004.415.2.045 (076.5)

О.П. Дишлевий

МЕТРИЧНЕ ЗАБЕЗПЕЧЕННЯ ВЛАСТИВОСТЕЙ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Національний авіаційний університет
E-mail: oleksiy.dyshlevyy@livenau.net

Стаття присвячена підбору метрик для властивості «розуміння» програм. Проводився аналіз властивості «розуміння» та проводився для неї підбір метрик методом «ціль-питання-метрика». Збір даних здійснювався шляхом вимірювання підібраних метрик спеціальними засобами. Аналіз даних здійснювався з використанням предметно-орієнтованого методу побудови залежностей між метриками програмного забезпечення. В результаті аналізу було підтверджено метричне забезпечення властивості «розуміння» та розрахована вагомість кожної метрики для властивості.

Статья посвящена подбору метрик для свойства «понимаемость» программ. Проводился анализ свойства «понимаемость» и к нему проводился подбор метрик методом «цель-вопрос-метрика». Сбор данных проводился путем измерения специальными средствами подобранных метрик. Анализ данных осуществлялся с использованием предметно-ориентированного метода построения зависимостей между метриками программного обеспечения. В результате анализа было подтверждено метрическое обеспечение свойства «понимаемость» и рассчитана весомость каждой метрики для этого свойства.

The article is dedicated to selection of software metrics for attribute understandability. The attribute understandability has been analyzed and selection of metrics has been made by «goal-question-metrics» method. Data collection has been made with using subject-oriented method of making dependences between software metrics. As a result of analysis software metrics has been confirmed for attribute understandability. Validity of each metric was calculated for this attribute.

Ключові слова: метрики, властивості, вимірювання, метод «ціль-питання-метрика», кореляційний аналіз

Вступ

При розробці та супроводі програмного забезпечення необхідний контроль та управління [1]. З цією метою проводять оцінки властивостей програмного забезпечення. Властивість – це деяка характеристика елемента програмного забезпечення [2].

Властивості програмного забезпечення дають його розуміння для аналізу та подальшого використання. З метою визначення кількісних показників властивості проводять вимірювання програмного забезпечення. Вимірювання проводиться за допомогою метрик [3]. Метрика – кількісне значення міри володіння системою, компонентом чи процесом заданою властивістю [2]. За допомогою метрик оцінюють властивості складових розробки програмного забезпечення (продуктів, процесів).

Проблема полягає в тому, що для розуміння програмного забезпечення та формулювання висновків керуються загальними властивостями, такими як «розуміння», «тестованість», «ефективність» і т.д. Для цих властивостей поки не було виведено єдиних метрик, які б їх оцінювали. Тому підбирають метрики програмного забезпечення, що доступні для вимірювання. Оскільки, найчастіше, властивості програмного забезпечення, які потребують визначення, є складними величинами, що не можна виміряти безпосередньо, то для характеристики властивості необхідно виконувати її декомпозицію на сукупність вузьких, підбираючи для неї ряд характеристичних метрик [4].

Отже, при підборі метричного забезпечення для властивості програмного забезпечення проводиться вибір та вимірювання деяких простих метрик, за допомогою яких оцінюється обрана властивість. Вирішенню задачі коректного підбору метрик присвячена дана стаття.

Огляд останніх досліджень

Існує три методи підбору метрик для властивостей програмного забезпечення: модель відповідальної особи [5] стандартизовані метрики [6] та ціль-питання-метрика [7].

Метод відповідальної особи передбачає підбір необхідних метрик безпосередньо відповідальною особою [5]. Підбір метрик в данному випадку пов'язується із задачами, які повинні бути вирішені, та рішеннями, які потрібно приймати при вирішенні даних задач. Рішення про вимірювання метрики приймаються на основі досвіду відповідальної особи або за рекомендаціями експертів.

Метод використання стандартизованих метрик пов'язаний з використанням рекомендацій чи стандартів деяких організацій,

які займаються дослідженнями в сфері розробки програмного забезпечення [6]. Цей метод базується на проведених емпіричних дослідженнях. Його використання обмежується сферою проведених досліджень, рекомендації по підбору метрик носять загальний характер.

Найбільш поширеним методом підбору метрик для властивостей програмного забезпечення вважається «ціль-питання-метрика» [5]. Цей метод базується на постановці цілей та запитань при підборі метричного забезпечення для властивості. Метод складається з трьох ключових етапів [7]:

- формулюються цілі, які повинні бути досягнуті шляхом оцінки властивості програмного забезпечення;
- формулюються питання для кожної цілі, на які потрібно отримати відповіді, для визначення чи досягнуті цілі;
- визначаються метрики, які можуть бути використані для відповідей на питання, які були поставлені на другому етапі.

Застосування методу «ціль-питання-метрика» включає в себе процеси планування, визначення цілей, питань та метрик, збір даних та аналіз даних (рис. 1).

Важливим аспектом при використанні цього методу є досвід та компетентність експерта [5]. В той самий час цей метод не є формалізованим, що дає певну долю ймовірності неадекватного підбору метрик. Крім цього, даний метод не дозволяє вибрати метричне забезпечення, які найкраще підходять для характеристики властивості.

Тому, для вирішення поставленої задачі пропонується формалізувати поставлену задачу за допомогою використання предметно-орієнтованого методу побудови залежностей між метриками програмного забезпечення [8].

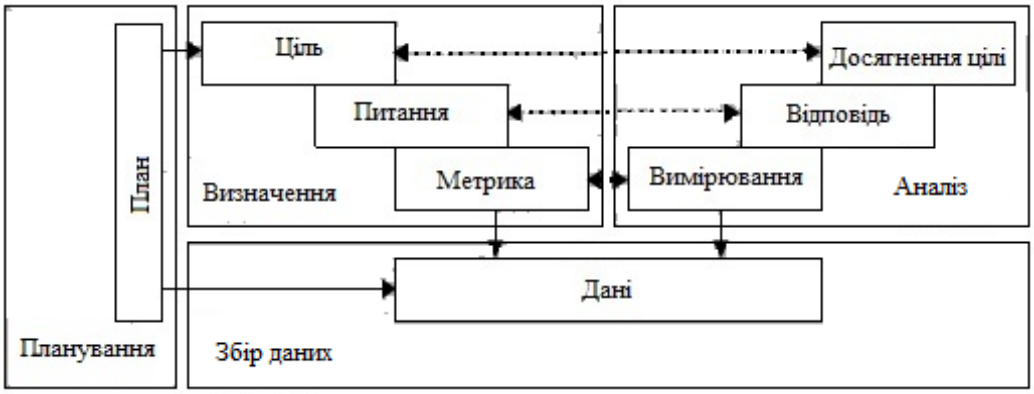


Рис.1 Метод «ціль-питання-метрика»

Цілі статті

Основними цілями статті є:

- формалізація методу «ціль-питання-метрика» шляхом застосування предметно-орієнтованого методу побудови залежностей між метриками програмного забезпечення;
- застосування формалізації методу «ціль-питання-метрика» для підбору метричного забезпечення властивості «розуміння»;
- можливості удосконалення формалізації.

Формалізований підбір та аналіз метрик для властивостей програмного забезпечення

Формалізація проводиться на етапі визначення метрик для поставленого питання та інтерпретації результатів вимірювань (рис 2).

Сутність формалізованого підбору та аналізу метрик для властивостей програмного забезпечення полягає в призначенні метрикам ваги, яка відображає зв'язок метрики з властивістю. Розрахунок ваги дозволить ранжувати метрики та відкинути найменш вагомі (рис. 3). Вага – це коефіцієнт, який визначає в якій мірі метрика оцінює властивість. Розрахунок ваги проводиться за допомогою методу побудови залежностей між метриками програмного забезпечення на етапі кореляційного аналізу [9].

Використання методу «ціль-питання-метрика» після додавання формалізації буде включати наступні етапи:

1. Постановка цілі.
2. Постановка питань.
3. Приблизний експертний підбір метрик для поставленого питання.
4. Вимірювання підібраних метрик для декількох проектів.
5. Експертне оцінювання поставлених запитань (цілей) та визначення їх мінімальних допустимих значень.
6. Зважування.
7. Ранжування та відсіяння метрик.

8. Побудова регресії, визначення мінімальних допустимих значень метрик та впливу метрик на властивість.

9. Виокремлення виміряних метрик для досліджуваного проекту.

10. Формулювання висновків про відхилення метрик від оптимальних значень.

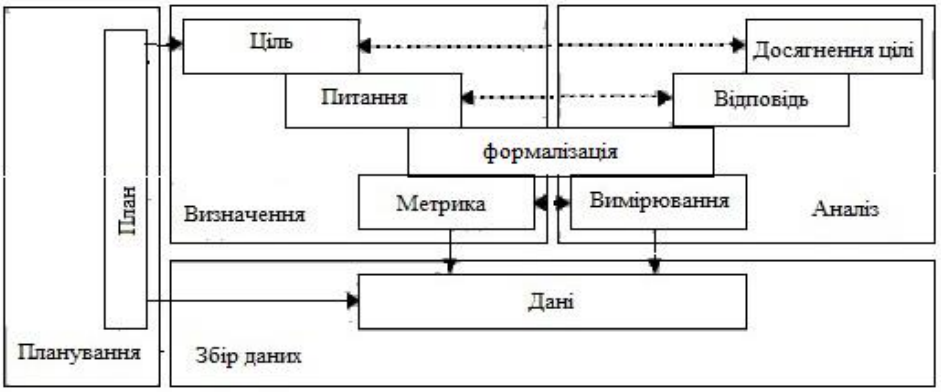


Рис.2 Формалізація в методі «ціль-питання-метрика»

Зважування проводиться шляхом розрахунку статистичних характеристик та коефіцієнтів кореляції. Тому з шостого кроку пропонується застосувати предметно-орієнтований метод побудови залежностей між метриками програмного забезпечення.

Рис.3 Ранжування

Предметно-орієнтований метод був розроблений для побудови залежностей між метриками програмного забезпечення при аналітичному виведенні функціональної залежності однієї метрики від іншої [8]. Реалізований метод в засібі [10], який використовується для формалізації підбору метрик та аналізу результатів вимірювань.

Сутність предметно-орієнтованого методу побудови залежностей між метриками програмного забезпечення полягає в тому, що побудова залежностей відбувається статистичними методами з врахуванням високої точності вимірювань програмного забезпечення без похибок та гіперболічною спадною залежністю між значеннями метрик та кількістю виміряних програм [8].

Перші три кроки проводяться як звичайно при використанні методу «ціль-питання-метрика».

Вимірювання метрик можна провести будь-яким засобом, який дозволяє отримати всі обрані метрики.

Попереднє отримання оцінок властивостей необхідне для визначення тенденцій у взаємозв'яках між метриками та властивостями. Тут важливо визначити, які метрики демонстрували гарні характеристики властивостей, а які – погані. Тому пропонується провести експертне оцінювання та визначення мінімальних допустимих значень властивостей для того ж програмного забезпечення, що вимірювалося [10]. Мінімально допустимі значення будуть необхідні при визначенні мінімально допустимих значень метрик.

Далі необхідно застосувати метод побудови залежностей між метриками таким чином. На першому етапі предметно-орієнтованого методу проводиться перевірка об'єму вхідних даних та розрахунок статистичних характеристик для метрик та експертних оцінок [8]. Важливою передумовою є великий об'єм вхідних даних. Тому для дослідження необхідно використати велику кількість програм.

Найважливішим етапом для формалізованого підбору метрик є другий [8]. На цьому етапі безпосередньо розраховується вагомість. Вага отримується як коефіцієнт кореляції, розрахований для метрики та експертної оцінки. Найбільше значення коефіцієнту серед вагомих метрик говорить про те, що ця метрика є найбільш вагомою для властивості. Крім цього, у випадку використання усіх доступних метрик, розраховується значущість, яка показує можливість застосування проведених розрахунків для підібраних проектів. У випадку незначущості багатьох метрик варто збільшити кількість досліджуваних проектів або провести їх якісний

аналіз. При попередній оцінці об'єму вибірки розрахунок значущості можна не проводити.

Після цього проводиться ранжування метрик по вагах. Відсіяння метрик відувається через визначення експертом границі відсіяння.

Шляхом побудови регресії на наступному етапі визначаються мінімальні допустимі значення метрик та тенденції у впливах метрик на властивість. Так як побудована регресія буде деякою функцією, то на основі її графіка можна говорити про стрімкість чи повільність зростання чи спадання властивості при зміні метрик.

Отримані графіки можна використовувати тільки як демонстрацію загальних закономірностей між метрикою та властивістю, але вони не показують чітку функціональну залежність. Це пов'язане з дослідженням метрик окремо одна від одної, а не всукупності.

Підбір метрик для властивості «розуміння» методом «ціль-питання-метрика»

Планування. Підбір метрик саме для властивості «розуміння» зумовлений широким колом її можливого використання. Її можна застосовувати при тестуванні, створенні документації, повторному використанні компонентів програмного забезпечення, рефакторингу, в зворотній інженерії [11]. Розуміння процесів та продуктів – це перший крок до більш ефективного їх контролю та управління.

Властивість «розуміння» показує наскільки просто чи складно зрозуміти написану програму. Це поняття включає декілька аспектів: зрозумілість програмного коду, що визначається зрозумілістю назв програмних конструкцій, простотою візуального сприйняття блоків (структурованості тексту програми), коментованістю програми, зрозумілість зв'язків у програмі (структурованості програми), зрозумілість компонентів програми та призначення програми. Необхідність включення до пояснення властивості «розуміння» призначення програми пояснюється складністю розуміння коду, написаного для вирішення вузькоспеціалізованих специфічних задач, зокрема для точних наук (наприклад, моделювання процесів при ядерних реакціях, чи деяких хімічних процесів), що потребує знань предметної області.

Вимірювання програмного забезпечення проводиться з ціллю забезпечення розуміння, контролю чи покращення програмного забезпечення [7]. Властивість «розуміння» є базовою для всіх трьох цілей розробки програмного забезпечення.

Визначення. При застосування методу «ціль-питання-метрика» для властивості «розуміння» виділяємо такі цілі: «оцінити зрозумілість програмного коду», «оцінити зрозумілість програми», що включає структурованість програми та зрозумілість компонентів, «оцінити спеціалізованість програми». Підбір метрик для всіх трьох цілей складає те метричне забезпечення, яке слід застосовувати для властивості «розуміння». Перша ціль дає можливість зрозумілість внутрішнього устрою окремих компонентів програми. Друга ціль дає можливість зрозуміти призначення компонентів програми та їх взаємозв'язок. Третя ціль допомагає зрозуміти складність програми, пов'язаної зі специфікою вирішення задач.

Постановка питань до цілі деталізує поставлену ціль для властивості, пов'язані з задачами, які вирішуються використанням властивості [11]. Постановка питань пов'язана з об'єктом дослідження: продукти, процеси, ресурси. Так як підбір метрик здійснюється для програмного забезпечення, відповідно питання для властивості ставляться тільки для програм. При цьому поставлені питання повинні тільки частково деталізувати цілі. Іншими словами, вини повинні знаходитися на рівень нижче від цілей та на рівень вище метрик. Тому питаннями до першої цілі є: «наскільки зрозумілі назви програмних конструкцій?», «наскільки добре структурований текст програми?», «наскільки коментована програма?»; питаннями до другої цілі є: «наскільки структурована програма?», «наскільки зрозумілі зв'язки компонентів?»; питанням до третьої цілі є «наскільки спеціалізовані задачі виконує програма?». Як видно із поставленого питання до третьої цілі, підібрати метрики для неї буде проблематично. В той самий час більш конкретне запитання поставити важко. Тому необхідно провести додатковий аналіз поставленої цілі на нижчому рівні.

Рівень метрик в методі «ціль-питання-метрики» відповідає рівню отримання кількісної інформації про досліджувані величини [11]. Метрики на нижчому прикладному рівні деталізують поставлені раніше питання. При підборі слід враховувати як прямі вимірювані метрики, так і вже виведені непрямі. Непрямі метрики дають більшу достовірність отриманих результатів.

Враховуючи особливості використовуваних вимірювачів, метриками для першої цілі є:

- питання «наскільки зрозумілі назви програмних конструкцій?»: «осмислені імена змінних у модулях», «осмислені імена модулів»

- питання «наскільки добре структурований текст програми?»: «модулі, що мають горизонтальні відступи», «модулі, що мають вертикальні відступи»
- питання «наскільки коментована програма?»: «модулі, які мають коментовані заголовки», «модулі, які мають коментарі між заголовком та кінцем», «число не коментованих конструкцій у модулі», «коментовані рядки модуля»

Метриками для другої цілі є:

- питання «наскільки структурована програма?»: «цикломатична складність», «структурна вкладеність», «метрики Хольстеда»
- питання «наскільки зрозумілі зв'язки компонентів?»: «зчеплення», «зв'язаність».

Метрики для третьої цілі, якій відповідає питання «наскільки спеціалізовані задачі виконує програма?», як було сказано раніше, підібрати досить проблематично. Визначити більшу специфічність можна по непрямих ознаках, таких як більше число коментарів в програмі та наявність спеціалізованої документації. При відсутності чи малому значенні цих ознак розуміння програми ускладнюється. Тому третя ціль на властивість розуміння буде впливати як деякий коефіцієнт ускладнення перших двох цілей. Чим більша спеціалізація програми, тим більше значення має коефіцієнт ускладнення. Даний коефіцієнт пропонується визначати експертам з предметної області, для якої написане програмне забезпечення.

Перевірка адекватності вибраних метрик поставленим цілям показує, що підібрані метрики для поставлених питань відповідають кожній із поставлених цілей. Виключенням є тільки остання ціль, для врахування якої пропонується використовувати коефіцієнт ускладнення. Загальна ієрархія цілей, питань та метрик приведена на рис. 4.

Збір даних. Даний етап пов'язаний із підготовкою матеріалів, до вимірювань. Цей етап може включати як фізичні вимірювання, так і проведення різного роду опитувань [11]. Тут вибирається вид, в якому будуть представлятися зібрані дані та засоби, які будуть для цього використовуватися.

Для проведення вимірювань вибрано більше 30 різних об'єктно-орієнтованих проектів з різною кількістю класів (від 100 до більше 1000). Вибір саме цих проектів пов'язаний із їх доступністю (вони відкриті) та широким розповсюдженням об'єктно-орієнтованого підходу при розробці програмного забезпечення. Основний засіб, який застосовувався для вимірювання об'єктно-орієнтованих проектів, – iPlasma [13]. Деякі прямі метрики були отримані за допомогою засобу, описаного в [14]. Об'єм досліджуваних даних обраний таким чином, щоб їх було достатньо для подальшого аналізу з використанням статистичних методів, деталі вирахування якого описані в [15].

Всі виміряні метрики були збережені до бази даних пакету статистичного аналізу для емпіричної інженерії програмного забезпечення, описаного в [10].

Аналіз. Аналіз є останнім етапом підбору метрик для властивості. Аналіз передбачає безпосередньо процес вимірювання та аналізу метрик, відповідей на питання та формулювання висновків про досягнення цілей. Етап аналізу даних проводиться на усіх трьох рівнях з рис.2 – вимірювання, відповідь та досягнення цілі – одночасно [11]. Висновки – кінцевий результат цього етапу. Цей етап включає в себе підготовку до отримання результатів, організацію процесів та безпосередньо отримання результатів та формулювання висновків на основі оброблених даних. В загальному цей метод передбачає експертне формулювання висновків на основі вимірювань [7]. Але для цього необхідні висококваліфіковані експерти, що не завжди є рентабельним для проекту. Тому саме на цьому етапі пропонується застосувати предметно-орієнтований метод побудови залежностей між метриками програмного забезпечення [8] для спрощення формулювання висновків за результатами вимірювань.

Вимірювання обраних метрик відбувалося за допомогою описаних вище засобів для вибраних проектів. Важливими даними при застосуванні запропонованого методу є експертні оцінки властивості «розуміння». Експертам пропонувалося оцінити запропоноване програмне забезпечення по десятибальній шкалі: один – у випадку цілком не зрозумілої програми, десять – у випадку повної зрозумілості. Було отримано більше 100 експертних оцінок експертів різного рівня кваліфікації та досвіду роботи. Дана кількість експертних оцінок, як показано в [15], є достатньою для вирішення поставленої задачі. Експертні оцінки, як і метрики, заносилися в базу даних та оброблялися засобом описаним в [10].

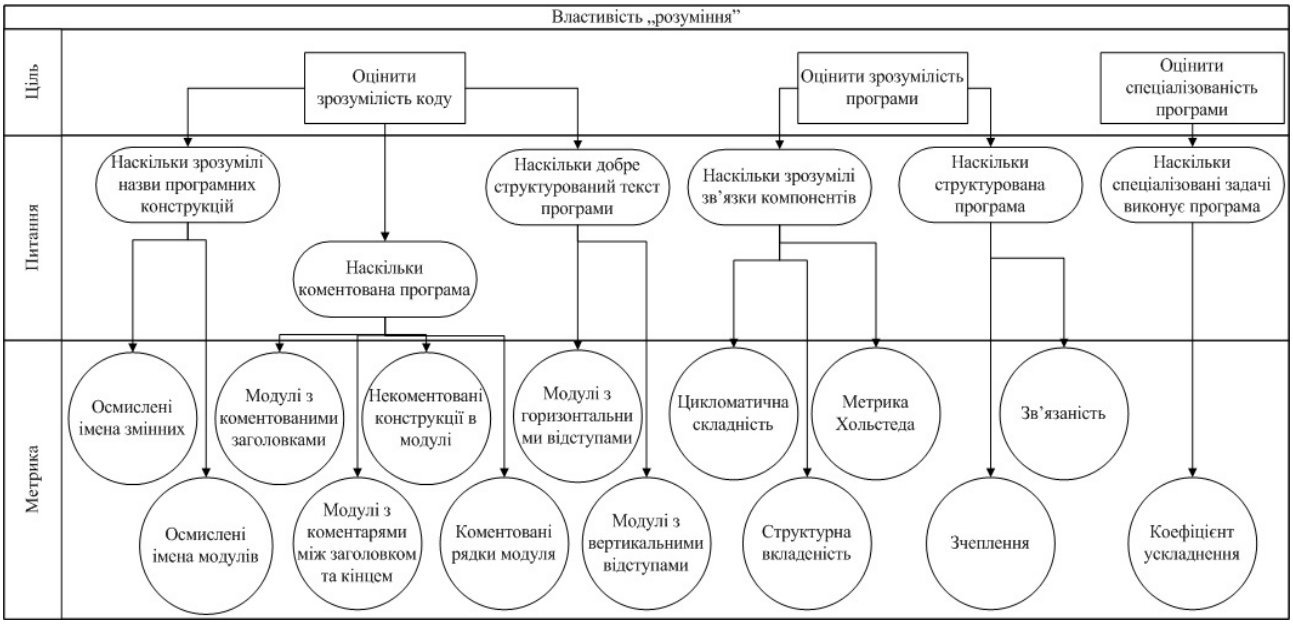


Рис.4 Структура підібраних цілей, питань та метрик для властивості «розуміння»

В результаті вимірювань був отриманий набір метрик для досліджуваних проектів. Наступним кроком є його аналіз та інтерпретація результатів. Для цього були отримані такі числові характеристики як математичне сподівання, середнє квадратичне відхилення, коефіцієнти асиметрії та ексцесу (табл. 1) з [16], на основі яких далі була визначена та підтверджена вагомість метрик для властивості через розрахунок коефіцієнтів кореляції та їх вагомості.

Табл. 1.

Початковий та центральний моменти метрик

Познач. метрики	**Мат. сподівання**	**Середнє квадрат. відхилення**	**Коеф. Асиметрії**	**Коеф. ексцесу**	**Пояснення метрики**
AMVM	48,63	53,8	2,36985	7,3523	Осмислені імена змінних
PMI	40,46	28,3	0,16160	-0,9208	Осмислені імена модулів
MH	86,86	31,8	-2,21767	3,2667	Модулі з коментованими заголовками
MC	27,60	39,5	1,03008	-0,6417	Модулі з коментарями між заголовком та кінцем
ANCSS	51,74	45,2	-0,06078	-1,8309	Некоментовані конструкції в модулі
APCM	16,95	20,7	1,27999	1,0080	Коментовані рядки модуля
MHS	26,84	39,6	1,06589	-0,6012	Модулі з горизонтальними відступами
MVS	74,32	37,3	-1,11969	-0,3388	Модулі з вертикальними відступами
CYCLO	87658,18	158076,0	3,06018	9,7879	Цикломатична складність
MAXNESTING	1,02	1,4	7,54477	129,4601	Структурна вкладеність
THE	0,64	1,1	5,72411	127,9911	Метрика Хольстеда (загальні зусилля по Хольстеду)
TCC	0,32	0,4	0,87421	-1,0284	Зчеплення
CINT	0,70	2,0	4,72266	30,8268	Зв’язаність
‘RA’	6,65	1,8	-0,29354	-0,9400	«розуміння»

Як видно з табл. 1, жодна з метрик та експертні оцінки не мають нормального розподілу. Це ще раз підтверджує виведену раніше в [8] закономірність відсутності нормального розподілу серед метрик.

Тому на наступному кроці аналізу була проведена парна рангова кореляція метрик та експертних оцінок з розрахунку коефіцієнтів Спірмена (1) [17]:

$$\hat{\tau}_c = 1 - \frac{6}{n(n^2-1)} \sum_{i=1}^{n} d_i \qquad (1)$$

де $\hat{\tau}_c$ - коефіцієнт Спірмена;

$d_i = r_{xi} - r_{yi}$;

r_{xi} та r_{yi} – порядкові номери варіант у варіаційних рядах за x та y.

Дані розрахунки необхідні для визначення вагомості кожної метрики при оцінці властивості «розуміння». Цією вагомісю і є кофіцієнт кореляції, який в данному випадку показує ступінь значимості даної метрики для властивості. Розраховані коефіцієнти приведені в табл. 2.

Табл. 2.

Коефіцієнти кореляції для властивості «розуміння» та прямих метрик

Назва метрики	Коефіцієнт кореляції Спірмена	Значущість	Коефіцієнт кореляції Спірмена (після розширення даних)	Значущість (після розширення даних)
AMVM	0,179806	+	0,179806	+
PMI	0,120730	+	0,120730	+
MH	0,238229	+	0,238229	+
MC	0,335050	-	0,259781	+
ANCSS	0,138274	+	0,138274	+
APCM	0,232238	+	0,232238	+
MHS	0,292728	-	0,171866	+
MVS	0,274440	+	0,274440	+
CYCLO	0,167060	+	0,167060	+
MAXNESTING	-0,059985	+	-0,059985	+
THE	-0,146379	+	-0,146379	+
TCC	-0,091219	+	-0,091219	+
CINT	-0,674901	-	-0,364582	+

Проте самі по собі коефіцієнти кореляції не повною мірою демонструють вагомість. Тому, крім коефіцієнтів кореляції, була визначена значущість отриманих коефіцієнтів. Вона показує чи є достатнім об'єм отриманих результатів для того, щоб говорити про вагомість метрики для властивості. Перевірка значущості проводилася з використанням статистичної характеристики t (2) [18]:

$$t = \frac{\hat{\tau}_c \sqrt{n-2}}{\sqrt{1-\hat{\tau}_c^2}} \qquad (2)$$

Якщо $|t| > t_{\alpha/2}$ (де $t_{\alpha/2}$ було отримано з таблиці розподілів t-Стьюдента), то робився висновок, що метрика значуща; якщо умова не виконувалася, то робився висновок, що метрика не значуща.

Як видно з табл. 2 (колонки 2-3), не є значущість метрики MC, MHS, CINT. Це говорить про необхідність збільшення об'єму дослідження даних для більш точних результатів. Після збільшення об'єму даних для цих метрик коефіцієнти стали значущими (табл.2, колонки 4-5). отримані значення коефіцієнтів кореляції говорять про те, що найбільш вагомою метрикою для властивості «розуміння» є метрика «зв'язаність», наступними за нею йдуть метрики «модулі з коментарями між заголовком та кінцем», «модулі з горизонтальними відступами», «модулі з вертикальними відстанями».

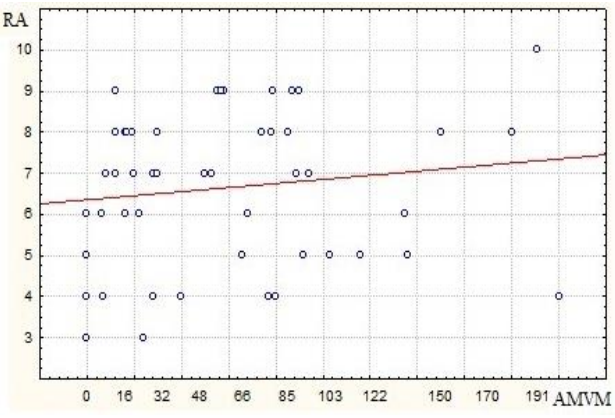


Рис.5 Регресія метрики «осмислені імена змінних» та властивості «розуміння»

Для аналізу закономірностей впливів метрик на властивість «розуміння» були побудовані регресії, де по одній осі відкладалися оцінки, а по іншій - метрики. Регресії будувалися на основі кореляційних полів шляхом побудови регресії та підбору найточнішої лінії. Вигляд регресії для кожної метрики зводиться до приведених на рис. 5-8.

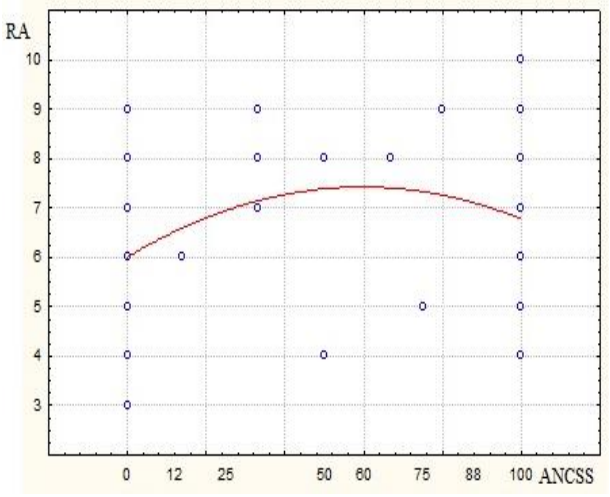


Рис.6 Регресія метрики «не коментовані конструкції в модулях» та властивості «розуміння»

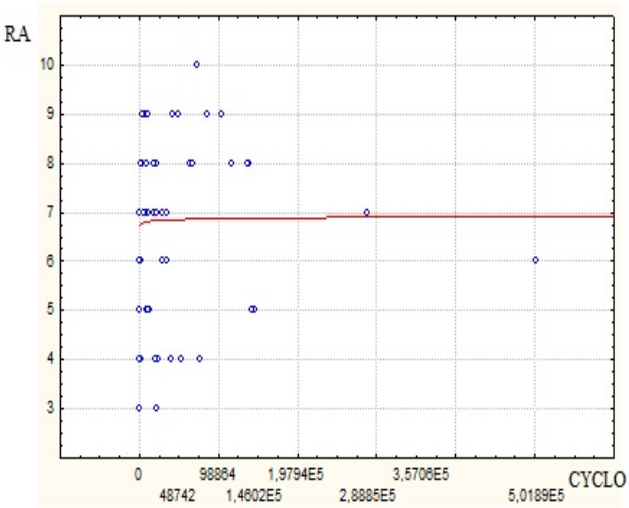


Рис.7 Регресія метрики «цикломатична складність» та властивості «розуміння»

Як видно з приведених графіків, при зростанні деяких метрик «зрозумілість» покращується, для деяких – практично не змінюється, для деяких погіршується. Закономірність, приведена на рис.5 присутня тільки для метрики «осмислені імена змінних», що говорить про покращення розуміння при збільшенні числа осмислених змінних. Закономірність, приведена на рис.6 присутня у такої групи метрик: «не коментовані конструкції в модулях» - говорить про необхідність коментувати не всі конструкції, а тільки важливі; «модулі з коментарями між заголовком та кінцем», «коментовані рядки модуля» - регресія схожа до попередньої метрики, що підтверджує зроблені висновки про необхідність коментування тільки важливих програмних конструкцій; «осмислені імена модулів», «модулі з коментованими заголовками», «модулі з горизонтальними відступами», «модулі з вертикальними відступами» - мають більш струмко зрозстаючі закономірності, що говорить про покращення зрозумілості при збільшенні числа осмислених імен модулів, коментарів до модулів та відступів. Вид лінії регресії з рис.7 присутній тільки у метрики «цикломатична складність» - говорить про те, що цикломатична складність практично не впливає на розуміння програм.

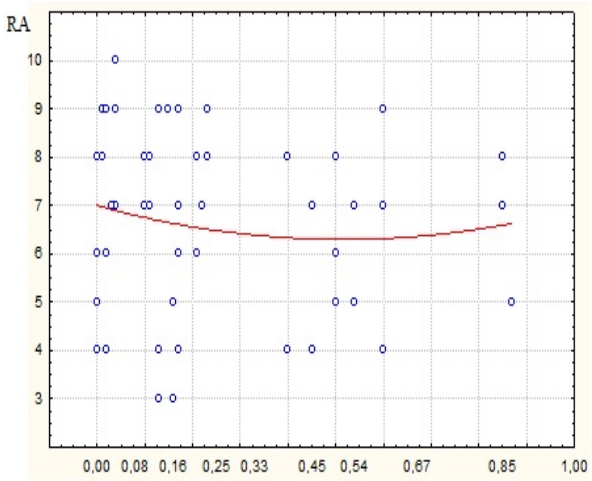


Рис.8 Регресія метрики «зчеплення» та властивості «розуміння»

Регресія,приведена на рис. 8 є у метрик «зчеплення», «структурна вкладеність», «зусилля по Хольстеду», «зв’язаність» - мають загальну тенденцію спадання, що говорить про погіршення розуміння при збільшенні цих метрик.

Іншими словами для кращого розуміння програми, вона повинна бути добре коментована, але без надмірності, та більш структурована. Хоча, якщо говорити в загальному, вагомість метрик не є дуже високою, що говорить про те, що скоріше всього, існують ще інші метрики, які є також вагомими для даної властивості з ще меншими значеннями вагомості. Тому існує потреба повторення даного дослідження з більшим числом метрик, а також детального дослідження метрики «цикломатична складність», так як коефіцієнт кореляції показує, що дана метрика впливає на властивість, а лінія регресії показує, що розуміння практично не змінюється.

Щодо коефіцієнту ускладнення, введеного раніше, його значення варто прийняти за 1. Причиною цього є те, що досліджуване програмне забезпечення розроблялося для використання розробниками програмного забезпечення, і використані алгоритми є цілком відомі для дослідників.

Формалізований підбір метрик

Як було сказано раніше, використання методу «ціль-питання-метрика» обмежується наявністю висококваліфікованих експертів та рентабельністю проекту. Тому альтернативою йому може слугувати застосування формалізованого підбору метрик для програмного забезпечення на базі предметно-орієнтованого методу побудови залежностей з самого початку дослідження [9]. Особливістю його застосування є підбір великої кількості метрик на етапах планування дослідження та визначення метрик для властивості. Для цього можна використати підбір метрик експертом не досить високої кваліфікації, запропонувавши йому відкинути з великого набору метрик для вимірювання ті, які точно не характеризують досліджувану властивість. А для інших – використати кореляційний аналіз. Метрики, які не характеризують властивість, будуть відкинуті за результатами дуже малої ваги [9] для властивості. Ті метрики, що не відкинулися, залишається ранжувати та виділити серед них найбільш вагомі.

Для дослідження закономірностей та визначення оптимальних значень метрик пропонується побудувати регресії. Вони дозволяють отримати графічне представлення отриманих даних.

Висновки

Отже, в результаті використання методу «ціль-питання-метрика» та предметно-орієнтованого методу побудови залежностей між метриками програмного забезпечення був виконаний підбір метрик для властивості «розуміння».

Висновки виконаної роботи:

- не завжди легко підібрати метрики для властивості «розуміння» методом «ціль-питання-метрика» через наявність великої кількості метрик;
- при аналізі даних експертами без допоміжних засобів оцінка властивості відбувається на основі невеликої кількості більш поширених чи відомих метрик, не враховуючи або слабо враховуючи інші метрики;
- при аналізі даних експертами без допоміжних засобів оцінка властивості з інтуїтивною градацією по вагомості;
- для підвищення точності підбору метрик властивості «розуміння» необхідно було провести формалізацію аналізу;
- формалізація аналізу дала градацію метрик по їх вагомості та необхідність уточнення деяких даних;
- на розуміння програми найбільше впливає структурованість та коментованість;
- виявлено, що надмірна коментованість ускладнює розуміння, як і сильне зчеплення та зв’язаність, а цикломатична складність практично не змінює розуміння;
- слабка вагомість багатьох метрик говорить про існування інших слабо вагомих метрик для властивості «розуміння», які не були включені експертом при підборі метрик методом «ціль-питання-метрика».

Тому можна узагальнити, що для проведення більш точних розрахунків з використанням менш кваліфікованих експертів необхідно доповнити неформалізований метод «ціль-питання-метрика» формалізацією. Вона полягає у використанні предметно-орієнтованого методу побудови залежностей між метриками програмного забезпечення для попередньої оцінки вибраних метрик та розрахунку їх вагомості на етапі аналізу результатів вимірювань.

Література

1. *Norman E. Fenton, Shari Lawrence Pfleeger* Software Metrics: A Rigorous and Practical Approach.- Cambridge University Press,1996.-638p.

2. IEEE Std 610.12-1990, IEEE *Standard Glossary of Software Engineering Terminology,* identifies terms currently in use in the field of Software Engineering. Standard definitions for those terms are established. – The Institute of Electrical and Electronics Engineering, New York, 1990.

3. *Forrest Shull, Janice Singer, Dag I.K. Sjoberg* Guide to Advanced Empirical Software Engineering. – Springer-Verlag London Limited 2008.-394p.

4. *Дишлевий О.П.* Перевірка адекватності метричних моделей властивостей програмного забезпечення // ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 2006: Матеріали Всеукраїнської конференції аспірантів та студентів.- К.:НАУ, 2007.- с.77-84.

5. *J. McGary, D. Card, C. Jones, B. Layman, W. Clark, J. Dean, and F. Hall.* Practical Software Measurement, Objective Information for Decision Makers, Addison-Wesley, Boston, 2002.-294p.

6. *Linda M. Laird, M. Carol Brennan* Software Measurement and Estimation: a practical approach. John Wiley & Sons, Inc., Hoboken, New Jersey 2006.- 257 p.

7. *Basili V.R., Weiss D.M.* A method for collection valid software engineering data // IEEE Transaction on Software Engineering, 10(6), pp. 728-38, 1984.

8. *Дишлевий О.П.* Предметно-орієнтований метод побудови залежностей між метриками програмного забезпечення / О.П. Дишлевий // Вісник НАУ. – 2009. – №3. – С. 206–212.

9. *Дишлевий О.П.* Підбір метрик для властивостей програмного забезпечення / О.П. Дишлевий // ПРОБЛЕМИ ПРОГРАМУВАННЯ. Науковий журнал. – 2010. – №2-3. – С. 237–242.

10. *Дишлевий О.П.* Пакет статистичного аналізу для емпіричної інженерії програмного забезпечення / О.П. Дишлевий // Наука і молодь. Прикладна серія. Збірник наукових праць. – 2009. – № 9. – С. 104–108.

11. *Хоменко В.А., Дышлевый А.П.* Метод экспертного оценивания свойств повторно используемых компонентов программного обеспечения // Том 1: Матеріали VIII Міжнародної науково-технічної конференції «Авіа-2007».- Т.1.- К.:НАУ, 2007, с. 13.169-13.172.

12. Rini van Solinger, Egon Berghout The Goal/Question/Metric Method: a practical guide for quality improvement of software development. – McGraw-Hill International (UK) Limited 1999. – 200p.

13. *Michele Lanza, Radu Marinescu.* Obect-oriented metrics in practice: Using software metrics to characterize, evaluate, and improve the design of object-oriented systems. – Springer- Verlag Berlin Limited 2006.-205p.

14. *Сидоров Н.А., Хоменко В.А.* Структура измерителя программ. // Проблеми транспорту: Зб. наук. пр. Випуск 2. – К: НТУ, 2005. - С. 190-195.

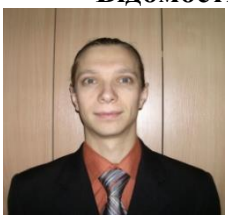
15. *Дишлевий О.П.* Застосування предметно-орієнтованого методу побудови залежностей між метриками програмного забезпечення / О.П. Дишлевий // Вісник НТУУ «КПІ». Інфоматика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2009. – №50. – С. 64–69.

16. *Вентцель Е.С.* Теория вероятностей: Учеб. для вузов. – 7-е изд. стер. – М.: Высш. шк., 2001. – 575 с.: ил.

17. *М. Кендалл, А. Стюарт,* Статистические выводы и связи. – Главная редакция физико-математической литературы из-ва «Наука», 1973. – 899 с.

18. *Айвазян С.А. и др.* Прикладная статистика: Исследование зависимостей: Справ. изд. / С.А. Айвазян, И.С. Енюков, Л.Д. Мешалкин; под. ред. С.А. Айвазяна. – М.:Финансы и статистика, 1985. – 487 с.

Відомості про автора:



Дишлевий Олексій Петрович – старший викладач кафедри інженерії програмного забезпечення факультету комп'ютерних наук. Закінчив Національний авіаційний університет в 2006р., спеціальність «Програмне забезпечення автоматизованих систем». Науковий напрям – емпірична інженерія програмного забезпечення
E-mail: oleksiy.dyshlevyy@livenau.net
тел. 8 050 743 76 74.

