ПРАКТИЧНА РОБОТА №4

ДОДАТКОВІ ВИДИ ТОЧОК ДЛЯ ОЦІНКИ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Мета: Навчитися оцінювати розмір програмного забезпечення шляхом застосування додаткових видів точок, а саме Feature points, Object points та Use-case points.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Оскільки метод функціональних точок вперше був використаний для інформаційних систем, фахівці думали про нього як про метод, який можна використовувати для такого роду програмного забезпечення. З цієї причини виникла величезна кількість модифікованих концепцій цього методу.

Сьогодні існує близько 35 різних версій, що застосовуються промисловістю. Найбільш поширені наступні:

- Feature points;
- Full function points;
- Object points;
- Data points;
- Story Points;
- Use-case points.

Feature points. Цей тип точок призначений для визначення функціонального розміру програмного забезпечення, особливо для програмного забезпечення в реальному часі або вбудованих програм. Метод функціональних точок був створений для оцінювання розміру програмного забезпечення Management Information Systems (MIS) але він був неточний для оцінювання розміру систем реального часу, або систем, які містять реалізацію складних алгоритмів. Тому Capers Jones розробив додатковий до Function Point від точок, який назвав Feature points. Найбільш помітна різниця між Function Point та Feature points полягає в тому, що останні використовують додатковий компонент, алгоритми, додаючи його до набору з п'яти компонентів Function

Point: inputs, outputs, inquiries, external interface files, and internal logical files. Більш важкі системи, схоже, мають більшу алгоритмічну складність, ніж програмне забезпечення MIS, але менше inputs, outputs, inquiries, external interface files, and internal logical files. Тому підрахувати такі системи за допомогою Function Point може бути трохи оманлива. Метод Feature Point є хорошою альтернативою для цього питання, оскільки він компенсує ці проблеми, враховуючи алгоритми. Кожному алгоритму також присвоюється значення ваги, як і решті компонентів. Крім того, значення, призначені для файлів логічних даних, зменшуються, оскільки вони менш значущі для більш важких типів систем. Таким чином, Unadjusted Feature Point обчислюються наступним чином:

Unadjusted Feature Point = (External Inputs * Weight) +
(External Outputs * Weight) + (Logical Internal Files * Weight) +
(Logical Interface Files * Weight) + (Inquiries * Weight) +
(Algorithms * Weight)

Посібник з практики підрахунку Function Point дає конкретні вказівки щодо визначення "ступеня впливу" від 0 до 5 для кожної з чотирнадцяти загальних характеристик системи. Вони також пропонуються для використання в Value Adjustment Factor для Feature Points. Цей розрахунок надає the Adjusted Feature Point для Feature Points.

Таким чином:

Value Adjustment Factor = Total Degree of Influence * .01 + .65

Нарешті,

Adjusted Feature Points = (Unadjusted Feature Points) *

(Value Adjustment Factor)

Full function points (FFP), метод який є розширенням FPA для систем реального часу. Він базується на роботі, проведеній the Software Engineering Management Research Laboratory at the Université du Québec à Montréal and

Software Engineering Laboratory in Applied Metrics (SELAM), і був представлений в 1997 році.

Програмне забезпечення систем реального часу містять більше даних управління та під процесів, ніж програмне забезпечення ІС. Оскільки FFP є продовженням FPA, усі правила IFPUG включені до FFP, але незначна кількість підмножин правил IFPUG, що стосуються концепцій управління, була значно розширена. FFP вирішує це шляхом введення додаткових типів Data Functions та Transaction Functions.

FFP, як і FPA, вимірює функціональний розмір, оцінюючи Data Functions та Transaction Function. Однак, на відміну від FPA, Transaction Function оцінюються на рівні під процесу. Тобто, в Processing Logic Form ідентифікуються підпроцеси, які класифікуються та оцінюються. Значення факторів функціонального розміру (FTR) нараховуються на рівні під процесу. Додатково підпроцеси в рамках кожного процесу Transaction Function класифікують на один із чотирьох типів:

- зовнішній контроль (ECE);
- зовнішній вихід управління (ECX);
- зчитування внутрішнього контролю (ICR);
- запис внутрішнього контролю (ICW).

Object Points (OP), які також називаються application points, щоб уникнути непорозумінь з об'єктами в об'єктно-орієнтованих системах, засновані на кількості вікон, звітів, модулів 3GL та правил. Для обчислення Object Points обчислюють кількість екранів, звітів та модулів, які було створено за допомогою інтегрованих середовищ інженерії програмного забезпечення (CASE). Зважаючи за допомогою таблиць результати обчислень та сумуючи з урахування повторно використаного коду отримують кількість Object Points.

Модель композиції програми (Application Composition Model – ACM) включає оцінку прототипу користувальницького інтерфейсу. Вона використовується на ранніх стадіях розробки проєкту та орієнтована на проєкти, що створюються із застосуванням сучасних інструментальних засобів та UML.

Мета моделі: оцінка інтерфейсу користувача, взаємодії із системою, продуктивності. Така оцінка дозволяє вибрати кілька можливих варіантів подальшого розвитку проєкту. ACM використовує як метрику розміру програмного забезпечення об'єктні одиниці.

Об'єктна одиниця визначається шляхом підрахунку кількості екранів інтерфейсу користувача, звітів і компонентів. Кожен із цих вимірів оцінюється за складністю.

Складність екрану оцінюється в залежності від кількості таблиць даних і кількості подання екранів (простий екран – 1, складний – 3). Складність звіту також оцінюється в залежності від кількості його подань (простий – 2, середній – 5, складний – 8).

Окремі компоненти, що відображають обчислювальні компоненти, архітектурні перебудови, генерацію графіки, вважаються складними та мають коефіцієнт 10.

Якщо припустити, що у проєкті буде використано r відсотків об'єктів із раніше створених проєктів, кількість нових об'єктних одиниць у проєкті Object Points (OP) можна розрахувати, як:

$$OP_{new} = OP \cdot \frac{100 - r}{100},$$

де OP – кількість об'єктних одиниць у раніше розроблених проєктах;

OPnew – кількість нових об'єктних одиниць у розглянутому проєкті.

Трудовитрати розраховуються за формулою:

$$E = \frac{OP_{new}}{PROD},$$

де E – трудовитрати у людино-місяцях;

PROD – продуктивність, встановлюється залежно від досвідченості працівника та зрілості середовища розробки.

Оцінюється за п'ятиінтервальною шкалою:

- дуже низька - 4;

- низька – 7;

- номінальна – 13;

- висока – 25;

- дуже висока – 50.

Story Points використовуються командами Agile під час спринтерської сесії. Кожної історії користувача (функція програмного забезпечення) призначається значення Story Points , засноване на рівні складності конкретної історії (функції). Балі - відносна міра виражається відповідно до числового діапазону, який, як правило, є обмеженим набором чисел. Числа вибираються на основі сприйняття колективом розміру роботи, яку необхідно виконати. Визначення розміру базується на рівні розуміння ступені складності проєкту, можливості команди та кількості роботи. Особливістю Story Points є те, що кожна команда визначає їх так, як вважає за потрібне. Одна команда може вирішити визначити Story Points як ідеальний робочий день (тобто день без будь-яких перерв - без зустрічей, без електронної пошти, без телефонних дзвінків, і так далі). Інша команда може визначити Story Points як ідеальний тиждень роботи або може визначити Story Points в іншої мірі складності історії.

Use-case points. У 1993 р. Густавом Карнером (Gustav Karner) з Objectory (нині Rational Software) був розроблений метод, заснований на аналізі Use-cases, а також Activity діаграм для визначення та оцінки проєктів, що створено за об'єктно-орієнтованим методом. З використанням Use-case points для оцінювання розміру існує два метода:

1. Ідентифікувати actors і Use-case.

2. Обчислити, як для функціональних точок, але використовуючи Use-case діаграми та Activity діаграми, кількість input, output файлів і data inquiries.

Відповідно до першого методу Use-case points обчислюються наступним чином:

1. Ідентифікуються і зважуються actors (UAW). Цей крок, полягає в тому, щоб класифікувати акторів як простих, середніх або складних. Простий актор представляє іншу систему з визначеним API інтерфейсом програмування,

середній актор це інша система, що взаємодіє через протокол, такий як TCP/IP, і складний актор може бути особа, яка взаємодіє через графічний інтерфейс або веб-сторінку. Вагові коефіцієнти присвоюється кожному типу актора (табл. 14).

Таблиця 14 – Оцінка акторів (UAW)

Тип	Коефіцієнт
Простий (використовує систему за допомогою перерозподіленого API)	1
Середній (використовує більш складний або гнучкий API)	2
Складний (в основному означає взаємодію з кінцевим користувачем)	3

2. Ідентифікуються і зважуються Use-case (UUCW). На цьому кроці кожен Use case визначається як простий, середній або складний, залежно від кількості транзакцій в описі Use case, включаючи вторинні сценарії. Транзакція є сукупність видів діяльності, яка або виконується повністю, або взагалі не виконується. Підрахування кількості транзакцій можна здійснити, підраховуючи кроки Use cas.

Іншим механізмом вимірювання складності Use case є підрахунок кількості класів, які будуть використано в містах транзакцій, як тільки буде визначено, які класи реалізують конкретний Use case. Use case простий складності – такий, що реалізований 5 або меншій кількістю класів, Use case середній складності - на 5-10 класів, а складний Use case, це більш ніж на десять класів. Ваги такі, як і раніше.

3. Обчислити Unadjusted Use-case points:

$$UUCP = UAW + UUCW$$

4. Обчислити значення Adjustment Factors (VAF). Коефіцієнти технічних факторів відповідають технічним факторам коригування складності методу FPA, а фактори зовнішнього середовища це множник для кількісної оцінки нефункціональних вимог, таких, наприклад, як простота використання та мотивація програміста. Фактори, що впливають на продуктивність,

асоціюються з вагою, а значення призначається кожному фактору залежно від ступеня впливу. Так 0 означає відсутність впливу, 3 є середнім, 5 означає сильний вплив.

Technical Complexity Factor Фактор технічної складності (TCF) обчислюється множенням значення кожного з них фактор (T1-T9) за його вагою, а потім додаючи всі ці числа, щоб отримати суму, що позначається як TFactor. Застосовується наступна формула:

$$TCF=0.6 + (0.01 * TFactor)$$

Environmental Factor фактор середовища (EF) обчислюється множенням значення кожного фактора (F1- F8) за його вагою та додаванням результатів, щоб отримати суму, що позначається EFactor. Застосовується наступна формула:

$$EF = 1.4 + (-0.03 * EFactor)$$

5. Обчислити Adjusted Use-case points (UCP = UUCP * VAF).

Data Points (Точки даних), пов'язані з підрахунком функціональних можливостей систем на основі бази даних, які були б корисними для визначення вартості програмного забезпечення, яке містить дуже велику базу даних. Будь-яке програмне забезпечення завжди складається з двох частин. Одна частина зберігає дані, а інша частина обробляє. Програма, велика чи маленька, містить деякі дані.

Можна розрахувати точки даних для програмного забезпечення з наступних десяти параметрів:

1) Обсяг даних. Фактична середня кількість трансакцій, які за день обробляються нашим програмним забезпеченням.

2) Тип системи управління базами даних, яка використовується для управління даними. Наприклад, RDBMS, ORDBMS, бази даних SQL тощо. Щоб визначити вагу бази даних, ми можемо враховувати різні фактори, такі як надані функції безпеки, підтримка PL/SQL, ємність керування транзакціями, керування відновленням, ємність зберігання тощо.

3) Архітектура середовища, в якому буде здійснюватися доступ до даних. Наприклад, однокористувацький, багатокористувацький, веб-орієнтований, клієнт-сервер тощо.

4) Загальна кількість таблиць (відношень). Наприклад, прості таблиці, розділені таблиці, вкладені таблиці тощо.

5) Складність ключів. Складність ключів можна визначити з атрибутами, що містять первинні або зовнішні ключи.

6) Складність відносин. Кількість об'єктів, крім таблиць. Наприклад, збережені процедури, збережені функції, тригери, методи тощо. Для цих типів об’єктів можна обчислити загальну кількість рядків кодів або обчислити FP за допомогою будь-якого стандартного методу, наприклад FPA, COCOMO або будь-якого іншого методу.

7) Загальна кількість індексів. Складність індексів залежить від загальної кількості полів, що використовуються для створення індексу.

8) Обсяг щоденних транзакцій, що обробляють базу даних.

9) Загальна кількість складних атрибутів. Наприклад, для зберігання зображень, документів, аудіо- та відео файлів тощо.

Для визначення ваги кожного компонента бази даних відповідно до їх складності та обсягу можна використовувати вагові показники. Рішення про вагові показники залежить від організації, тому що компанія, яка фактично розробляє програмне забезпечення, краще знає про вартість усіх компонентів, які використовуються при розробці системи. Розрахунок DCP відбувається за наступною формулою:

$$DPC = \sum_{i=1}^{n} (W_i * V_i)|$$

Після розрахунку кількості точок даних (DPC) його можна додати до Function Point Count, щоб оцінити остаточну вартість програмного забезпечення.

ЗАВДАННЯ

1. Використовуючи власні або запропоновані викладачем програмні застосунки обчислити наступне:

- кількість Feature points;

- Кількість Object Points;

- кількість Use-case points за першим і другим методами.

2. Визначити економічний вплив отриманих результатів на реалізацію програмного застосунка.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Які види додаткових точок існують і чому.

2. Поясніть потребу у розробці кожного виду додаткових точок.

3. Як обчислюються кількість алгоритмів.

4. Поясніть сутність Feature Point Analysis.

5. Поясніть концепцію Full Function Points.

6. Поясніть як враховується повторне використання при обчисленні точок додаткових видів.

7. Поясніть сутність Object Points.

8. Поясніть сутність Object Points Analysis.

9. Поясніть сутність Use-case points.

10. Поясніть сутність Use-case points Analysis за першим методом.

11. Поясніть сутність Use-case points Analysis за другим методом.