моменту, коли прийняття додаткових заходів щодо зниження ризику, будучи технічно здійсненним, буде визнано непропорційно витратним. Ситуація, коли менеджери зможуть звести на нуль або зменшити ймовірність і ступінь впливу абсолютно всіх ризиків практично неможлива. Принцип ALARP вважає за доцільне прийняти наявність тих чи інших ризиків і не витрачати час і бюджет на зниження їх ймовірності і мінімізацію впливу ризиків. Згідно з принципом ALARP, всі ризики можна оцінити і розділити на три категорії: високопріоритетні (пріоритет 0.66 – 1.00); середньопріоритетні (пріоритет 0.33 – 0.65); фонові (пріоритет 0.01 – 0.32).

5. Тестування гібридного методу оцінки складності проекту

Запропонований метод було протестовано на прикладі оцінки складності проектування додатку «Hostel Issue Reporting», яке дозволяє відправляти запит з описом проблеми, що виникла з обладнанням в готелі. Опис функцій програми: користувач вибирає одну з категорій виниклих проблем на головному вікні; завантажується майстер «звіт про проблеми», де користувач вибирає адреса будівлі, номер поверху (необов'язково), службу, проблему, надає опис проблеми, вказує своє ім'я, номер телефону і відправляє запит; запис про нову проблему автоматично створюється в довідковій службі адміністратора; запис про підтвердження запиту відображається користувачеві зі створеним номером запиту, який може використовуватися пізніше для довідки. Також додаток містить прихований екран «Налаштування адміністратора», що використовується при першій установці. Для першого запуску додатка адміністратор повинен встановити адресу будівлі, в якому пристрій буде встановлено. Крім того, адміністратор може додавати додаткові категорії, крім тих, які встановлені за замовчуванням. Основне екранне вікно додатку міститьє коротку інструкцію про те, як використовувати додаток, і пропонує вибрати категорію проблем.

Згідно з запропонованим методом була сформована модель функціональних точок для програмного засобу, що реалізує інформаційну систему управління запитів про несправність обладнання, та визначені межі проектованого програмного засобу. Очевидно, що цей засіб є локальним і обмін даними з іншими програмними засобами не передбачає.

Розроблюваний програмний засіб буде працювати з локальною базою даних, що складається з 7

таблиць – «User», «Category», «Request», «Problem», «Address», «e-mail», «RequestStatus». Таким чином, цей засіб буде мати 7 внутрішніх логічних файлів (ILF). Всі внутрішні логічні файли містять від 1 до 3 типів елементів записів (RET): символьний формат, числовий формат і формат дата / час. Число типів елементів даних внутрішніх логічних файлів DET – від 2 до 5. Таким чином, рівень складності внутрішніх логічних файлів низький [6].

У програмному засобі є 19 зовнішніх входів (EI), що характеризуються різними рівнями складності. Крім того, засіб має: 15 зовнішніх виходів, що відповідають різним повідомленням при додаванні або видаленні записів (12 мають низький рівень складності, 3 – середній); 9 зовнішніх запитів, що представляють собою поля введення для пошуку, виведення результату пошуку тощо. (7 мають низький рівень складності, 2 – високий). Основними характеристиками для даної системи є: обмін даними (оцінка 2); розподілена обробка даних (оцінка 3); (інтенсивність взаємодії з користувачем (оцінка 4); ергономічність (оцінка 5; гнучкість (оцінка 2). Інші характеристики не мають суттєвого значення для даного програмного продукту. Сумарне значення ваг всіх розглянутих характеристик дорівнює 14. Кількість рядків коду додатку *SLOC* була визначена за методом, заснованим на використанні так званого «мовного множника», який представляє собою середню кількість рядків вихідного коду конкретної алгоритмічної мови, що припадає на одну нормовану функціональну точку:

$$SLOC = AFP * LM , \quad (7)$$

де LM – мовний множник.

Згідно з розрахунками за запропонованим методом були отримані наступні значення показників складності розробки додатку «Hostel Issue Reporting»: UFP – 202; VAF – 0.81; AFP – 163.62; $SLOC$ – 8671.86.

Фінальна оцінка складності розробки продукту була представлена в людино-годинах (МН), кожна одиниця яких має відповідний грошовий еквівалент:

$$MH = AFP * C , \quad (8)$$

де C – коефіцієнт, що визначає, скільки людино-годин еквівалентно одній функціональній точці (зазвичай приймається рівним 6).

Таким чином, згідно з розрахунками, для розробки додатку «Hostel Issue Reporting» MH = 982 люд.-год.

Використання розрахунків за методом Монте-Карло (для MH = 982 люд.-год.) дозволило зробити висновок, що аналіз і опрацювання вимог будуть виконані за 196 люд.-год., проектування і дизайн за 246 люд.-год., розробка за 393 люд.-год., тестування за 98 люд.-год., підтримка за 43 люд.-год. Було проаналізовано вплив ризиків по кожному із зазначених етапів для визначення діапазонів допустимих значень. Наприклад, в рамках фази аналізу і опрацювання вимог існує ризик того, що вимоги не будуть з'ясовані в повній мірі, що призведе до незапланованого обсягу робіт. Імовірність виникнення даного ризику 40%, ступінь впливу на проект 80%, отже, пріоритет ризику дорівнює 32%.

З оглядом на вплив негативних ризиків (32%) в межах фази аналізу і опрацювання вимог, можемо зробити висновок, що різниця між оптимістичною і найбільш вірогідною оцінкою становитиме 18%. За результатами розрахунків були визначені граничні значення по кожній фазі розробки програмного продукту з урахуванням впливу негативних і позитивних ризиків.

Адаптуючи дану модель за принципом ALARP, можна зробити висновок, що фінальна оцінка складності проекту повинна бути завищена з урахуванням часу, необхідного на обробку ризиків і застосування відповідної стратегії. У базовому експерименті порогове значення у вигляді оцінки складності програмного продукту становило 982 людино-години. Збільшення цієї оцінки (за допомогою створення буфера на обробку високопріоритетних і середньопріоритетних ризиків) до 1055 людино-годин дозволить дати максимально об'єктивну, а головне конкурентоспроможну на ринку оцінку, що суттєво збільшить ймовірність успішного закінчення проекту.

Висновки

У статті був розглянутий модифікований гібридний підхід до оцінки складності IT проектів на основі удосконалення методу функціональних точок за рахунок застосування механізмів ризик-менеджменту та прогнозування зусиль, пов'язаних з розробкою програмних систем.

Оскільки більшість комп'ютерних систем взаємодіють з іншими системами, то мають бути визначені межі поділу між розроблюваним проектом або додатком і зовнішніми додатками або доменом користувача. Після встановлення таких меж, компоненти розроблюваного програмного забезпечення можуть бути класифіковані, ранжовані і підраховані.

Загальний обсяг продукту в невирівняних функціональних точках (UFP) визначається шляхом підсумовування для всіх інформаційних об'єктів і елементарних операцій. Це дозволяє визначити кількості невирівняних та вирівняних функціональних точок в програмному продукті, що розроблюється.

Наступним кроком запропонованого методу є визначення допустимого порогу ризиків, з якими

розробник буде готовий взяти проект в розробку. Коригування МФТ-оцінки передбачає ідентифікацію позитивних і негативних ризиків на кожному з етапів проекту. Кожен ризик аналізується за параметрами ймовірності його виникнення та його впливу на проект. Це дозволяє визначити пріоритет для обчислення діапазонів допустимих значень по кожному з етапів розробки програмного забезпечення.

В статті наведено результати тестування запропонованого методу на прикладі оцінки складності проектування програмного додатку «Hostel Issue Reporting», яке дозволяє відправляти запит з описом проблеми, що виникла з обладнанням в готелі. Результати тестування свідчать про те, що метод дозволяє формувати обґрунтовані рекомендації щодо оцінювання реальної складності ІТ проектів.

Список літератури:

[1] Филипенко О. М. Управління проектами: навч. посібник / О. М. Филипенко, Т. С. Колєснік. – Харків : ХДУХТ, 2016. – 161 с

[2] Herron D. E. Function Point Analysis: Measurement Practices for Successful Software Projects [Текст] / D. E. Herron, D. Garmus, – T.: Addison-Wesley Professional, 2001. – 39 p.

[3] Жмаева Ю.В. Адаптивное прогнозирование оптимального количества ресурсов IT проекта по методологии Agile [Текст] / Ю.В. Жмаева, Л.Э. Чалая, С.Г. Удовенко// АСУ и приборы автоматики. – 2015. – № 173. – С. 4–13.

[4] Capers, J. The Economics of Software Quality [Текст] / J. Capers– N.: Pearson Education, 2011. – 62 p.

[5] Cooper, B. The Lean Entrepreneur: How Visionaries Create Products, Innovate with New Ventures, and Disrupt Markets [Текст] / Cooper, B. – London: Wiley, 2016. – 133 p.

[6] Жмаєва Ю.В. Оцінювання складності розробки програмного забезпечення з використанням модифікованого методу фунціональних точок/ Ю.В. Жмаєва, Л.Е. Чала, С.Г. Удовенко // Праці Міжнародної науково-практичної конференції «Математичне моделювання процесів в економіці та управлінні проектами і програмами (ММР-2018)». – Харків-Миколаїв, 2018. – С.58-61

Надійшла до редколегії 10.10.2018