Лабораторна робота 6

Тема: Об’єктно-орієнтовані метрики коду

Мета: навчитися підраховувати та аналізувати метрики об'єктно-орієнтованої розробки: інкапсуляції, спадкування, поліморфізму і отриманих повідомлень.

Об'єктно-орієнтовані метрики вводяться з метою:

- поліпшити розуміння якості продукту;
- оцінити ефективність розробки;
- поліпшити якість роботи на етапі проектування.

Одними з найбільш поширеніших наборів об’єктно-орієнтованих метрик є метрики Чидамбера-Кемерера та метрики Фернанда Андреу (MOOD).

Метрики Чидамбера-Кемерера орієнтовано на класи, які є фундаментальним елементом об'єктно-орієнтованого підходу. Тому вимірювання та метрики для окремого класу, ієрархії класів та взаємодії класів є важливими для інженера-програміста, якому необхідно оцінити якість проекту.

Метрика 1: Зважені методи на клас WMC (Weighted Methods Per Class) - відносна міра складності класу

Припустимо, в класі C визначено n методів зі складністю $c_1...,c_i,...,c_n$. Кількість методів і їх складність є індикатором витрат на реалізацію і тестування класів. Крім того, чим більше методів, тим складніше дерево наслідування (всі підкласи успадковують методи батьківського класу). З ростом кількості методів у класі його застосування стає більш специфічним і тим самим обмежується можливість багаторазового використання. За цих причин метрика WMC повинна мати розумно низьке значення:

$$WMC = \sum_{i=1}^{n} c_i$$

Дуже часто застосовують спрощену версію метрики. При цьому припускають, що $c_i = 1$, і тоді WMC - кількість методів у класі.

Варіанти обліку методів в класі:

- Підраховуються тільки методи поточного класу. Успадковані методи ігноруються. Обґрунтування - успадковані методи вже підраховані в тих класах, де вони визначалися. Таким чином, інкрементність класу - кращий показник його функціональних можливостей, який відображає його право на існування.
- Підраховуються методи, які визначено у поточному класі, і всі успадковані методи. Цей підхід підкреслює важливість простору станів у розумінні класу (а не інкрементного класу).

Існує ряд проміжних варіантів. Наприклад, підраховуються поточні

методи і методи, які прямо наслідуються від батьківських класів. Аргументом на користь такого підходу є те, що на поведінку дочірнього класу найбільше впливає специфікація батьківських класів.

Метрика 2: Висота дерева наслідування DIT (Depth of Inheritance Tree) визначається як максимальна довжина шляху від листа до кореня дерева успадкування класів.

Для окремого класу DIT - це довжина максимального шляху від даного ласу до кореневого класу в ієрархії класів. Висока ієрархія класів (велике значення DIT) призводить до більшої складності проекту, оскільки означає залучення більшої кількості методів і класів. Разом з тим, велике значення DIT вказує, що багато методів можуть використовуватися багаторазово.

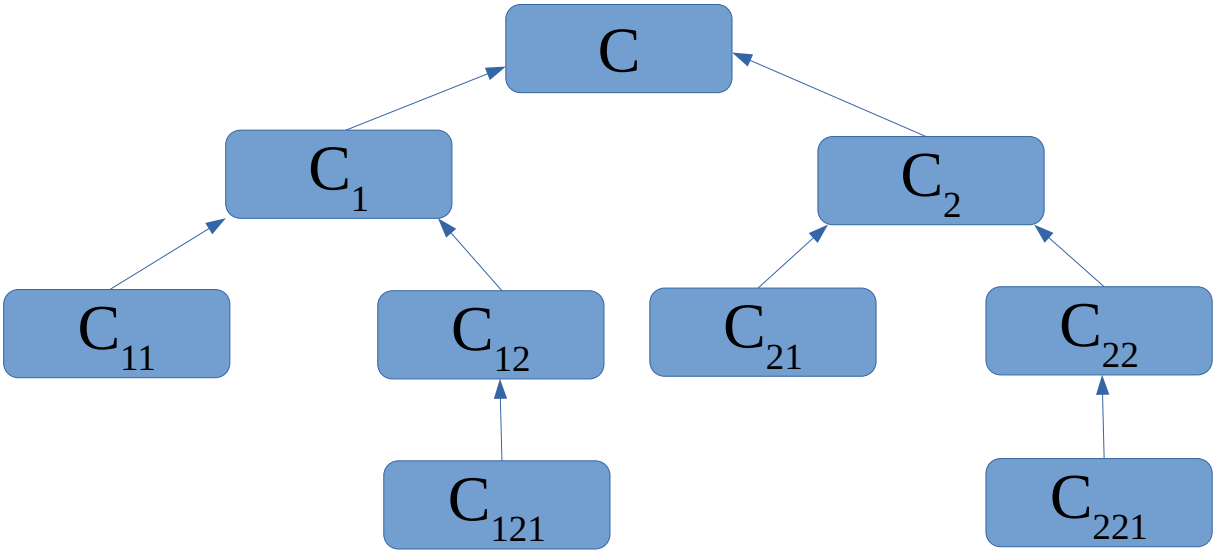

Рис. 1. Глибина дерева наслідування класів

Метрика 3: Кількість дітей NOC (Number of children) - кількість безпосередніх спадкоємців класу в ієрархії класів.

Із збільшенням NOC зростає багаторазовість використання, однак послаблюється абстракція батьківського класу. Крім того, кількість дітей характеризує потенційний вплив класу на проект. При зростанні NOC зростає кількість тестів, необхідних для перевірки кожного дочірнього класу.

Метрики DIT і NOC є кількісними характеристиками форми і розміру структури класів. Добре структурована об'єктно-орієнтована система частіше буває організована як ліс класів, ніж як надвисоке дерево. На думку Г.Буча, слід будувати збалансовані по висоті і ширині структури успадкування: зазвичай не вище, ніж 7 ± 2 рівня, і не ширше, ніж 7 + 2 гілки.

Метрика 4: Зчеплення між класами об’єктів CBO (Coupling between object classes)—це кількість взаємодій, передбачених для класу, тобто кількість класів, з якими він з'єднаний. З'єднання означає, що методи даного класу

використовують методи або поля іншого класу.

Інше визначення метрики має наступний вигляд: CBO дорівнює кількості зчеплень класу; зчеплення утворює виклик методу або властивості в іншому класі. З ростом CBO багаторазовість використання класу, ймовірно, зменшується. Очевидно, що чим більша незалежність класу, тим легше його повторно використовувати в інших програмах. Високе значення CBO ускладнює модифікацію і тестування, яке йде після модифікації. Чим більше кількість зчеплень, тим вище чутливість всього проекту до змін в окремих його частинах. Мінімізація кількості зчеплень між об'єктами покращує модульність і сприяє інкапсуляції проекту. CBO для кожного класу повинно мати розумно низьке значення. Це узгоджується з рекомендаціями щодо зменшення зчеплення стандартного програмного забезпечення.

Метрика 5: Відповідь для класу RFC (Response For a Class)

Множина відповідей класу RS — це множина методів, які можуть виконуватись у відповідь на надходження повідомлень в об'єкт цього класу:

$$RS = \{M\} U_{all\_i} \{R_i\}$$

де {Ri} — множина методів, що викликаються методом і, {M} — множина всіх методів у класі. Метрика RFC дорівнює кількості методів у множині відповіді, тобто дорівнює потужності цієї множини:

$$RFC = |RS|.$$

Інше визначення метрики: RFC — це кількість методів класу плюс кількість методів інших класів, що викликаються із даного класу. Метрика RFC є мірою потенційної взаємодії даного класу з іншими класами, дозволяє судити про динаміку поведінки відповідного об'єкта в системі. Дана метрика характеризує динамічну складову зовнішніх зв'язків класів. Якщо у відповідь на повідомлення може бути викликано велику кількість методів, то ускладнюються тестування і налагодження класу, оскільки від розробника тестів потрібен більший рівень розуміння класу, зростає довжина тестової послідовності. З ростом RFC збільшується складність класу. Найбільша величина відповіді може використовуватися при визначенні часу тестування.

Метрика 6: Відсутність зв’язності в методах LCOM (Lack of Cohesion in Methods)

Кожен метод всередині класу звертається до одного або декількох властивостей (змінні екземпляру). Метрика LCOM показує, наскільки методи не пов'язані один з одним через властивості (змінні). Якщо усі методи звертаються до однакових властивостей, то LCOM = 0.

Використовуються наступні параметри:

- *NC* - кількість пар методів без загальних змінних екземпляру;
- *C* - кількість пар методів із загальними змінними екземпляру;
- I_j - набір змінних екземпляру, що використовуються методом M_j.

$$NC = \|\{I_{ij} | I_i \cap I_j = 0\}\|$$

$$C = \|\{I_{ij} | I_i \cap I_j \neq 0\}\|$$

$$LCOM = max(0, NC - C)$$

LCOM — це кількість пар методів, не пов'язаних з властивостями класу, мінус кількість пар методів, що мають такий зв'язок. Зв’язність методів всередині класу повинна бути високою, оскільки це сприяє інкапсуляції. Якщо LCOM має високе значення, то методи слабо пов'язані один з одним через властивості. Це збільшує складність і приводить до зростанню ймовірність помилок при проектуванні. Високі значення LCOM означають, що клас, ймовірно, треба спроектувати краще (розбиттям на два або більше окремих класів). Обчислення LCOM допомагає визначити недоліки у проектуванні класів, оскільки ця метрика характеризує якість інкапсуляції даних і методів в оболонці класу. Таким чином, зв'язність у класі бажано зберігати високу, тобто слід прагнути найнижчого значення LCOM.

MOOD - Metrics for Object Oriented Design
Основними цілями MOOD-набору є:

- покриття базових механізмів об'єктно-орієнтованої парадигми, таких як інкапсуляція, успадкування, поліморфізм, посилка повідомлень;
- формальне визначення метрик, що дозволяє уникнути суб'єктивності вимірювання;
- незалежність від розміру оцінюваного програмного продукту;
- незалежність від мови програмування, на якому написаний оцінюваний продукт.

1. Фактор закритості методу (Method Hiding Factor, MHF):

$$MHF = \sum_{i=1}^{TC} Mh_i \Big/ \sum_{i=1}^{TC} (Mv_i + Mh_i)$$

де Mv_i – кількість видимих методів класу,
Mh_i – кількість прихованих методів класу,
TC — кількість класів.

Із збільшенням MHF зменшуються щільність дефектів у системі і витрати на їх усунення. Зазвичай розробка класу представляє собою покроковий процес,

при якому до класу додається все більше і більше деталей (прихованих методів). Така схема розробки сприяє зростанню як значення MHF, так і якості класу.

2. Фактор закритості атрибуту (Attribute Hiding Factor, AHF)

Фактор закритості атрибута AHF обчислюватися за формулою:

$$AHF = \frac{\sum_{i=1}^{TC} A_h(C_i)}{\sum_{i=1}^{TC} A_d(C_i)}$$

де $A_h(C_i)$ — кількість прихованих атрибутів класу C_i (інтерфейс класу),

$A_d(C_i)$ – загальна кількість атрибутів, визначених у класі C_i (без врахування наслідування),

TC – загальна кількість класів.

В ідеальному випадку всі атрибути повинні бути доступні тільки для методів відповідного класу (AHF= 100%). Тому, чим менше за 100% AHF, тим більше це вказує на необхідність додаткової уваги до відповідного класу та можливого його доопрацювання.

3. Фактор наслідування методу (Method Inheritance Factor, MIF)

Фактор наслідування методу обчислюється за формулою:

$$MIF = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)}$$

де $M_i(C_i)$ - кількість наслідуваних та не перевизначених методів класу C_i, $M_a(C_i)$ - кількість всіх методів, доступних у класі C_i.

Якщо значення MIF = 0, то це вказує, що в системі відсутнє ефективне наслідування. Наприклад, усе наслідувані методи перевизначено.

Із збільшенням MIF зменшується щільність дефектів і витрати на виправлення помилок. Теоретично, дуже великі значення MIF(70-80%) вказують на можливість зворотнього ефекту, але наявного підтвердження цього факту і його чіткої експериментальної перевірки немає.

4. Фактор наслідування властивості (Attribute Inheritance Factor, AIF)
Цей фактор обчислюється за наступною формулою:

$$AIF = \frac{\sum_{i=1}^{TC} A_i(C_i)}{\sum_{i=1}^{TC} A_a(C_i)}$$

де $A_i(C_i)$ - кількість наслідуваних та не перевизначених атрибутів класу C_i;
$A_a(C_i)$ – загальна кількість атрибутів, визначених у класі C_i,
TC – загальна кількість класів.

5. Фактор поліморфізму (Polymorphism Object Factor, POF)
Фактор поліморфізму POF дорівнює:

$$POF = \frac{\sum_{i=1}^{TC} M_o(C_i)}{\sum_{i=1}^{TC} M_n(C_i) * DC(C_i)}$$

де $M_o(C_i)$ - кількість наслідуваних та перевизначених методів класу C_i;
$M_n(C_i)$ - кількість нових методів, доступних у класі C_i;
DC - кількість класів, що наслідують клас C_i.

Помірне використання поліморфізму зменшує як щільність дефектів, так і витрати на доопрацювання. Однак при POF > 10% можливий зворотний ефект.

6. Фактор зчеплення (Coupling Factor, COF)

Зчеплення фіксує наявність між класами відносини клієнт-постачальник (client-supplier), тобто клас-клієнт містить щонайменше одне не наслідуване посилання на атрибут або метод класу-постачальника. Наявність відношення визначається за формулою :

$$is\_client(C_c, C_s)\begin{cases} 1 & if \quad Cc \Rightarrow Cs \cap Cc \neq Cs \\ 0 & else \end{cases}$$

Тоді, якщо позначити загальну кількість класів через TC, то

$$COF = \frac{\sum_{i=1}^{rc}\left[\sum_{j=1}^{rc} is\_client(C_i, C_j)\right]}{TC^2 - TC}$$

Зчеплення негативно впливають на якість ПЗ - збільшує складність, зменшує інкапсуляцію і можливості повторного використання.

Кожна з наведених метрик відноситься до основного механізму об'єктно-орієнтованої парадигми: інкапсуляції (MHF та AHF), спадкування (MIF та AIF), поліморфізму (POF) і отриманих повідомлень (COF). У визначеннях MOOD не використовуються специфічні конструкції мов програмування.

Завдання

1. Вивчіть теоретичні матеріал лабораторної роботи.
2. Для обраного проекту зробіть обчислення значень наведених вище об'єктно-орієнтованих метрик для окремих класів та в цілому.
3. Підготуйте звіт, в якому наведіть результати та пояснення до них.

Рекомендована література

1. Chidamber, S. R. A Metrics Suite for Object Oriented Design [Text] / S.R.Chidamber, C.F.Kemerer // IEEE Transactions on Software Engineering.— June 1994. —vol. 20, No. 6. —pp. 476-493.
2. Graham, I. Object-Oriented Methods. Principles & Practice [Text] / I.Graham. 3rdEdition. —Addison-Wesley, 2000. —864 pp. —ISBN 978-0201619133.
3. Hitz, M. Measuring Coupling in Object-Oriented Systems. [Text] / M.Hitz, B.Montazeri // Object Currents. —Apr 1996. —vol. 2, No. 4. —17 pp.
4. Lorenz, M. Object-Oriented Software Metrics [Text] / M.Lorenz, J.Kidd. — Prentice Hall, 1994. —146p. —ISBN 978-0131792920.

Контрольні запитання

1. Чи можна перевизначити атрибут класу і в якій мові програмування?
2. Яким чином рахувати фактор поліморфізму, якщо кількість нащадків дорівнює нулю?
3. Про що свідчить високе число DIT?
4. Про що свідчить низьке число метрики MIF?
5. Про що свідчить низьке значення фактору закритості атрибуту і як його підвищити?