ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи

на ступінь вищої освіти бакалавр

на тему**: «РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ДОСЛІДЖЕННЯ МЕТРИК ВЕРСІЙ ПРИКЛАДНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА МОВІ JAVA»**

Виконала: студентка 5 курсу, групи ППЗ-52

Спеціальності

121 Інженерія програмного забезпечення

(шрифт і назва спеціальності)

Курінна Ю.О.

(прізвище та ініціали)

Керівник Гаманюк І.М.

(прізвище та ініціали)

Рецензент_____

(прізвище та ініціали)

Київ - 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ

ТЕЛЕКОМУНІКАЦІЙ

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 Інженерія програмного забезпечення

(шифр і назва)

Завідувач кафедри

Інженерія програмного забезпечення

_ _____ Негоденко О.В.

“____” _____2021 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

Курінній Юлії Олегівні

1. Тема роботи: Розробка програмного забезпечення для дослідження метрик версій прикладного програмного забезпечення на мові JAVA.
2. Керівник роботи: Гаманюк Ігор Михалович старший викладач кафедри ІПЗ
3. Затверджені наказом вищого навчального закладу від 16.03.2021 року №65
4. Строк подання студентом роботи 01 . 06 . 2021 року
5. Вихідні дані до роботи: розробити програмне забезпечення для дослідження метрик
6. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
    1. Методи дослідження метрик;
    2. Засіб дослідження метрик;
    3. Необхідність дослідження метрик;
    4. Аналіз метрик версійності ПЗ;
7. Графічна частина роботи представлена на 30,34,55,57,58,59,60 сторінках диплому.
8. Дата видачі завдання 19.04.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз та підбір літератури по обраній тематиці бакалаврської роботи, формування завдання	19.04.2021	Викон.
2	Аналіз статичного аналізу з допомогую готових метрик порівняння	24.04.2021	Викон.
3	Дослідження методів статистичного аналізу	08.05.2021	Викон.
4	Аналіз та підбір метрик для аналізу пз на java	17.05.2021	Викон.
5	Аналіз отриманих результатів дипломної роботи. Підсумки роботи	20.05.2021	Викон.
6	Розробка доповіді і презентації	01.06.2021	Викон.

Студентка _____ Курінна Ю.О.

(підпис) (прізвище та ініціали)

Керівник роботи _____ Гаманюк І.М.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи: «Засіб дослідження метрик версій прикладного программного забезпечення на мові Java»: 55 с., 12 рис., 5 табл., 21 літературне джерело.

МЕТРИКА, ВЕРСІЙНІСТЬ, АНАЛІЗ, СКЛАДНІСТЬ, СУПРОВІД.

Об’єкт дослідження – зміна значень метрик програмного забезпечення залежно від версії.

Мета роботи – проаналізувати зміну метрик в процесі еволюції програмного забезпечення, визначити необходність внесення змін до цього процес з метою оптимізації проектів.

Методи дослідження – пошук проектів для дослідження, вибір метрик для дослідження, вимірювання обраних метрик для знайдених проектів, створення прикладного програмного забезпечення для аналізу значень метрик версій, аналіз метрик версій розробленим програмним забезпеченням, інтерпретація результатів аналізу.

Встановлено – що у великих проектах (більше 1000 класів), незалежно від умов розробки, нарощується функціональність, про що свідчить зростання метрик розміру та складності, в деяких проектах проводиться реструктуризація та виправляються значні помилки проектування, щоб подальша розробка стала можливою. Зростає складність супроводження.

Прогнозні припущення щодо розвитку об'єкта дослідження – зміна значень метрик програмного забезпечення залежно від версії дає змогу прогнозувати зміни в складності проектів, затратах на розробку та супровід, визначити чи буде ефективним подальший розвиток проекту.

ЗМІСТ

ВСТУП

На відміну від матеріальних виробничих галузей, прості підходи не можуть бути прийняті на підставі середньої продуктивності праці в проектах. Це, перш за все, вважається великою помилкою, оскільки економічні показники проекту не лінійно залежать від обсягу робіт та великої кількості трудової інтенсивності.

Тому для вирішення цього завдання використовуються комплексні і досить складні методики, які вимагають високої відповідальності в застосуванні і певного часу на адаптацію (калібрування коефіцієнтів).

Сучасні комплексні системи оцінки характеристик проектів створення ПЗ можуть бути використані для вирішення наступних завдань:

- попередня, постійна і підсумкова оцінка економічних параметрів проекту: трудомісткість, тривалість, вартість;
- оцінка ризиків по проекту: ризик порушення строків та невиконання проекту, ризик збільшення трудомісткості на етапах налагодження й супроводу проекту тощо;
- прийняття оперативних управлінських рішень - на основі відстеження певних метрик проекту можна своєчасно попередити виникнення небажаних ситуацій та усунути наслідки непродуманих проектних рішень.

У загальному випадку застосування метрик дозволяє керівникам проектів і підприємств вивчити складність розробленого або навіть розроблюваного проекту, оцінити обсяг робіт, стилістику розроблюваної програми і зусилля, витрачені кожним розробником для реалізації того чи іншого рішення. Однак метрики можуть служити лише рекомендаційними характеристиками, ними не можна повністю керуватися, так як при розробці ПО програмісти, прагнучи мінімізувати або максимізувати ту чи іншу міру для своєї програми, можуть вдаватися до хитрощів аж до зниження

ефективності роботи програми. Крім того, якщо, наприклад, програміст написав мала кількість рядків коду або вніс невелике число структурних змін, це зовсім не означає, що він нічого не робив, а може означати, що дефект програми було дуже складно відшукати. Остання проблема, однак, частково може бути вирішена при використанні метрик складності, тому що в більш складною програмою помилку знайти складніше.

Метою дипломної роботі є аналіз зміну метрик в процесі еволюції програмного забезпечення, візначіті необходність внесення змін до цього процес З метою оптімізації проектів.

Для досягнення поставленої мети необхідно віконаті наступні завдання:

Знайте три проекти напісані намові Java з відкрітім кодом. Об'єм проектів не менше 1000 класів. Шкірної проект має буті доступною не менше Як у десяти версіях.

- Зверни метрики для Дослідження версійності
- Зверни засіб вімірювання метрик
- Віміряті обрані метрики для кожної з десяти версій знайденіх проектів
- Розробіті прикладні програмне забезпечення для статистичного аналізу метрик та побудова графіків метрик в за лежності від версій
- Проаналізуваті метрики знайденіх проектів
- Інтерпретуваті результати аналізу

Об'єкт дослідження - метрики програмного забезпечення.

Предмет дослідження - зміна значень метрик програмного забезпечення перелогових від версії, дослідження складності та якості програмного забезпечення.

1. МЕТРИКИ ВЕРСІЙ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Версійність програмного забезпечення.

Версійність програмного забезпечення (Software versioning) - це процес присвоєння унікальних версійністних імен або версійністних нумерацій до програмного продукту.

Існує безліч схем і категорій (major, minor) версійність програмного забезпечення. Як версії продукту може виступати дата (Date, Year of release) в тому чи іншому форматі, наприклад YYYY-MM-D, різні буквено-цифрові (Alphanumeric codes) позначення або кодові назви.

Однак найбільш поширеною схемою є числова послідовність (Sequence-based identifiers), що присвоюється тієї чи іншої версії продукту. На рисунке 1 показан зразок числової схеми. При використанні цього підходу кожному випуску присвоюється унікальний ідентифікатор, який складається з кількох цифрових послідовностей. Як правило цифрове значення версії збільшується відповідно до модернізаціями удосконаленням програмного продукту.

Версія в цьому випадку задається значенням однієї з цифр, кожна з яких відповідає за різні характеристики продукту, наприклад:

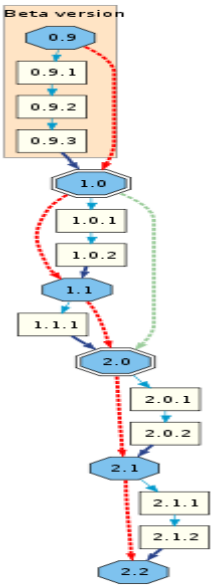


Рис 1. Цифрове подання версій програмного продукту***major.minor [. build [. revision]]***

або

major.minor [. maintenance [. build]]

Значення major, як правило, змінюється при значних змінах у функціональності продукту, minor збільшується коли зміна менш значимо.

У будь-якому випадку, слід пам'ятати про те, що версія задається людиною, а не машиною і не існує жодних суворих правил і процедур для ведення версійність продукту. Якщо розробник хоче подчекнуть значні зміни у функціональності продукту, немає ніяких заборон на те, щоб він збільшив значення major.

1.2. Метрики у версіях прикладного програмного забезпечення на Java.

Метрика програмного забезпечення - це міра, що відображає чисельне значення деякої властивості програмного забезпечення або його специфікацій.

Оскільки кількісні методи добре зарекомендували себе в інших областях, багато теоретиків і практики інформатики намагалися перенести даний підхід і в розробку програмного забезпечення. Як сказав Том Демарко, «ви не можете контролювати те, що не можете виміряти».

При вимірювання метрик версій програмних продуктів підходить велика частина існуючих метрик. Для наочного порівняння різних версій, зміни їх властивостей і витрачаються, в основному розглядаються метрики, що вимірюють кількісні значення. Але в силу таких причин, як зміна алгоритму або методу роботи програми \ підпрограми кількісні значення не зможуть показати зміни проекту в повній мірі.

Мінімальний набір використовуваних метрик версійності включає:

• порядок зростання (аналіз властивостей і закономірностей алгоритмів),

• кількість рядків коду (сумарна кількість рядків вихідного коду),

• Цикломатичне складність (метрика визначальна складність програм),

• кількість помилок на 1000 рядків коду,

• ступінь покриття коду тестуванням (метрика, що показує який відсоток вихідного коду був протестований),

• покриття вимог (метрика, яка визначає рівень проведених тестів по відношенню до функціональних вимог системи),

• кількість класів і інтерфейсів,

• зв'язність (метрика, определяющая уровень взаимосвязанности одних модулей с другими).

Для порівняння значень метрик різних версій програмних проектів на мові Java найкраще проводити аналіз таких проектів як argoUML, ісхожний код яких включає до 1000 класів. Open Source проект подібного масштабу дозволить отримати більшу колчіество значень метрик, що вплине на точність загальних результатів порівнянь різних програмних продуктів.

1.3 Необхідність дослідження метрик версій програмного забезпечення.

У версійність програмного забезпечення використовується безліч метрик, що дозволяють визначити вимірювані параметри коду, в процесі його еволюціонування.

Кількість рядків коду (Source Lines of Code - SLOC) - це метрика програмного забезпечення, що використовується для вимірювання його обсягу з допомогою підрахунку кількості рядків у тексті вихідного коду. Як правило, цей показник використовується для прогнозу трудовитрат на розробку конкретної програми, або для оцінки продуктивності праці.

Цикломатичне складність програми (Cyclomatic complexity of a program) - структурна міра складності програм, яка використовується для вимірювання якості програмного забезпечення, заснована на методах статичного аналізу коду. Цикломатичне складність дорівнює збільшеному на одиницю Цикломатичне число графа програми.

Покриття коду (Code coverage) - міра, яка використовується для тестування програмного забезпечення. Вона показує відсоток, наскільки вихідний код програми був протестований. Техніка покриття коду в основному використовується для систематичного тестування ПЗ.

Покриття вимог дозволяє оцінити ступінь повноти системи тестів по відношенню до функціональності системи. У порівнянні з покриттям коду, покриття вимог дозволяє виявити нереалізовані вимоги, але не дозволяє оцінити повноту стосовно її програмної реалізації. Одна і та ж функція може бути реалізована за допомогою зовсім різних алгоритмів, що вимагають різного підходу до організації тестування.

Статистичні методи аналізу, в версійність, мають характерні їм недоліки - вони дозволяють вирішувати і передбачати можливі проблеми лише для типових завдань, так як повністю спирається лише на статистику. Даний метод не підходить для вирішення унікальних завдань.

2. МЕТОД ДОСЛІДЖЕННЯ МЕТРИК ВЕРСІЙ ПРИКЛАДНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Вибір та вимірювання метрик версій прикладного програмного забезпечення.

Метрики складності програм прийнято розділяти на три основні групи:

• метрики розміру програм;

• метрики складності потоку керування програм;

• метрики складності потоку даних програм.

Метрики першої групи базуються на визначенні кількісних характеристик, пов'язаних з розміром програми, і відрізняються відносною простотою. До найбільш відомих метрика даної групи відносяться число операторів програми, кількість рядків вихідного тексту, набір метрик Холстеда. Метрики цієї групи орієнтовані на аналіз вихідного тексту програм. Тому вони можуть використовуватися для оцінки складності проміжних продуктів розробки.

Метрики другої групи базуються на аналізі керуючого графа програми. Представником цієї групи є метрика МакКейб.

Керуючий граф програми, який використовують метрики даної групи, може бути побудований на основі алгоритмів модулів. Тому метрики другої групи можуть застосовуватися для оцінки складності проміжних продуктів розробки, починаючи з роботи 6.

Метрики третьої групи базуються на оцінці використання, конфігурації та розміщення даних у програмі. У першу чергу це стосується глобальних змінних. До даної групи відносяться метрики Чепіно.

2.1.1. Розмірно-орієнтовані метрики (показники оцінки обєму)

LOC- оцінка (Lines Of Code)

Розмірно-орієнтовані метрики прямо вимірюють програмний продукт і процес його розробки. Грунтуються такі метрики на LOC-оцінках.

Цей вид метрик побічно вимірює програмний продукт і процес його розробки. Замість підрахунку LOC-оцінок при цьому розглядається не розмір, а функціональність або корисність продукту.

Найбільшого поширення в практиці створення програмного забезпечення отримали розмірно-орієнтовані метрики. В організаціях, зайнятих розробкою програмної продукції для кожного проекту прийнято реєструвати наступні показники:

• загальні трудовитрати (в людино-місяцях, людино-годинах);

• обсяг програми (в тисячах рядках вихідного коду-LOC);

• вартість розробки;

• обсяг документації;

• помилки, виявлені протягом року експлуатації;

• кількість людей, які працювали над виробом;

• термін розробки.

На основі цих даних зазвичай підраховуються прості метрики для оцінки продуктивності праці (KLOC / людино-місяць) і якості виробу.

Ці метрики не універсальні та спірні, особливо це стосується такого показника як LOC, який істотно залежить від мови програмування.

Кількість рядків вихідного коду (Lines of Code - LOC, Source Lines of Code - SLOC) є найбільш простим і поширеним способом оцінки обсягу робіт за проектом.

Спочатку даний показник виник як спосіб оцінки обсягу роботи за проектом, в якому застосовувалися мови програмування, які мають досить простою структурою: «один рядок коду = одна команда мови». Також давно відомо, що одну й ту ж функціональність можна написати різною кількістю рядків, а якщо візьмемо мова високого рівня (C + +, Java), то можливо і в одному рядку написати функціонал 5-6 рядків - це не проблема. І це було б півбіди: сучасні засоби програмування самі генерують тисячі строк коду на дріб'язкову операцію.

Тому метод LOC є тільки оцінними методом (який треба брати до відома, але не спиратися в оцінках) і ніяк не обов'язковим.

Залежно від того, яким чином враховується подібний код, виділяють два основні показники SLOC:

a. кількість «фізичних» рядків коду - SLOC (використовувані абревіатури LOC, SLOC, KLOC, KSLOC, DSLOC) - визначається як загальна кількість рядків вихідного коду, включаючи коментарі і порожні рядки (при вимірюванні показника на кількість порожніх рядків, як правило, вводиться обмеження - при підрахунку враховується кількість порожніх рядків, що не перевищує 25% загального числа рядків у вимірюваному блоці коду).

b. Кількість «логічних» рядків коду - SLOC (використовувані абревіатури LSI, DSI, KDSI, де «SI» - source instructions) - визначається як кількість команд і залежить від мови програмування. У тому випадку, якщо мова не допускає розміщення кількох команд на одному рядку, то кількість «логічних» SLOC буде відповідати числу «фізичних», за винятком числа порожніх рядків і рядків коментарів. У тому випадку, якщо мова програмування підтримує розміщення декількох команд на одному рядку, то одна фізична рядок має бути врахована як декілька логічних, якщо вона містить більше однієї команди мови.

Для метрики SLOC існує велика кількість похідних, покликаних отримати окремі показники проекту, основними серед яких є:

• кількість порожніх рядків;

• кількість рядків, що містять коментарі;

• відсоток коментарів (відношення рядків коду до рядків коментарю, похідна метрика стилістики);

• середня кількість рядків для функцій (класів, файлів);

• середня кількість рядків, що містять вихідний код для функцій (класів, файлів);

• середня кількість рядків для модулів.

Метрика стилістики та зрозумілості програм

Іноді важливо не просто порахувати кількість рядків коментарів в коді і просто співвіднести з логічними рядками коду, а дізнатися щільність коментарів. Тобто код спочатку був документований добре, потім - погано. Або такий варіант: шапка функції або класу документована і коментувати, а код немає.

Fi = SIGN (Nкомм. i / Ni – 0,1)

Загальна оцінка F = ΣFi

Суть метрики проста: код розбивається на n-рівні шматки і для кожного з них визначається Fi

Разом по SLOC

Потенційні недоліки SLOC, на які націлена критика:

• некрасиво і неправильно зводити оцінку роботи людини до кількох числовим параметрами і по них судити про продуктивність. Менеджер може призначити найбільш талановитих програмістів на складний ділянку роботи; це означає, що розробка цієї ділянки забере найбільший час й породить найбільшу кількість помилок, через складність завдання. Не знаючи про ці труднощі, інший менеджер за отриманими показниками може вирішити, що програміст зробив свою роботу погано.

• Метрика не враховує досвід співробітників та їх інші якості

• Спотворення: процес вимірювання може бути спотворений за рахунок того, що співробітники знають про вимірюваних показниках і прагнуть оптимізувати ці показники, а не свою роботу. Наприклад, якщо кількість рядків вихідного коду є важливим показником, то програмісти будуть прагнути писати якомога більше рядків і не будуть використовувати способи спрощення коду, що скорочують кількість рядків (див. врізку про Індію).

• Неточність: ні метрик, які були б одночасно і значущими є і досить точні. Кількість рядків коду - це просто кількість рядків, цей показник не дає уявлення про складність розв'язуваної проблеми. Аналіз функціональних точок був розроблений з метою кращого вимірювання складності коду і специфікації, але він використовує особисті оцінки вимірює, тому різні люди отримають різні результати.

• Метрика SLOC не відображає трудомісткості зі створення програми.

2.1.2. Метрики складності

Крім показників оцінки обсягу робіт за проектом дуже важливими для одержання об'єктивних оцінок по проекту є показники оцінки його складності. Як правило, дані показники не можуть бути обчислені на самих ранніх стадіях роботи над проектом, оскільки вимагають, як мінімум, детального проектування. Однак ці показники дуже важливі для отримання прогнозних оцінок тривалості і вартості проекту, оскільки безпосередньо визначають його трудомісткість.

Об'єктно-орієнтовані метрики

У сучасних умовах більшість програмних проектів створюється на основі підходу, у зв'язку з чим існує значна кількість метрик, що дозволяють отримати оцінку складності об'єктно-орієнтованих проектів (див. табл. 1).

Табл. 1. Метрики складності об'єктно-орієнтованих проектів

Метрика	Опис
Зважена насиченість класу 1 (Weighted Methods Per Class (WMC)	Відображає відносну міру складності класу на основі Цикломатичне складності кожного його методу. Клас з більш складними методами і великою кількістю методів вважається більш складним. При обчисленні метрики батьківські класи не враховуються.

Зважена насиченість класу 2 (Weighted Methods Per Class (WMC2))	Міра складності класу, заснована на тому, що клас з великим числом методів, є більш складним, і що метод з великою кількістю параметрів також є більш складним. При обчисленні метрики батьківські класи не враховуються.
Глибина дерева спадкування (Depth of inheritance tree)	Довжина найдовшого шляху успадкування, що закінчується на даному модулі. Чим глибше дерево успадкування модуля, тим може виявитися складніше передбачити його поведінку. З іншого боку, збільшення глибини дає більший потенціал повторного використання даним модулем поведінки, визначеного для класів-предків.
Кількість дітей (Number of children)	Число модулів, безпосередньо успадковують даний модуль.Большіе значення цієї метрики вказують на широкі можливості повторного використання; при цьому занадто велике значення може свідчити про погано вибраної абстракції.
Зв'язність об'єктів (Coupling between objects)	Кількість модулів, пов'язаних з даним модулем в ролі клієнта або постачальника. Надмірна зв'язність говорить про слабкість модульної інкапсуляції і може перешкоджати повторному використанню коду.
Відгук на клас (Response For Class)	Кількість методів, які можуть викликатися екземплярами класу; обчислюється як сума кількості локальних методів, так і кількості віддалених методів

Метрики Холстеда

Метрика Холстеда відноситься до метрика, обчислюваним на підставі аналізу числа рядків і синтаксичних елементів вихідного коду програми.

Основу метрики Холстеда складають чотири вимірювані характеристики програми:

- NUOprtr (Number of Unique Operators) - число унікальних операторів програми, включаючи символи-роздільники, імена процедур і знаки операцій (словник операторів);
- NUOprnd (Number of Unique Operands) - число унікальних операндів програми (словник операндів);
- Noprtr (Number of Operators) - загальна кількість операторів у програмі;
- Noprnd (Number of Operands) - загальна кількість операндів у програмі.

На підставі цих характеристик розраховуються оцінки:

- Словник програми (Halstead Program Vocabulary, HPVoc): HPVoc = NUOprtr + NUOprnd;
- Довжина програми (Halstead Program Length, HPLen): HPLen = Noprtr + Noprnd;
- Обсяг програми (Halstead Program Volume, HPVol): HPVol = HPLen log2 HPVoc;
- Складність програми (Halstead Difficulty, HDiff): HDiff = (NUOprtr / 2) × (NOprnd / NUOprnd);
- На основі показника HDiff пропонується оцінювати зусилля програміста при розробці за допомогою показника HEff (Halstead Effort): HEff = HDiff × HPVol.

Метрики цикломатичної складності за Мак-Кейбу

Показник Цикломатичне складності є одним з найбільш поширених показників оцінки складності програмних проектів. Даний показник був розроблений вченим Мак-Кейбом в 1976 р., відноситься до групи показників

оцінки складності потоку керування програмою і обчислюється на основі графа керуючої логіки програми (control flow graph). Даний граф будується у вигляді орієнтованого графа, в якому обчислювальні оператори або вирази представляються у вигляді вузлів, а передача управління між вузлами - у вигляді дуг.

Показник Цикломатичне складності дозволяє не тільки зробити оцінку трудомісткості реалізації окремих елементів програмного проекту і скоригувати загальні показники оцінки тривалості та вартості проекту, але й оцінити пов'язані ризики і прийняти необхідні управлінські рішення.

Спрощена формула обчислення Цикломатичне складності розподілена так:

$$C = e - n + 2$$

де e - число ребер, а n - число вузлів на графі керуючої логіки.

Як правило, при обчисленні Цикломатичне складності логічні оператори не враховуються.

У процесі автоматизованого обчислення показника Цикломатичне складності, як правило, застосовується спрощений підхід, відповідно до якого побудова графа не здійснюється, а обчислення показника здійснюється на підставі підрахунку кількості операторів керуючої логіки (if, switch і т.д.) і можливої кількості шляхів виконання програми.

Цикломатичне число Мак-Кейба показує необхідну кількість проходів для покриття всіх контурів Сільносвязанная графа або кількості тестових прогонів програми, необхідних для вичерпного тестування за принципом «працює кожна гілка».

Показник Цикломатичне складності може бути розрахований для модуля, методу та інших структурних одиниць програми.

Існує значна кількість модифікацій показника Цикломатичне складності.

- «Модифікована» Цикломатичне складність - розглядає не кожне розгалуження оператора множинного вибору (switch), а весь оператор як єдине ціле.
- «Сувора» Цикломатичне складність - включає логічні оператори.
- «Спрощене» обчислення Цикломатичне складності - передбачає обчислення не на основі графа, а на основі підрахунку керуючих операторів.

Метрики Чепина

Існує кілька її модифікацій. Розглянемо більш простий, а з точки зору практичного використання - досить ефективний варіант цієї метрики.

Суть методу полягає в оцінці інформаційної міцності окремо взятого програмного модуля за допомогою аналізу характеру використання змінних зі списку введення-виведення.

Всі безліч змінних, складових список введення-виведення, розбивається на чотири функціональні групи.

1. Безліч «P» - що вводяться змінні для розрахунків та для забезпечення виводу. Прикладом може служити використовувана в програмах лексичного аналізатора змінна, що містить рядок вихідного тексту програми, тобто сама змінна не модифікується, а лише містить вихідну інформацію.

2. Безліч «M» - модифікуються або створювані всередині програми змінні.

3. Безліч «C» - змінні, що беруть участь в управлінні роботою програмного модуля (керуючі змінні).

4. Безліч «T» - не використовуються в програмі ("паразитні") змінні. Оскільки кожна змінна може виконувати одночасно декілька функцій, необхідно враховувати її в кожній відповідній функціональній групі.

Далі вводиться значення метрики Чепіно:

$$Q = a1P + a2M + a3C + a4T$$

де a1, a2, a3, a4 - вагові коефіцієнти.

Вагові коефіцієнти використані для відображення різного впливу на складність програми кожної функціональної групи. На думку автора метрики найбільшу вагу, рівний трьом, має функціональна група С, так як вона впливає на потік управління програми. Вагові коефіцієнти інших груп розподіляються наступним чином: a1 = 1; a2 = 2; a4 = 0.5. Ваговий коефіцієнт групи Т не дорівнює нулю, оскільки "паразитні" змінні не збільшують складності потоку даних програми, але іноді ускладнюють її розуміння. З урахуванням вагових коефіцієнтів вираз набуде вигляду:

$Q = P + 2M + 3C + 0.5T$

2.1.3. Попередня оцінка на основі статистичних методів в залежності від етапів розробки програми

При використанні інтегрованих інструментальних засобів у компаній, що розробляють типові рішення (під цю категорію потрапляють так звані «інхаузери» - компанії, що займаються обслуговуванням основного бізнесу) з'являється можливість будувати прогнози складності програм, грунтуючись на зібраної статистики. Статистичний метод добре підходить для вирішення подібних типових задач і практично не підходить для прогнозу унікальних проектів. У разі унікальних проектів застосовуються інші підходи, обговорення яких знаходиться за рамками даного матеріалу.

Типові завдання як з рогу достатку падають на відділи розробки з бізнесу, тому попередня оцінка складності могла б сильно спростити завдання планування і управління, тим більше що є накопичена база по проектах, в якій збережені не тільки остаточні результати, але і всі початкові і проміжні.

Виділимо типові етапи у розробці програм:

- розробка специфікації вимог до програми;
- визначення архітектури;

- опрацювання модульної структури програми, розробка інтерфейсів між модулями. Опрацювання алгоритмів;
- розробка коду і тестування.

Тепер спробуємо розглянути ряд метрик, часто використовуваних для попередньої оцінки на перших двох етапах.

Попередня оцінка складності програми на етапі розробки специфікації вимог до програми

Для оцінки за результатами роботи даного етапу може бути використана метрика прогнозованого числа операторів Nпрогн програми:

Nпрогн = NF*Nед

де:

NF - кількість функцій чи вимог у специфікації вимог до розроблюваної програми;

Nед - одиничне значення кількості операторів (середнє число операторів, що припадають на одну середню функцію або вимога). Значення Nед - статистичне.

Попередня оцінка складності на етапі визначення архітектури

Си = NI / (NF * NIед * Ксл)

де:

NI - загальна кількість змінних, переданих по інтерфейсах між компонентами програми (також є статистичної);

NIед-одиничне значення кількості змінних, переданих по інтерфейсах між компонентами (середнє число переданих по інтерфейсах змінних, що припадають на одну середню функцію або вимога);

Ксл - коефіцієнт складності програми, що розробляється, враховує зростання одиничної складності програми (складності, що припадає на одну функцію або вимога специфікації вимог до програми) для великих і складних програм в порівнянні з середнім ПС.

Підведення підсумків

Табл. 2. Склад метрик їх вплив і аналіз ефективності використання

Метрика	Навіщо потрібна	Впливає на ...	Аналіз на основі статистичних даних (як тренд, так і прогноз)
Зусилля розробника при реалізації.	Наскільки ефективний працю розробника.	Точність прогнозів оцінки трудомісткості при виконанні організацією типових або мало відрізняються запитів	Можна аналізувати зусилля розробника в часовому зрізі або в зрізі по релізів або проектам. Виявляти, на яких завданнях програміст повністю викладається, а які йому не до душі. Тренд дозволить менеджеру краще розуміти, хто і яких завданнях максимально ефективний при формуванні команди нового проекту, а також які підсистеми щодо складні, а які - прості.

Довжина і обсяг програми		Оцінку обсягу змін	Збільшується або зменшується обсяг програми в часі. Використовуємо для прогнозу складності на ранніх етапах на основі статистики.
Аналіз Цикломатичне складності.		Оцінку складності змін	Складність зростає чи ні? Використовуємо для прогнозу складності на ранніх етапах на основі статистики.
Зусилля програміста при розробці.	Для визначення складності реалізації того чи іншого блоку коду (класу, функції і т.д.)	Розуміння того, наскільки інтелектуально-затратною для розробника була та чи інша функція.	Аналізується збільшення або зменшення зусиль розробника багато часу. На попередніх етапах метрику можна використовувати для прогнозу.
Кількість рядків на реалізацію вимоги.	Міряємо загальну температуру. Ця метрика приймається	Розуміння КПД.Отслежіваем сплески.	Сигнал небезпеки при виявленні збільшення кількості рядків під час виконання

	до уваги при аналізі реалізації запиту.		типового запиту Використовуємо для оцінки складності на ранніх етапах на основі статистики.
Кількість коментарів на одиницю коду.	Код повинен бути документований. Якщо співвідношення коду до коментарі не 1:4, то розробник зобов'язаний доопрацювати.	Якість коду, його прозорість.	Загальна культура розробників росте чи ні?
Інші кількісні метрики (число функцій, класів, файлів).	Ставлення нових функцій до зміненим.	Кількість доданих, вилучених і змінених рядків по відношенню до попередньої версії.	Якщо зростає - добре.
Щільність дефектів на	Кількість дефектів на	Похідна метрика:	Якщо ні - погано.

одиницю коду.	1-у рядок коду	кількість рядків / число дефектів	

Підводячи підсумок, хотілося б відзначити, що жодної універсальної метрики не існує. Будь-які контрольовані метричні характеристики програми повинні контролюватися або в залежності один від одного, або в залежності від конкретного завдання, крім того, можна застосовувати гібридні заходи, проте вони так само залежать від простіших метрик і також не можуть бути універсальними. Строго кажучи, будь-яка метрика - це лише показник, який сильно залежить від мови і стилю програмування, тому ні одну міру не можна зводити в абсолют і ухвалювати будь-які рішення, грунтуючись тільки на ній.

Отже, обрані метрики:

NDD Number of direct descendants	Чим глибше дерево успадкування модуля, тим може виявитися складніше передбачити його поведінку. З іншого боку, збільшення глибини дає більший потенціал повторного використання даним модулем поведінки, визначеного для класів-предків.
HIT Height of inheritance tree	Великі значення цієї метрики вказують на широкі можливості повторного використання; при цьому занадто велике значення може свідчити про погано вибраної абстракції.
NOPNumber of packages NOC Number of classes	Збільшується або зменшується обсяг програми в часі. Глибокий аналіз змін по релізів (версіями, зборках) дає зрозуміти: Кількість змін (на що завгодно) - скільки разів один і той

NOM Number of methods LOC Lines of code	же блок коду коректувався. Можливо виявити вузьке місце в програмі: інтенсивно мінливий блок коду може впливати на загальну якість програми (потенційне місце виникнення помилок). Можливо, необхідно змінити архітектуру блоку.
CYCLO Cyclomatic complexity	Оцінку складності змін. Використовуємо для прогнозу складності на ранніх етапах на основі статистики.
CBO Class Coupling CALL Calls per method FOUT Fan out (number of other methods called by a given method)	Кількість модулів, пов'язаних з даним модулем в ролі клієнта або постачальника. Надмірна зв'язність говорить про слабкість модульної інкапсуляції і може перешкоджати повторному використанню коду.
Brain Class	Класи Що реалізують основну логіку програмного продукту
Brain Method	Методи Що реалізують основну логіку програмного продукту
God Class	Класи, Що містять велику кількість Даних або велику кількість методів. У великих програмних продуктах можут буті Такі класи в обмеженій кількості. В цілому Такі класи Важко підтрімуваті. Смороду порушують принцип інкапсуляції.
Data Class	Класи з данімі

Співвідношення метрик.

Цифри вказують співвідношень; кольору вказують, де відносини вписуються в стандартні діапазони (похідний від численних проектів з відкритим кодом). Кожне ставлення або зелений (в межах діапазону), синій (нижче діапазону), або червоний (поза діапазону) (Див. Табл. 3)

Табл. 3. Співвідношення метрик

	Low	Medium	High
CYC/LOC	0.16	0.20	0.24
LOC/NOM	7	10	13
NOM/NOC	4	7	10
NOC/NOP	6	17	26
CALL/NOM	2,01	2,62	3,20
FOUT/CALL	0.56	0.62	0.68

2.1.4. iPlasma

iPlasma містить більше 80 метрик.

Метрики iPlasma діляться на 4 групи:

- Метрики розміру – включають розміри об’єкту аналізу(наприклад, LOC)

- Метрики складності – включають складність об’єкту дослідження(наприклад, Cyclomatic Complexity)

- Метрики взаємозв’язку – включає обмін даними між об’єктами (наприклад, Coupling Between Objects)

- Метрики зв’язаності класів – включає зв’язаність класів між собою(наприклад, Tight Class Cohesion)

iPlasma був успішно використаний для аналізу дизайн з більш ніж десяти реальних, промислових систем, включаючи дуже великі з відкритим вихідним кодом системи (> 1 Mloc), як і Mozilla (C + +, 2560000 LOC) і Eclipse (Java, 1360000 LOC) . iPlasma був також використаний протягом кількох заходів консультації для промислових партнерів, більшість з них

беруть участь в розробці великих програм програмного забезпечення для телекомунікаційних систем.

Інструменти:

Огляд піраміди

Огляд піраміда являє собою інтегровану, метрики основі кошти як описати і охарактеризувати загальну структуру об'єктно-орієнтована система, в кількісному аспектах складності, зв'язку та використання спадкування. (Рис. 1.)

Рис. 2. Піраміда огляду

Insider

Одним з ключових аспектів iPlasma в тому, що всі аналізи інтегровані і можуть використовуватися в єдиній формі за допомогою гнучкого інтерфейсу, називається Insider. (Рис. 2.)

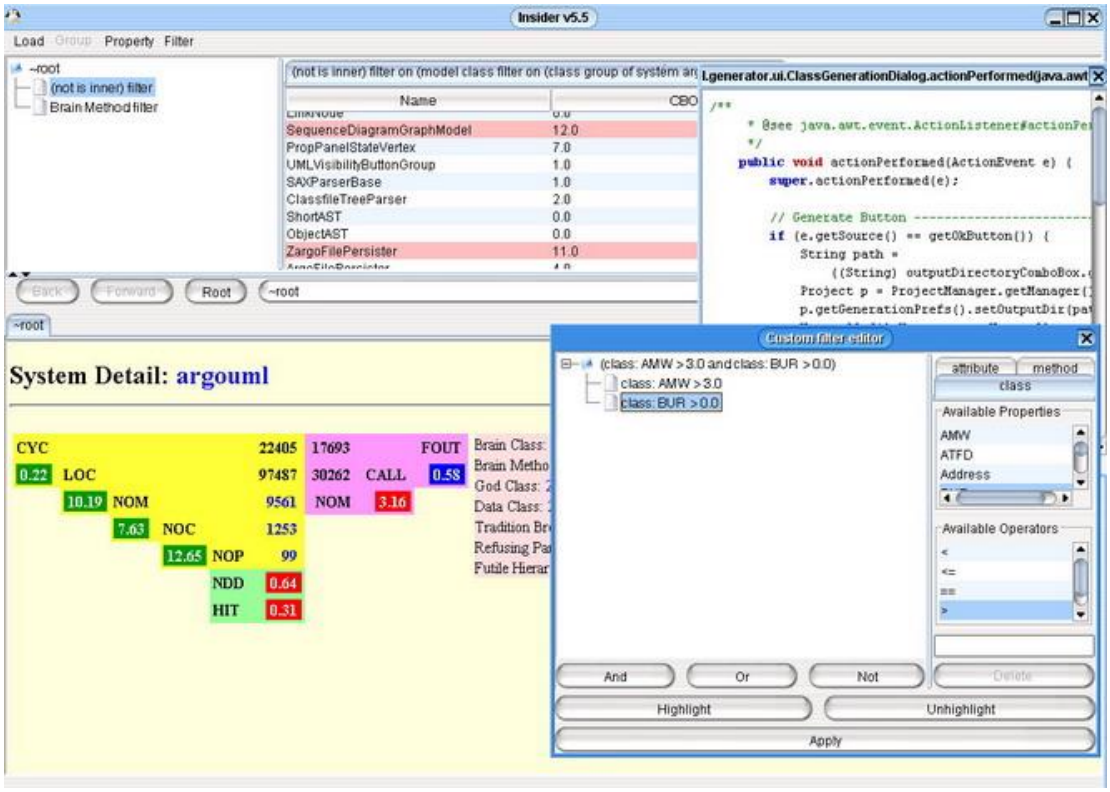


Рис. 3. Insider

2.1.5. Analyst4j

Розгляд питання про оцінку та використання Analyst4j для вашого проекту, якщо ви хочете:

- Для цього причина (Парето) аналіз і знайти 20% ваших програмних артефактів, які відповідальні за більшість ефект.

- Знайти Antipatterns елементи швидко (Blob класів, спагетті кодекс, швейцарський ніж класи ..)

- Виконання користувальницьких аналізу та інтерпретації, засновані на автоматизованих метрик програмного забезпечення.

- Розумієте поточного супроводу проекту.

- Знайти резерви для поліпшення роботи із забезпечення майбутнього ремонтопридатність.

- Прийняти та підтримувати з відкритим вихідним кодом Java-продуктів. Оцініть свої кандидати якість / ремонтопридатність використанням Analyst4j.

- Зрозуміти ефект від використання метрик програмного забезпечення, або Дізнайтеся Analyst4j можете дізнатися для вас.

- Оцініть з тестування на основі таких показників, як Цикломатичне Складність і Halstead Effort.

Метрики Analyst4j:

- об'єктно-орієнтованих метрик:

- Зважена Складність методи (WMC)

- Відповідь Для класу (RFC)

- Відсутність зчеплення методи (LCOM)

- Зчеплення між об'єктами (CBO)

- Глибина дерева спадкування (DIT)

- Кількість дітей (NOC)

- Метрики складності:

- Цикломатичне складності (Маккейб)

- Основні складності (EC)

- Холстед Метрики складності (Холстед зусилля, том)

- Ремонтопридатність Індекс Метрична

- Кодекс Метрики

Дані метрики дають Загальну картину про написань код програмного продукту, дають кращє розуміння інших метрик.

Analyst4j автоматізує наступні метрики коду через Чотири рівня (Метод, Клас, Файл и Пакет):

Метод Рівня

- Number of lines of Code (NLOC_MTD)
- Percentage of comments (POC_MTD)
- Number of Variables (NOV_MTD)
- Number of Unused Variables (NOUV_MTD)
- Number of comment lines (CL_MTD)
- Number of Parameters (NOP_MTD)

- Number of Unused Parameters (NOUP_MTD)

Клас Рівня

- Number of Lines of Code(NLOC_CLS)
- Number of Parents (NOPNT_CLS)
- Number of Fields (NOFLD_CLS)
- Percentage of Non-Private Fields (NPFP_CLS)
- Percentage of Non-Private Methods (NPMP_CLS)
- Number of Inner Classes (NOIC_CLS)

Файл Рівня

- Number of Lines of Code (NLOC_FIL)
- Halstead Effort / Volume (HE_FIL, HV_FIL)
- SEI Maintainability Index (MI_FIL)

Пакет Рівня

- Number of Lines of Code (NLOC_PKG)
- Number of Classes (NOCLS_PKG)
- Number of Interfaces(NOIFC_PKG)
- Number of Files (NOF_PKG)

Приклад роботи Analyst4j можна побачити на Рис. 3.

Тег	Назва метрики	Опис
LOC	Рядки коду	Ця метрика вказує на кількість не закоментованих рядків коду в функції (LOCf),в модулі (LOCm), або в проекті (LOCp). LOC найчастіше всього використовується для того, щоб отримати розміри програмного продукту.
MVG	Цикломатична складність за Мак Кейбом	Вимірювання проводиться на основі направленого ациклічного графа, який представляє потік контролю в межах кожної функції. Спершу пропонується проаналізувати мінімальне число тестових випадків, щоб гарантувати, що всі частини кожної функції безпомилково працюють.
COM	Коментовані рядки	Вказує на кількість закоментованих рядків коду в програмі. Найчастіше використовується окремо від інших метрик, але інколи використовується разом з метриками цикломатичної складності та загальної кількості рядків коду
L_C,M_C	LOC/COM, MVG/COM	Дивись вище
FO,FOc,FOv FI,FIc,FIc	Fan-out, Fan-in	Дані метрики вказують на: fan-out - число інших модулів, які використовують модуль А, поки fan-in - число інших модулів, які використовують модуль А.
HKS, HKSv, HKSc	Henry-Kafura/Shepperd measure	Цю метрику отримує кожний атрибут fan-in і fan-out кожного модуля.
NOM	Число модулів	Загальна кількість модулів, що спостерігаються в проекті
WMC	Зваженість методів класу	Метрика, що були запропонована Chidamber і Kemerer, - граф числа функцій, визначених в модулі, помноженому на навантажуючий чинник. Лише навантажуючий алгоритм, запропонований в

		оригінальному формулюванні, - загальноприйнята надбавка однієї одиниці за функцію.
REJ	Рядки, що ігноруються	Метрика, що вказує на не порожні рядки коду, що не закоментовані, яку не проаналізував аналізатор. Це більше перевірка достовірності даних на згенеровані повідомлення.

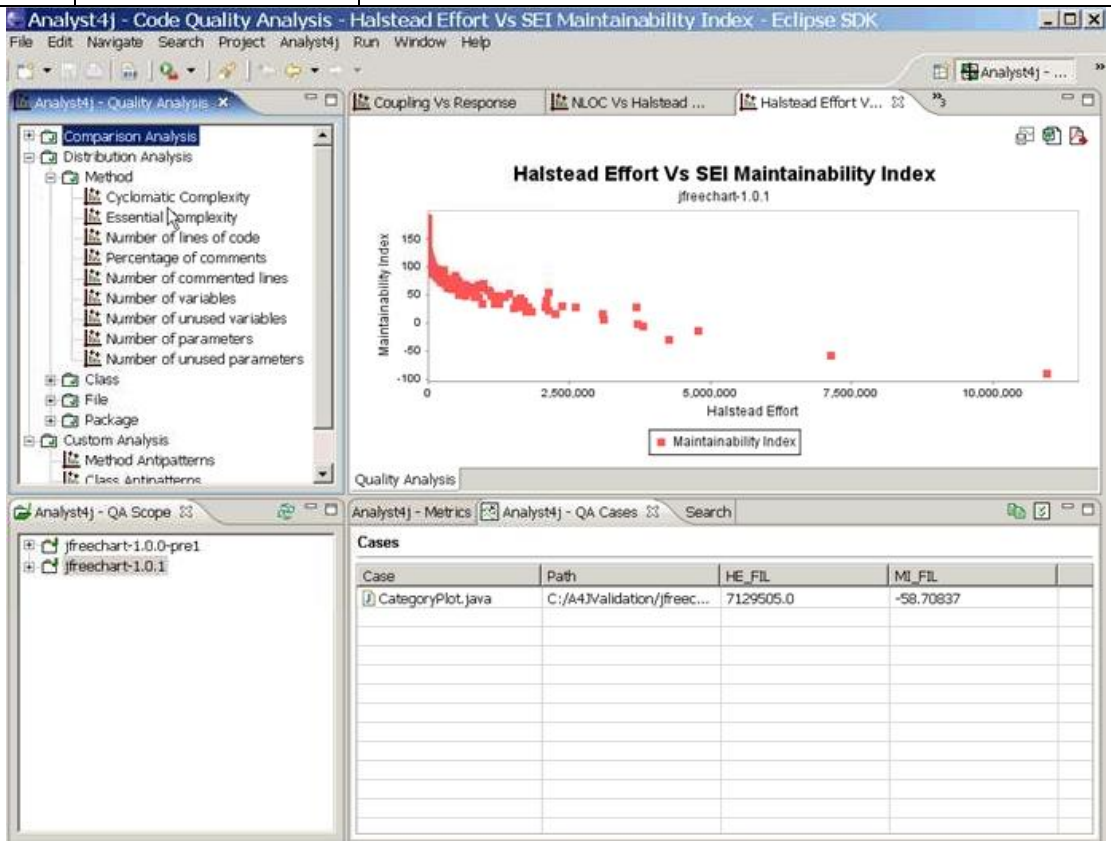


Рис. 4. Приклад роботи Analyst4j

2.1.6. CCCC

CCCC - інструмент для аналізу вихідного коду в мовах C та C++. Це консольна програма. Звіти по виміряних метриках зберігаються у HTML файлі. В останніх версіях може бути проведено аналіз коду на Java & Ada95.

Метрики CCCC:

Обгрунтування вибору засобу вимірювання

Критерії вибору:

1. Наявність обраних метрик

Всі з вимірюваних метрик є у iPlasma.

В Analyst4j та CCCC відсутні:

- Brain Class
- Brain Method
- God Class
- Data Class

В CCCC відсутні:

- NOM/NOC
- NOC/NOP
- CALL/NOM
- FOUT/CALL

2. Системні вимоги

iPlasma і CCCC не потребують встановлення додаткових програмних засобів.

Analyst4j потребує:

- JRE 5.0
- Eclipse 3.1 or higher

3. Зручність використання

- За зручністю використання найгіршою є CCCC, т.я. доступно тільки в консольному режимі, і надає звіти тільки у форматі HTML, які незручно програмно обробляти.
- В Analyst4j є більше можливостей для вимірювання та налаштувань, що дозволяє швидше оброблювати велику кількість однотипних проектів.
- У iPlasma автоматизація відсутня, і немає налаштувань, тому обробка великої кількості проектів може зайняти багато часу, порівняно з Analyst4j. Але маємо звіти в зручному форматі для програмної обробки.

Засобом вимірювання метрик обрано iPlasma.

2.2. Вибір методів статистичного аналізу для метрик версій програмного забезпечення.

Обчислення статистичних характеристик

Базовою кількісною характеристикою вибірки або варіаційного ряду є початкові статистичні моменти *k-го (k* = 1, 2,...) порядку:

$$\hat{\nu}_k = \frac{1}{n}\sum_{i=1}^{n} x_i^k = \frac{1}{n}\sum_{i=1}^{r} x_i^k n_i = \sum_{i=1}^{r} x_i^k f_i$$

та центральні моменти

$$\hat{\mu}_k = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\nu}_1)^k = \frac{1}{n}\sum_{i=1}^{r}(x_i - \hat{\nu}_1)^k n_i = \sum_{i=1}^{r}(x_i - \hat{\nu}_1)^k f_i .$$

Між $\hat{\mu}_k$ та $\hat{\nu}_k$ існує зв'язок:

$$\hat{\mu}_2 = \hat{\nu}_2 - \hat{\nu}_1^2, \quad \hat{\mu}_3 = \hat{\nu}_3 - 3\hat{\nu}_1\hat{\nu}_2 + 2\hat{\nu}_1^3,$$

$$\hat{\mu}_4 = \hat{\nu}_4 - 4\hat{\nu}_1\hat{\nu}_3 + 6\hat{\nu}_1^2\hat{\nu}_2 - 3\hat{\nu}_1^4,$$

$$\dots$$

$$\hat{\mu}_k = \sum_{s=0}^{k}\binom{k}{s}(-\hat{\nu}_1)^s \hat{\nu}_{k-s} .$$

Математичне сподівання

У теорії ймовірностей, очікуване значення (або очікування, або математичне очікування, або середня, або перший момент) випадкова величина середньозваженої всіх можливих значень, що ця випадкова величина може взяти на себе. Ваги, які використовуються при розрахунку цієї середньої відповідають ймовірності в разі дискретної випадкової величини, щільності або у випадку безперервної випадкової величини. З суворої теоретичної точки зору, очікуване значення інтеграла від випадкової величини щодо її імовірнісну міру.

Очікуване значення може бути інтуїтивно зрозуміла закон великих чисел: очікуване значення, коли воно існує, майже напевно межа вибіркового середнього як розмір вибірки зростає до нескінченності. Більш

неформально, це може бути витлумачено як довгострокових середніх результатів багатьох незалежних повторень експерименту (наприклад, рулон кістки). Значення не може чекати в загальному сенсі, "очікуване значення" саме по собі може бути малоймовірно або взагалі неможливо (наприклад, при наявності 2,5 дитини), як і вибіркового середнього.

Очікуване значення не існує для деяких дистрибутивів з великими "хвостами", такі як розподіл Коші.

Можна побудувати очікуване значення дорівнює ймовірність події, приймаючи очікування індикатор функції, яка є одним, якщо подія відбулася, і нулю інакше. Це співвідношення може бути використаний для перекладу властивості очікувані значення у властивості ймовірностей, наприклад, використанням закону великих чисел для обгрунтування оцінки ймовірності за частотами.

Визначення

Припустимо, що випадкова величина X може приймати значення x1 з імовірністю p1, значення x2 з імовірністю p2, і так далі, аж до значення xk з імовірністю p. Потім очікування цієї випадкової величини X визначається як

$$\mathrm{E}[X] = x_1 p_1 + x_2 p_2 + \ldots + x_k p_k \ .$$

Так як всі ймовірності пі додати до одного: P1 + P2 + ... + P = 1, очікуване значення можна розглядати як зважений інтегральний замір, з пі буття ваги:

$$\mathrm{E}[X] = \frac{x_1 p_1 + x_2 p_2 + \ldots + x_k p_k}{p_1 + p_2 + \ldots + p_k} \ .$$

Якщо всі результати XI рівноймовірно (тобто, p1 = p2 = ... = pк), то середньозважена перетворюється в простий середньої. Це інтуїтивне: очікуване значення випадкової величини є середнє значення це може зайняти, таким чином очікуване значення, що ви очікуєте відбудеться в

середньому. Якщо результати XI НЕ рівноймовірно, то середнє арифметичне повинні бути замінені зважений інтегральний замір, який бере до уваги той факт, що деякі результати більш вірогідні, ніж інші. Інтуїція, однак, залишається той же: очікуване значення X є те, що ви очікуєте, відбудеться в середньому.

Дисперсія

Дисперсія випадкової величини - міра розкиду даної випадкової величини, тобто її відхилення від математичного сподівання. Позначається D [X] в російській літературі і (англ. variance) у закордонній. У статистиці часто вживається позначення або. Квадратний корінь з дисперсії, рівний, називається середньоквадратичним відхиленням, стандартним відхиленням або стандартним розкидом. Стандартне відхилення вимірюється в тих же одиницях, що і сама випадкова величина, а дисперсія вимірюється в квадратах цієї одиниці виміру.

З нерівності Чебишева випливає, що випадкова величина віддаляється від її математичного сподівання на більш ніж k стандартних відхилень з імовірністю менше 1 / k ². Так, наприклад, як мінімум в 75% випадків випадкова величина віддалена від її середнього не більше ніж на два стандартних відхилення, а в приблизно 89% - не більше ніж на три.

Визначення

Нехай - випадкова величина, визначена на деякому вероятностном просторі. Тоді

$$D[X] = M\left[(X - M[X])^2\right]$$

де символ M означає математичне очікування.

Середньоквадратичне відхилення

Стандартне відхилення є широко використовуваним вимірювання мінливості або різноманіття, використовуваних в статистиці та теорії ймовірностей. Вона показує, скільки варіації або "дисперсії" є від середнього (середнє або очікуване значення). Низьке стандартне відхилення вказує, що

дані точки, як правило, дуже близька до середньої, у той час як високе стандартне відхилення вказує, що дані розподілені в широкому діапазоні значень.

Технічно, стандартне відхилення статистичної населення, даних, або розподіл ймовірностей квадратному кореню з дисперсії. Це алгебраїчно простий, хоча практично менш надійні, ніж середнє абсолютне відхилення. Корисна властивість стандартним відхиленням в тому, що, на відміну від дисперсії, це виражається в тих же одиницях, що і дані.

Крім вираження мінливості населення, стандартне відхилення зазвичай використовується для вимірювання довіри до статистичних висновків. Наприклад, похибка у виборчих даних визначається шляхом розрахунку очікуваного стандартного відхилення результатів, якщо ж опитування повинні були проводитися кілька разів. Повідомили похибка, як правило, приблизно в два рази стандартне відхилення - радіус 95 довірчий інтервал відсотків. У науці, вчені зазвичай повідомляють стандартне відхилення експериментальних даних, і лише ефекти, які далеко за межами діапазону стандартного відхилення вважаються статистично значимими - нормальна випадкова помилка чи зміни вимірювань, таким чином, відрізняється від причинного варіації. Стандартне відхилення також має важливе значення в галузі фінансів, де стандартне відхилення від норми прибутку на інвестиції міра волатильності інвестицій.

Середньоквадратичне відхилення для нормального розподілу ймовірності можна побачити на Рис. 4.

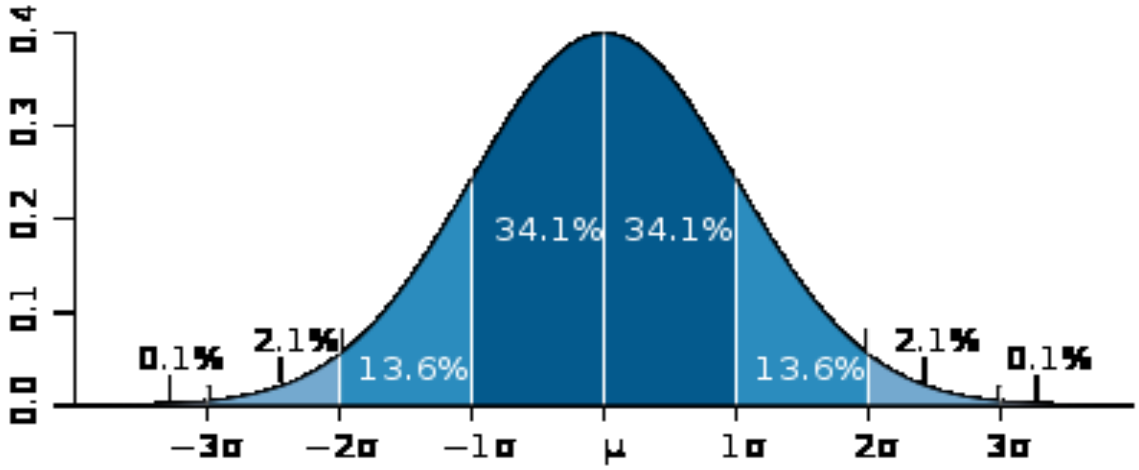


Рис. 5. Середньоквадратичне відхилення для нормального розподілу

Визначення населення значень

Нехай X-випадкова величина з середнім значенням μ:

$$\mathrm{E}[X] = \mu.$$

При цьому оператор E позначає середню або очікуване значення X. Тоді стандартне відхилення X є кількість

$$\sigma = \sqrt{\mathrm{E}[(X - \mu)^2]}.$$

Тобто, стандартне відхилення σ (сигма) є квадратний корінь з дисперсії X, тобто квадратний корінь із середнього значення (X − μ)2.

Стандартне відхилення (одномірні) розподіл ймовірностей ж, що і випадкова величина, що має, що розподіл. Не всі випадкові величини мають стандартне відхилення, так як ці очікувані значення може не існувати. Наприклад, стандартне відхилення випадкової величини, який слід розподілу Коші не визначено, тому що його очікуваний μ значення не визначено.

Коефіцієнт асиметрії

У теорії ймовірності та статистики, асиметрія є мірою асиметрії розподілу ймовірностей речова випадкова величина. Асиметрії значення може бути позитивним чи негативним, або навіть не визначено. Якісно, негативна асиметрія вказує, що хвіст на лівій стороні функція щільності

ймовірності довше правій стороні і велика частина значень (можливо, в тому числі середній) лежать праворуч від середнього значення. Позитивний нахил вказує, що хвіст на правій стороні більше, ніж з лівого боку і більшу частину значення лежать ліворуч від середнього значення. Нульове значення вказує, що значення відносно рівномірно розподілені по обидві сторони означає, як правило, але не обов'язково передбачає симетричний розподіл.

Розглянемо розподіл на малюнку. Бари на правій стороні конуса розподіл інакше, ніж грати на лівій стороні. Ці звужується сторони називаються хвостами, і вони забезпечують візуальні засоби для визначення, який з двох видів асиметрії розподілу:

негативна асиметрія: лівий хвіст довший, маса розподіл зосереджено на правому малюнку. Вона має відносно невелике число малих величин. Розподіл називається лівий перекіс або "перекіс вліво". Приклад (спостережень): 1,1000,1001,1002,1003

позитивний нахил: правий хвіст довший, маса розподіл зосереджено на лівій частині малюнка. Вона має відносно невелике число високих значень. Розподіл називається правою перекіс або «перекіс вправо". Приклад (спостережень): 1,2,3,4,100.

Якщо розподіл симетрично, то середнє = середній і є нулю асиметрії. (Якщо, крім того, розподіл одним видом, то середнє = середній режим =). Це випадок підкидання монети або серії 1,2,3,4, ... Однак слід зазначити, що зворотне твердження невірне в цілому, тобто нулю асиметрії не означає, що середнє = середній.

Приклади для позитивного та негативного коефіцієнту асиметрії можна побачити на Рис. 6.

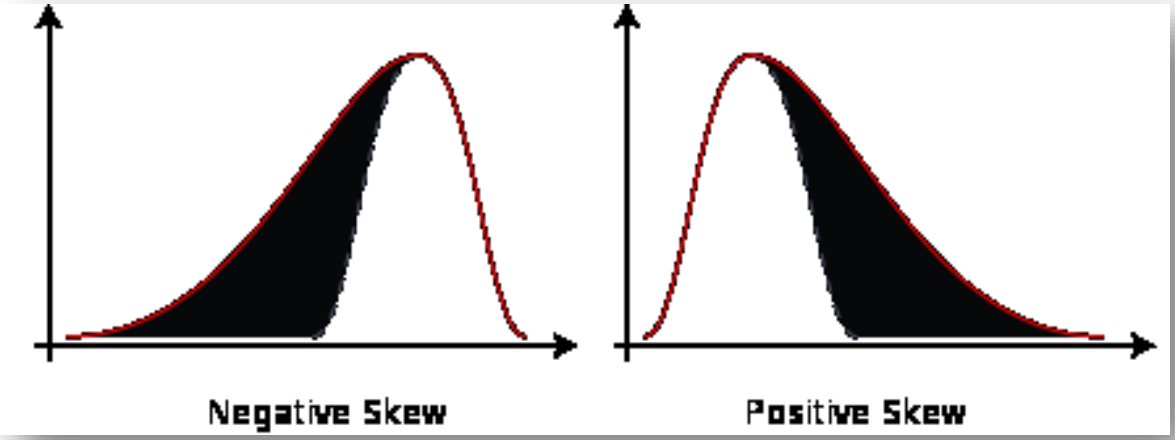


Рис. 6. Позитивний та негативний коефіцієнт псиметрії

Визначення

Асиметрія випадкової величини X є третім стандартизованих момент, що позначається γ1 і визначається як

$$\gamma_1 = \mathrm{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \frac{\mu_3}{\sigma^3} = \frac{\mathrm{E}\left[(X-\mu)^3\right]}{(\mathrm{E}\left[(X-\mu)^2\right])^{3/2}} = \frac{\kappa_3}{\kappa_2^{3/2}},$$

μ3, де є третій момент про середню μ, σ є стандартне відхилення, і E, є очікування оператора. Останнє рівність висловлює асиметрія з точки зору співвідношення третє κ3 кумулянт і 1.5th Потужність другої κ2 кумулянт. Це аналогічно визначенню ексцес як четвертого кумулянт нормований на квадрат другий кумулянт.

Асиметрії також іноді позначається косою [X].

Формула, що виражає асиметрія з точки зору не-центральний момент E [X3] може бути виражена шляхом розширення попередньої формулою,

$$\gamma_1 = \mathrm{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \frac{\mathrm{E}[X^3] - 3\mu\mathrm{E}[X^2] + 3\mu^3 - \mu^3}{\sigma^3} = \frac{\mathrm{E}[X^3] - 3\mu\sigma^2 - \mu^3}{\sigma^3}.$$

Коефіцієнт асиметрії задає ступінь асиметричності щільності ймовірності щодо осі, що проходить через її центр ваги. Коефіцієнт асиметрії визначається третім центральним моментом розподілу. У будь-якому симетричному розподілі з нульовим математичним очікуванням, наприклад нормальним, всі непарні моменти, в тому числі і третій, дорівнюють нулю, тому коефіцієнт асиметрії теж дорівнює нулю.

Коефіцієнт ексцесу

У теорії ймовірності та статистики, ексцес (від грецького слова κυρτός, kyrtos або kurtos, тобто опуклі) є мірою "островершінность" про розподіл ймовірностей речова випадкова величина, хоча деякі джерела наполягають, що важкі хвости, і не островершінность, є те, що дійсно вимірюється ексцес. Вища ексцес коштів більше різниці, є результатом рідко граничні відхилення, на відміну від частих скромно розміру відхилень.

Визначення

Нехай задана випадкова величина, така що $\mathbb{E}|x|^4<\infty$.

Коефіцієнт ексцесу розподілу випадкової величини визначається формулою:

$$\gamma_2=\frac{\mu_4}{\sigma^4}-3,$$

де

$\mu_4=\mathbb{E}\left[(x-\mathbb{E}x)^4\right]$ — четвертий центральний момент випадкової величини x;

$\sigma^2=\mathbb{D}[x]=\mathbb{E}\left[(x-\mathbb{E}x)^2\right]$ — дисперсія або другий центральний момент випадкової величини x;

Нормальний розподіл має нульовий ексцес, $\gamma_2=0$.

Якщо хвости розподілу «легше», а пік гостріше, ніж у нормального розподілу, то $\gamma_2>0$.

Якщо хвости розподілу «важчим», а пік більш «приплюснутий», ніж у нормального розподілу, то $\gamma_2<0$

Область можливих значень ексцесу $\gamma_2 \in [-2,\infty)$.

Обгрунтування

Варіаційні ряди в даному випадку не використовуються, т.я. кількість значень у кожному ряді 10 (мала кількість), і вони є унікальними.

Мода і медіана також не використовується так як не дає практичної користі. Мода вказує найбільш ймовірне значення вибірки, а у заданих всі значення є унікальними, тобто з однаковою ймовірністю. Медіана ділить площу під гістограмою на дві рівні частини, що в при порівнянні з математичним сподіванням може вказати на розподіл величин в ряді. В даному випадку використаємо для цього коефіцієнт асиметрії.

Вимірюється:

Математичне сподівання

Дає змогу побачити середнє значення всього ряду

Дисперсія

Міра відхилення випадкової величини від математичного сподівання

Середньоквадратичне відхилення

Міра розсіювання випадкової величини від математичного сподівання.

Коефіцієнт асиметрії.

Коефіцієнт асиметрії задає ступінь асиметричності щільності ймовірності щодо осі, що проходить через її центр ваги. Коефіцієнт асиметрії визначається третім центральним моментом розподілу. У будь-якому симетричному розподілі з нульовим математичним очікуванням, наприклад нормальним, всі

непарні моменти, в тому числі і третій, дорівнюють нулю, тому коефіцієнт асиметрії теж дорівнює нулю.

Ступінь сглаженности щільності ймовірності в околиці головного максимуму задається ще однією величиною - коефіцієнтом ексцесу.

Він показує, наскільки гостру вершину має щільність ймовірності в порівнянні з нормальним розподілом. Якщо коефіцієнт ексцесу більше нуля, то розподіл має більш гостру вершину, ніж розподіл Гауса, якщо менше нуля, то більш плоску.

2.3. Порівняння результатів статистичного аналізу прикладного програмного забезпечення.

M(X) – Математичне сподівання

D(X) – Дисперсія

Dev(X) – Середньоквадратичне відхилення

Skew(X) – Коефіцієнт асиметрії

Kurt(X) – Коефіцієнт ексцесу

M(X) – Математичне сподівання

Математичне сподівання відображає середнє значення набору метрик.

Математичним сподіванням визначаємо середнє (в нашому випадку) значення, т.я. всі значення мають однакову вірогідність.

D(X) – Дисперсія

Дисперсія вказує відхилення значень метрик з набору відносно математичного сподівання.

Dev(X) – Середньоквадратичне відхилення

Середньоквадратичне відхилення вказує розсіювання значень метрик з набору відносно математичного сподівання.

Середньоквадратичне відхилення дає змогу побачити на яку максимальну величину відхиляється графік метрики від математичного сподівання (визначає ступінь змін).

Skew(X) – Коефіцієнт асиметрії

Задає степінь асиметричності розподілу набору метрик.

Якщо коефіцієнт асиметрії від'ємний, це означає що головний максимум знаходиться в правій частині графіку, що означає ріст характеристик за часом.

Якщо коефіцієнт асиметрії додатній, це означає що головний максимум знаходиться в лівій частині графіку, що означає зниження характеристик за часом.

Kurt(X) – Коефіцієнт ексцесу

Вказує на гостроту вершини головного максимуму.

Якщо коефіцієнт ексцесу від'ємний, це означає що пік головного максимуму на графіку згладжено, що значить плавні зміни за часом.

Якщо коефіцієнт ексцесу додатний, це означає що пік головного максимуму на графіку є гострим, що значить різкі зміни за часом. Чим вищий коефіцієнт, тим більш різкі зміни.

Якщо коефіцієнти асиметрії і ексцесу дорівнюють нулю, маємо нормальний розподіл.

Підведемо висновки. Математичне сподівання визначає середнє значення, середньоквадратичне відхилення вказує максимальне відхилення від математичного сподівання. Коефіцієнт асиметрії визначає в якій частині графіку знаходиться головний максимум, коефіцієнт ексцесу визначає гостроту піку головного максимуму. Поєднання статистичних характеристик дозволяє визначити форму графіку, не будуючи його, та прослідкувати ключові зміни.

3.ЗАСІБ ДОСЛІДЖЕННЯ МЕТРИК ВЕРСІЙ

3.1. Вибір засобів розробки.

Розробка навчального прикладного програмного забезпечення такого плану потребує пакету програм для рішення проблеми.

ПЗ має реалізовувати такі функції:

1. Імпорт звітів з метриками (робота з файлами)
2. База даних для зберігання проектів і значень метрик (робота з БД)
3. Розрахунок статистичних характеристик для наборів метрик (математичні обчислення)
4. Побудова графіків для наборів метрик (компонент для побудови графіків)

Платформою для розробки є MS Windows. Для реалізації програмного продукту пакетом програм буде:

- .NET Framework 4.0
- C#
- Microsoft Visual Studio 2010 C# Express

Сервер для рішення задачі розгортається локально, має підтримувати мінімальну функціональність і взаємодіяти з Microsoft Visual Studio 2010 C# Express. Тому обираємо:

- MS SQL Server Compact 3.5

Використання даного серверу дозволить спростити розробку, так як, порівняно з MS SQL Server 2008, встановлення та налагодження проходить простіше, можливе використання бази даних без наявності самого серверу, що більше підходить для локального проекту.

Інтерфейс для програмного продукту можне бути реалізований на Windows Forms або WPF. WPF є новітньою технологією, порівняно з Windows Forms, опис інтерфейсу реалізовано на мові XAML, що дає змогу розширювати та змінювати використані компоненти інтерфейсу.

- WPF

Для малювання графіків існує компонент Teleric Rad Tools for WPF (Developer Edition), що можна використовувати як доповнення до інтерфейсу WPF.

- Teleric Rad Tools for WPF (Developer Edition)

Повний список програмного забезпечення для рішення здачі:

- .NET Framework 4.0
- C#
- Microsoft Visual Studio 2010 C# Express
- MS SQL Server Compact 3.5
- WPF
- Teleric Rad Tools for WPF (Developer Edition)

3.2. Проектування бази даних.

Загальні положення

База даних має складатись з проектів і їх версій з виміряними метриками.

При доступі до бази даних користувач спочатку отримує доступ до проектів, де може додати, видалити проект, змінити його назву, або перейти до метрик версій проекту.

При переході до метрик версій проекту мають відображатись тільки метрики обраного версій обраного проекту. Користувач може додати, видалити, змінити метрики.

Сутність Проекту

Короткий опис сутності. Містить інформацію про назву проекту.

Атрибути. Сутність характеризується наступними атрибутами:

- Ідентифікаційний номер проекту
- Назва проекту

Зв’язки. Сутність має наступні зв’язки з іншими сутностями:

- Один ПРОЕКТ відповідає від 0 до n ЗНАЧЕННЯМ

Бізнес–правила.

- «Назва проекту» унікально ідентифікує Проект
- При видаленні проекту видаляються всі Значення, що посилаються на нього

Сутність Значення

Короткий опис сутності. Містить інформацію про значення метрик кожної версії проекту.

Атрибути. Сутність характеризується наступними атрибутами:

- Ідентифікаційний номер версії
- Назва проекту
- Назва версії
- Avg_NDD
- Avg_HIT
- NOP
- Sum_NOC
- Sum_NOM
- Sum_LOC
- Sum_CYCLO
- Sum_CCL
- Sum_FANOUTCLASS
- Sum_CBo
- brainClass
- brainMethod
- godClass
- dataClass

Зв'язки. Сутність має наступні зв'язки з іншими сутностями:

- Одне ЗНАЧЕННЯ відповідає одному ПРОЕКТУ

Бізнес–правила.

- «Ідентифікаційний номер» унікально ідентифікує Значення
- Значення не може існувати без Проекту

Схема БД

Схема БД наведена на Рис. 6.

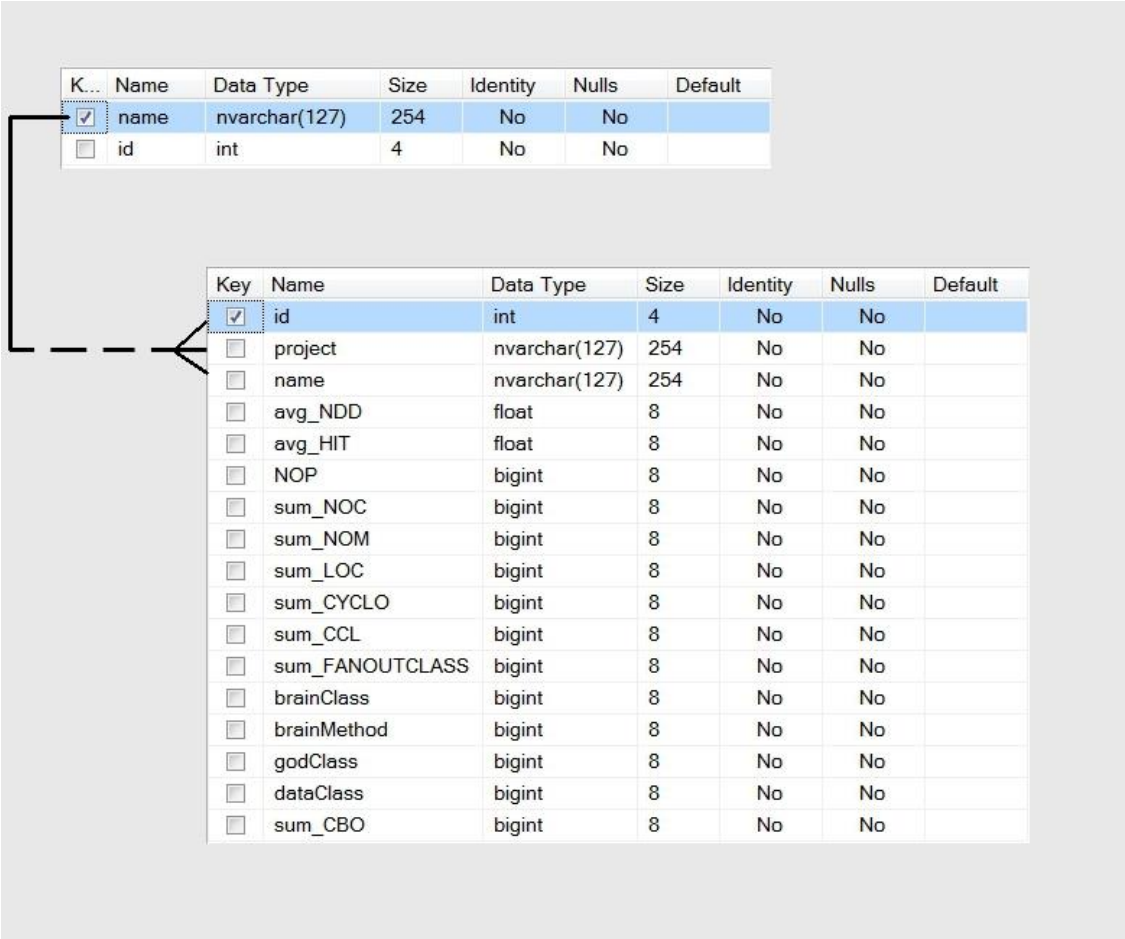


Рис. 7. Схема БД

3.3. Проектування засобу дослідження метрик версій.

Функціональні вимоги:

1. Робота з проектами
a. Додати проект
b. Видалити проект
c. Змінити проект
d. Сортувати проекти по ідентифікаційному номеру або назві
e. Перейти до списку метрик версій проекту
2. Робота з метриками версій
a. Додати версію - імпорт звітів з метриками (робота з файлами)
b. Видалити версію

c. Редагувати версію

d. Сортувати проекти по ідентифікаційному номеру або назві

e. Побудувати графіки та статистику

Не функціональні вимоги:

1. Розміщення на локальному комп’ютері
2. База даних для зберігання проектів і значень метрик (робота з БД)

Інтерфейс проекту:

При запуску програми користувач бачить список доступних проектів (Рис. 7.)

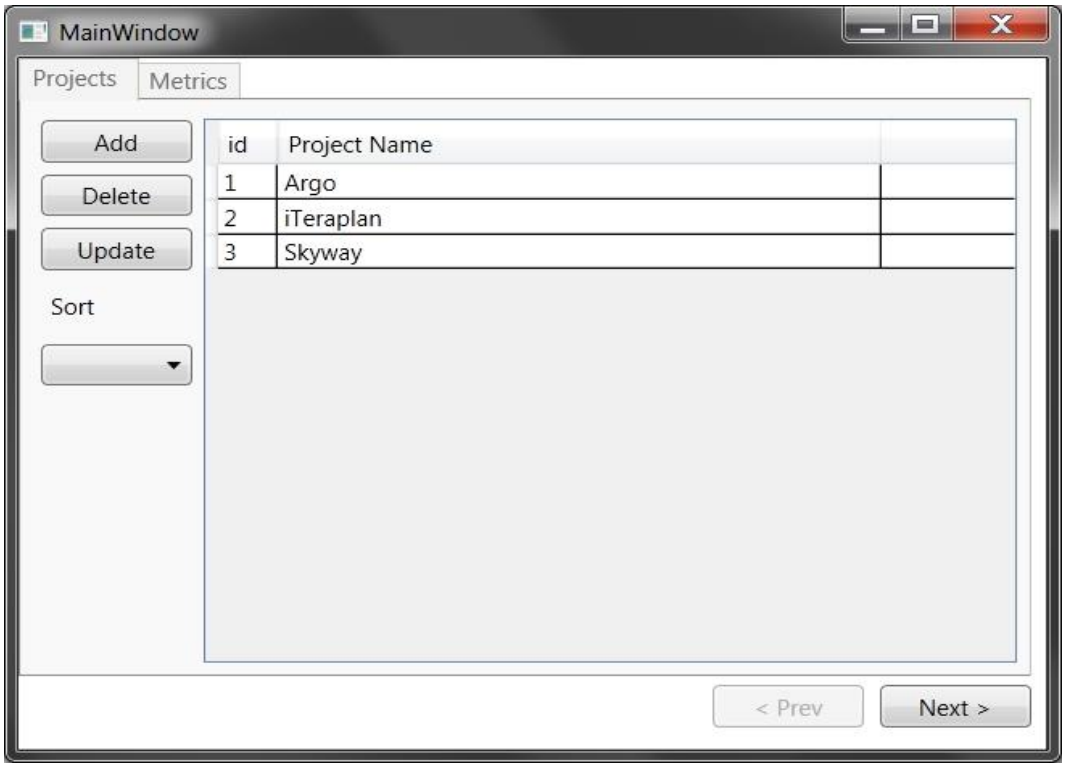


Рис. 8. Список проектів

Доступні функції вікна «Проект» (Рис. 7.):

- Додати проект
- Видалити проект
- Змінити проект
- Сортувати проекти по ідентифікаційному номеру або назві
- Перейти до списку метрик версій проекту

При переході до списку метрик версій проекту, в тому самому вікні з’являється список метрик версій проекту (Рис. 8.)

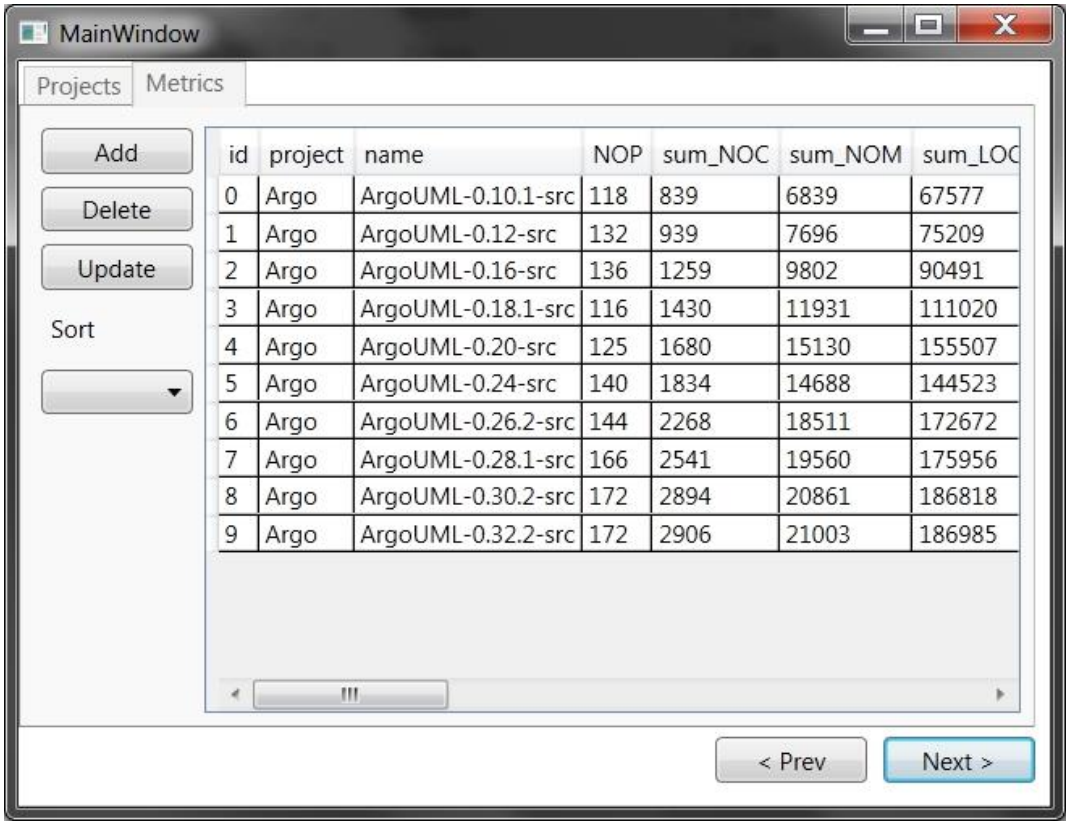


Рис. 9. Список метрик версій проекту

Доступні функції вікна «Метрики» (Рис. 8.):

- Додати версію
- Видалити версію
- Редагувати версію
- Сортувати проекти по ідентифікаційному номеру або назві
- Побудувати графіки та статистику

Графіки та статистика будуються у новому вікні (Рис. 9.)

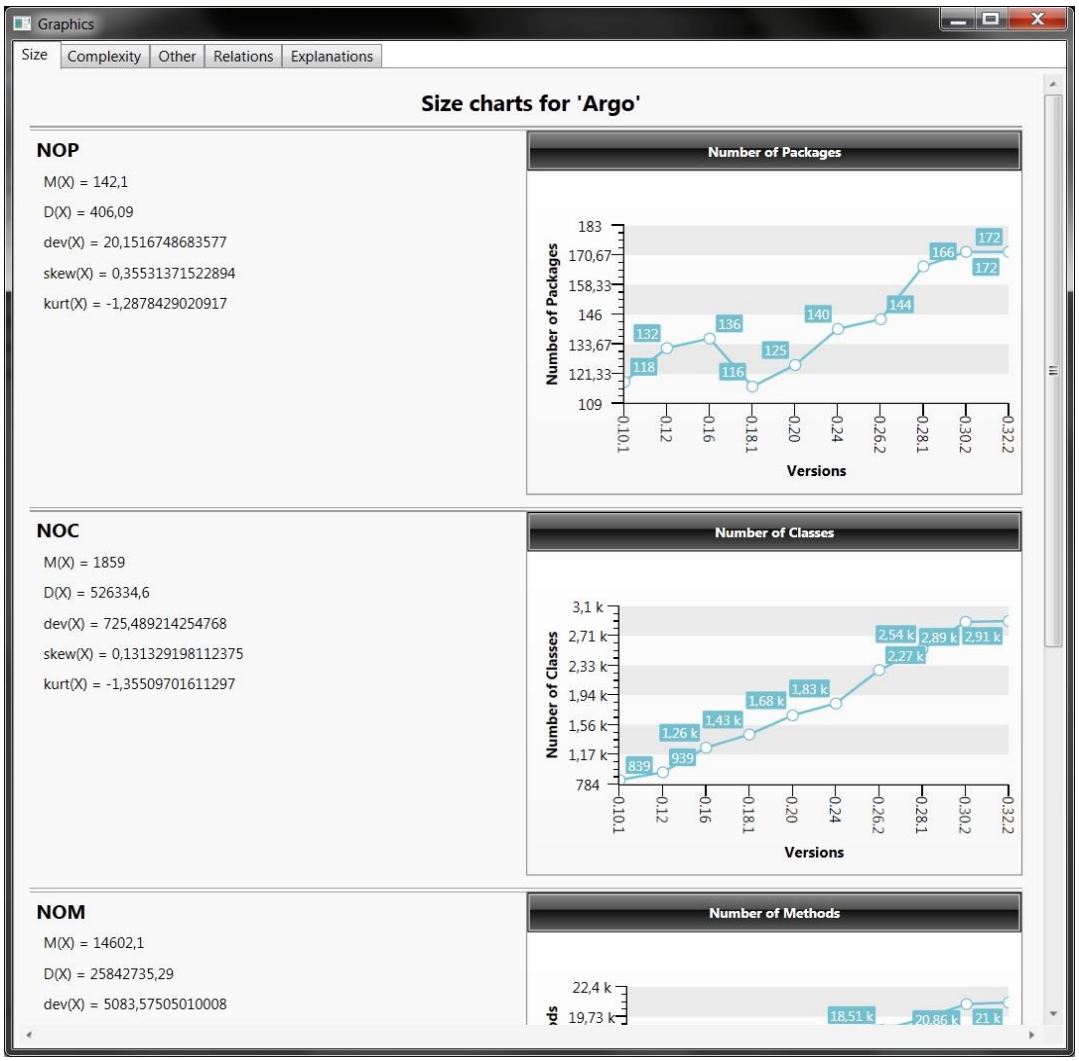


Рис. 10. Графіки та статистика

У вікні графіків та статистики (Рис. 10.) метрики згруповано по закладкам. Користувач може переміщатись по закладкам обираючи групу метрик.

Архітектура проекту

Діаграми залежностей (Рис. 11.)

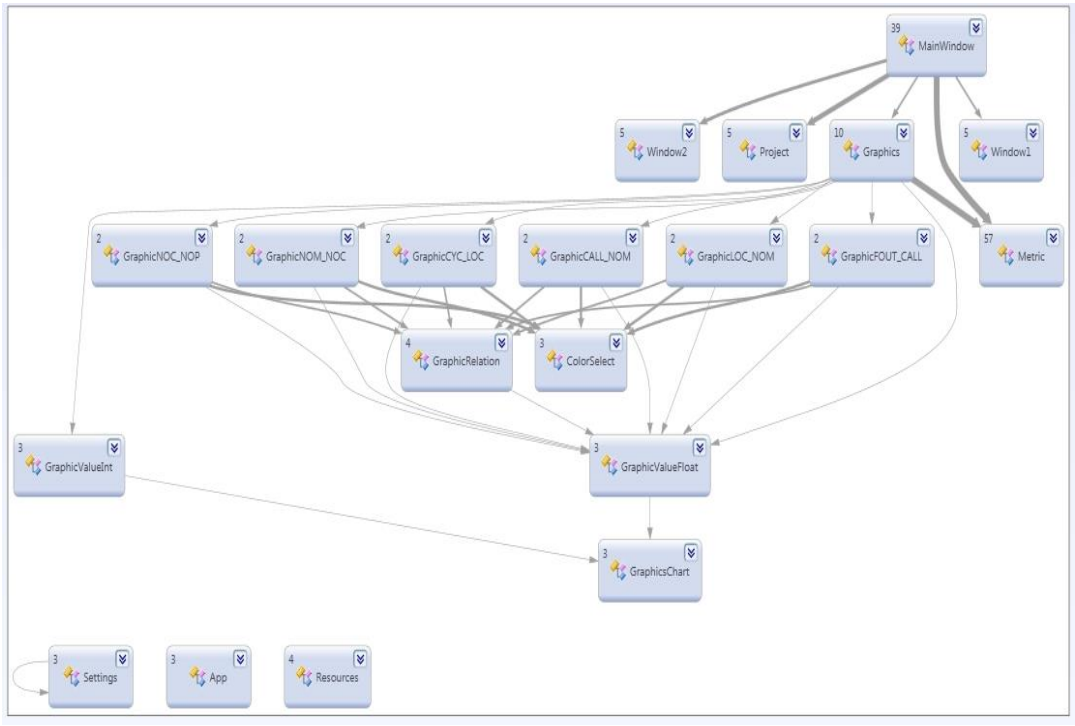


Рис. 11. Діаграма залежностей

Діаграми класів (Рис. 12.)

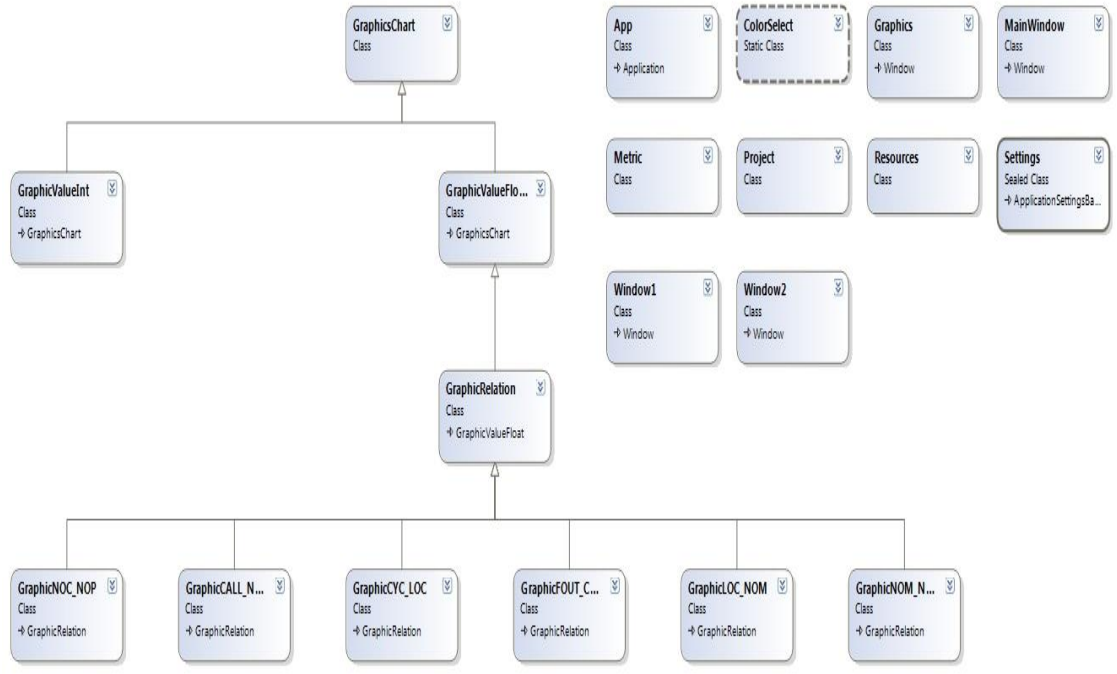


Рис. 12. Діаграма класів

4.ЗАСТОСУВАННЯ ЗАСОБУ ДОСЛІДЖЕННЯ МЕТРИК ВЕРСІЙ ПРИКЛАДНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

4.1. Вибір прикладного програмного забезпечення для дослідження та вимірювання метрик.

Критерії вибору проектів:

- Проекти мають бути написані на Java, з відкритим кодом (для аналізу в iPlasma)
- Проекти мають бути великими – більше 1000 класів
- Мають бути доступними не менше 10 версій проекту
- Проекти мають суттєво відрізнятись у своїй версійності

4.1.1. Argo UML

ArgoUML це потужна і проста у використанні інтерактивних, графічна середовище розробки програмного забезпечення, що підтримує проектування, розробки та документування об'єктно-орієнтованих програм.

Якщо ви знайомі з сімейством програмних додатків, званих автоматизації інженерного програмного забезпечення (CASE) інструментів, то ви повинні знайти ArgoUML миттєво знайомі.

Користувачів ArgoUML є розробники програмного забезпечення і архітекторів, розробників програмного забезпечення, бізнес-аналітиків, системних аналітиків та інших фахівців, що беруть участь в аналізі, проектуванні та розробці програмних додатків. Основні риси:

- Відкриті стандарти: XMI, SVG і PGML
- 100% залежить від платформи, завдяки виключно для використання Java
- Open Source, дозволяє розширювати і налаштуванні.
- Когнітивні функції, такі як: відображення в дії, опортуністичні дизайну, розуміння і вирішення проблем

Даний проект обрано через його найдовший термін розробки (2001-2009рр) та доступність всіх версій на протязі розробки. Проект весь час розробки був відкритим.

4.1.2. iTeraplan Community Edition

iTeraplan є веб-архітектури підприємства Управління IT-інструмент для моделювання ландшафтів і візуалізації. Ви можете документувати, аналізувати, планувати і контролювати свій IT-ландшафт відповідно до бізнес-цілей. Випробування можливостей і ефектів бізнес-ідей.

Перша версія проекту (1.0) була закритою. Після виходу другої версії (2.0) проект розділився на дві частини - Community Edition та Enterprise Edition. Enterprise Edition є платною програмою, з ширшою функціональністю. Community Edition розповсюджується по ліцензії Affero GNU Public License та AGPLv3 Open Source Licence, доступний відкритий код. Роки розробки: 2008-2011рр.

4.1.3. Skyway Builder Community Edition

Скайуей Builder Community Edition (CE) є Eclipse основі, генерація коду інструмент для прискорення розвитку Весна додатків. Весна можливості MVC лісу дозволяють користувачам створювати Весна-основі Java-додаток CRUD у хвилинах.

Версії проекту 1.0-5.0 є закритими, відкритою є тільки остання версія 6.0. Таким чином можна прослідкувати розвиток проекту, що довгий час розвивався закрито, після чого було надано змогу отримати вільний доступ до коду, документації, та прийняти участь в розробці. Проект будучи комерційним перетворився на відкритий, ліцензія Apache License V2.0, GNU General Public License version 3.0 (GPLv3). Роки розробки: 2008-2009рр.

4.2. Аналіз метрик версій програмного забезпечення.

Аналіз проектів:

- Argo UML
- iTeraplan Community Edition
- Skyway Builder Community Edition

Аналіз метрик версій програмного забезпечення розміщено в Додатку 1.

4.3. Порівняння результатів аналізу прикладного програмного забезпечення.

Порівняння результатів аналізу проектів:

- Argo UML
- iTeraplan Community Edition
- Skyway Builder Community Edition

Argo UML

В цілому видно що розмір проекту постійно збільшується по таким характеристикам, як NOC, NOM, LOC, при чому ріст проекту сповільнюється з часом.

По коефіцієнту ексцесу для NOC=-1.35, NOM=-1.42, LOC=-1.47 видно, що у більш пізніх версіях, починаючи з середини (версія 0.24), розмір класів зменшується і кількість коду в методах зменшується, це пов’язано з розбиттям задач на більшу кількість підзадач, ніж у ранніх версіях.

Щодо NOP, є провал трохи лівіше середини графіка (версія 0.18.1). Беручи до уваги незмінність значень метрик NOC, NOM, LOC і різке зростання NDD і HIT, можна зробити висновок про реструктуризацію проекту.

NDD та HIT в останніх версіях проекту збільшується, але NDD зростає швидше ніж HIT, це дає широкі можливості для повторного використання.

CYCLO, CCL, FANOUT, CBO збільшуються таким же чином як і NOC, NOM, LOC, що дає зробити висновок про поступове рівномірне нарощування функціональності.

У версії 0.24 CYCLO, NOM, LOC, Data Class стрімко падають, після чого продовжують зростання. Це означає про відмову від деякої функціональності або виправлення серйозної помилки, т.я. ріст інших метрик є стабільним.

Значення метрик Brain Method та God Class перестає збільшуватись після версії 0.20, в той час як Brain Class, Data Class продовжує зростати. Це означає, що логіка програми перерозподіляється по більшій кількості класів, і, якщо брати до уваги ріст NDD і HIT, можна зробити висновок, що це пов’язано з метою зменшити кількість дій виконуваних класами, і покращити повторне використання.

Відношення CYC/LOC, LOC/NOM, NOM/NOC, NOC/NOP, CALL/NOM, FOUT/CALL в цілому відповідають промисловим стандартам, і, як було сказано вище, класи та методи подрібнюються з часом, при цьому кількість класів у пакеті зростає.

ВИСНОВКИ

Підводячи підсумок, хотілося б відзначити, що жодної універсальної метрики не існує. Будь-які контрольовані метричні характеристики програми повинні контролюватися або в залежності один від одного, або в залежності від конкретного завдання, крім того, можна застосовувати гібридні заходи, проте вони так само залежать від простіших метрик і також не можуть бути універсальними. Строго кажучи, будь-яка метрика - це лише показник, який сильно залежить від мови і стилю програмування, тому ні одну міру не можна зводити в абсолют і ухвалювати будь-які рішення, грунтуючись тільки на ній.

Важливою властивістю наведених метрик є можливість передбачення рівня складності, который може бути виведений на основі вимірювання непрямих метрик. При цьому отримання вимірювання на верхньому рівні ієрархії здійснюється зазвичай розрахунком зваженої суми метрик нижнього рівня.

Досвід керування якістю показує, що фінансові витрати, зроблені для покращення якості продукту, є безумовно доцільними і дають у підсумку високий економічний ефект. Причина, по якій багато організацій утримуються від таких витрат, полягає, насамперед, у труднощах пов'язаних з плануванням і оцінкою результатів підвищення якості. Частою є ситуація, коли реалізується рішення про підвищення якості, грунтуючись на неформальних, інтуїтивних способах оцінки якості. Це неминуче веде до неефективного витрачання ресурсів і фактично збільшує реальну ціну якості.

Ретельно проведений метричний аналіз якості відповідно до цілей розробки створює основу для коректного планування та контролю витрат на якість для досягнення необхідних показників та ефективності використання ресурсів.

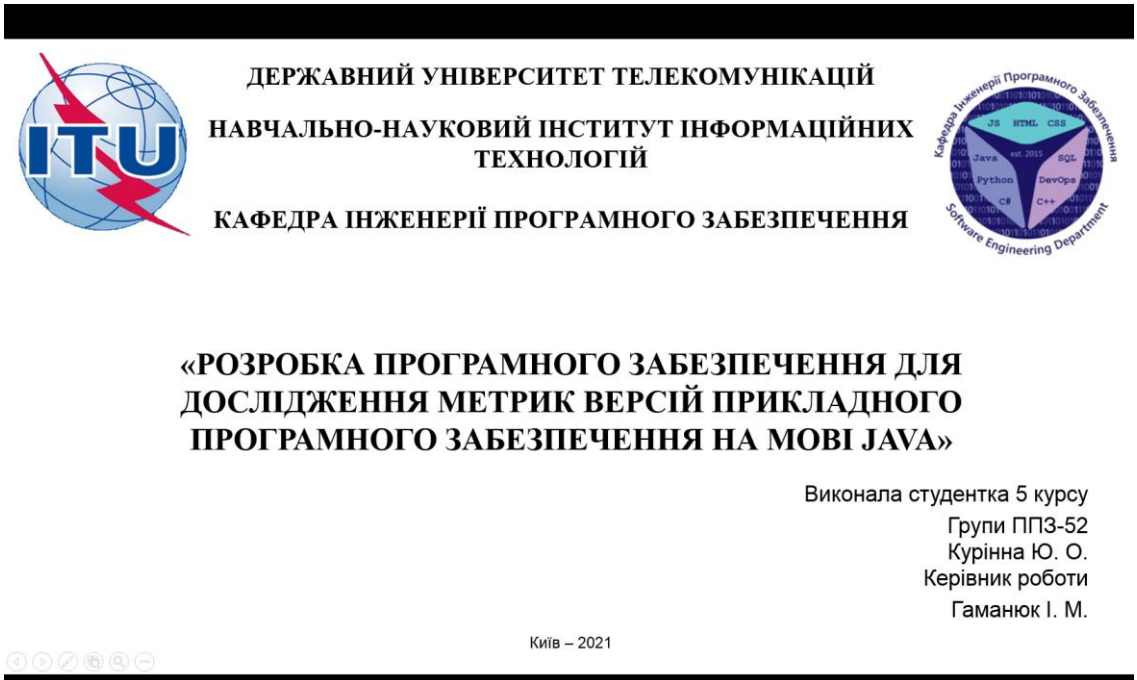
Таким чином, в даній роботі були отримані наступні результати:

- Defined metrics for research versiynosti

- Found three software products for analysis
- Measured metric versions for products found
- Developed application software for statistical analysis and charting metrics metrics depending on the version
- Analysis of metrics developed by the software analysis and interpretation of results

Розроблений програмний продукт рекомендовано використовувати для дослідження метрик версій програмного забезпечення. Продукт може бути розширений доданням функціональності в управлінні проектами та метриками версій, можуть бути додані нові метрики.

Демонстраційні матеріали



ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ДОСЛІДЖЕННЯ МЕТРИК ВЕРСІЙ ПРИКЛАДНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА МОВІ JAVA»

Виконала студентка 5 курсу
Групи ППЗ-52
Курінна Ю. О.
Керівник роботи
Гаманюк І. М.

Київ – 2021

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи**- Проаналізувати зміну метрик в процесі еволюції програмного забезпечення, визначити необхідність внесення змін до цього процесу з метою оптимізації проектів.
- **Об'єкт дослідження-** Зміна значень метрик програмного забезпечення залежно від версії.
- **Предмет досліджень-** Аналіз зміни метрик в процесі еволюції ПЗ та актуальність внесення змін з метою оптимізації у проектах.

2

ТЕХНІЧНІ ЗАВДАННЯ

- *1.* Дослідити предметну область
- *2.* Виділити найбільш підходящі метрики,для визначення версійності програмного забезпечення
- *3.* Аналізування значень метрик різних Java проектів та їхзміни в різних версіях
- *4.* Розробка програмного продукту,аналізує отримані значення метрик версій програмних проектів
- *5.* Порівняння результатів і аналіз зміни значень метрик

3

Засоби розробки

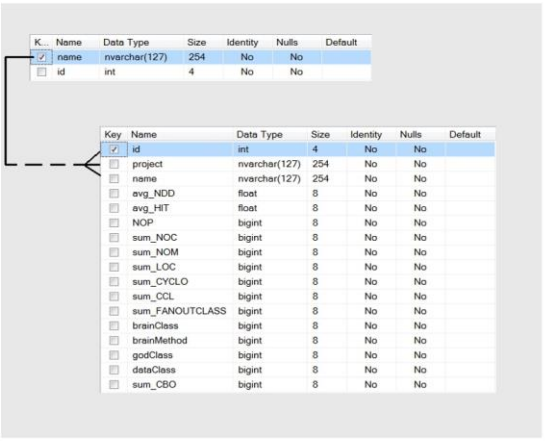
- .NET Framework 4.0
- C#
- Microsoft Visual Studio 2010 C# Express

Для розробки бази даних використовується:

- MS SQL Server Compact 3.5

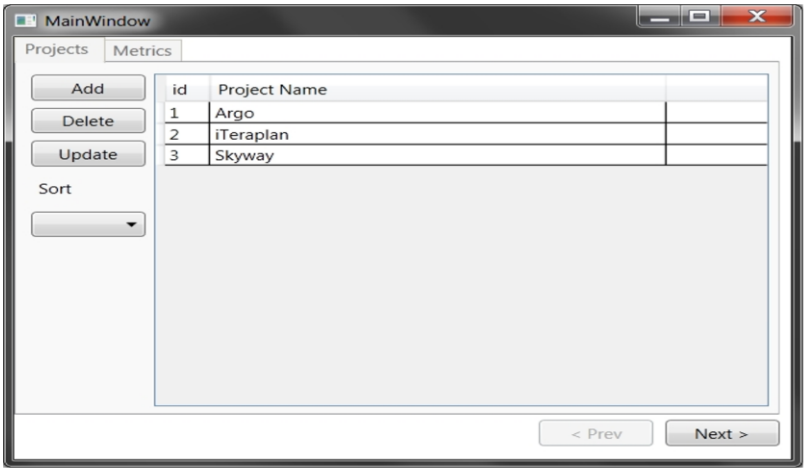
4

База даних



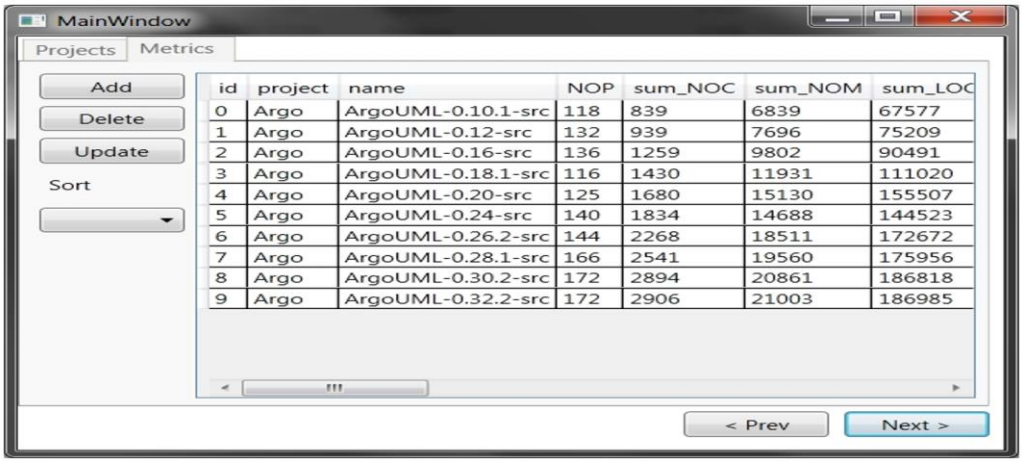
5

Список проектів



6

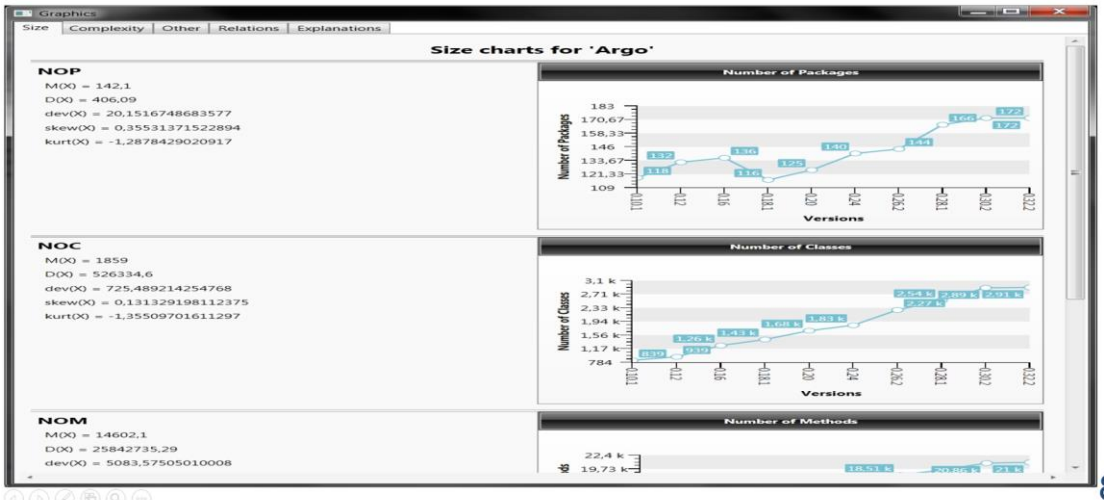
Список метрик версій проекту



id	project	name	NOP	sum_NOC	sum_NOM	sum_LOC
0	Argo	ArgoUML-0.10.1-src	118	839	6839	67577
1	Argo	ArgoUML-0.12-src	132	939	7696	75209
2	Argo	ArgoUML-0.16-src	136	1259	9802	90491
3	Argo	ArgoUML-0.18.1-src	116	1430	11931	111020
4	Argo	ArgoUML-0.20-src	125	1680	15130	155507
5	Argo	ArgoUML-0.24-src	140	1834	14688	144523
6	Argo	ArgoUML-0.26.2-src	144	2268	18511	172672
7	Argo	ArgoUML-0.28.1-src	166	2541	19560	175956
8	Argo	ArgoUML-0.30.2-src	172	2894	20861	186818
9	Argo	ArgoUML-0.32.2-src	172	2906	21003	186985

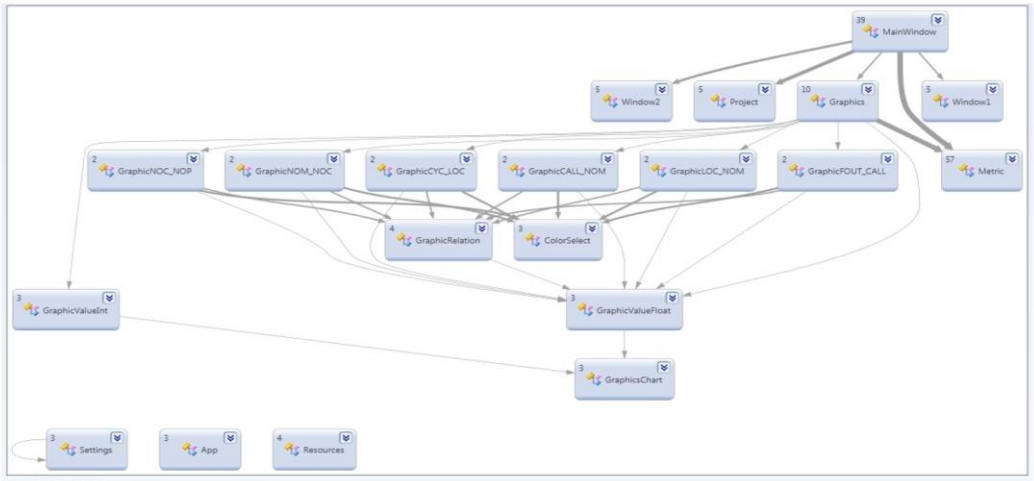
7

Графіки та статистика



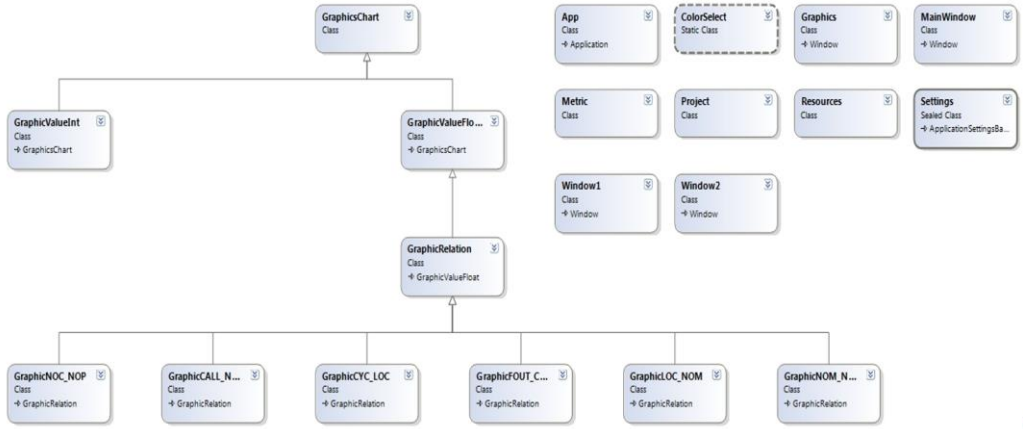
8

Діаграма залежностей



9

Діаграма класів



10

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Шевченко С.М. Використання криптографічних засобів для захисту корпоративної інформації / Ю.Д. Жданова, С.М. Шевченко // Актуальні проблеми забезпечення інформаційної та кібернетичної безпеки: Матеріали третьої міжнародної науково-технічної конференції. Збірник тез. 27.10.2017, ДУТ, м. Київ — К.: ДУТ, 2017. — С. 64.

ДЯКУЮ ЗА УВАГУ!