ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

УДК 004.415.2.045 (076.5)

[1]**М.Г. Луцький**, к.т.н., проф.
[2]**О.П. Дишлевий**, ст. викл.

МЕТРИЧНЕ ЗАБЕЗПЕЧЕННЯ ВЛАСТИВОСТІ «РОЗУМІННЯ» ПРОГРАМ

Національний авіаційний університет
[1]E-mail: lutskyi.maksym@rada.gov.ua
[2]E-mail: oleksiy.dyshlevyy@livenau.net

Розглянуто підбір метрик для властивості «розуміння» програм. Проведено аналіз властивості «розуміння» та підібрано метрики методом «ціль–питання–метрика». Збір даних виконано вимірюванням підібраних метрик спеціальними засобами. Аналіз даних проведено предметно-орієнтованим методом побудови залежностей між метриками програмного забезпечення. Підтверджено метричне забезпечення властивості «розуміння». Розраховано вагомість кожної метрики для властивості.

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.

Рассмотрен подбор метрик для свойства «понимаемость» программ. Проведен анализ свойства «понимаемость» и подобраны метрики методом «цель–вопрос–метрика». Сбор данных проведен измерением подобранных метрик специальными средствами. Анализ данных осуществлен предметно-ориентированным методом построения зависимостей между метриками программного обеспечения. Подтверждено метрическое обеспечение свойства «понимаемость». Рассчитана весомость каждой метрики для этого свойства.

Вступ

У багатьох процесах, пов'язаних зі створенням та супроводженням програмного забезпечення, проводиться вимірювання [1].

Вимірювання програмного забезпечення застосовують, наприклад, при документуванні, модифікації, контролю якості, контролю та керування процесами розроблення для підвищення їх ефективності [2]. Вимірювання проводиться за допомогою метрик [3].

Метрика – кількісне значення міри володіння системою, компонентом чи процесом заданої властивості [4].

За допомогою метрик оцінюють властивості складових розроблення програмного забезпечення (продуктів, процесів).

Властивість – це деяка характеристика сутності програмного забезпечення [4].

Властивості програмного забезпечення дають його розуміння для аналізу та подальшого використання.

Вищий рівень абстракції для властивості відповідає повному розумінню програмного забезпечення, нижчий – показує окремий аспект програмного забезпечення.

Властивості нижчого рівня простіше охарактеризувати на основі простих числових показників. Але, оскільки це нижчий рівень абстракції, то на основі цих властивостей важко або неможливо оцінити загальні характеристики програмного забезпечення.

Найчастіше властивості програмного забезпечення, які потребують визначення, є складними величинами, які не можна виміряти безпосередньо, тому для характеристики властивості необхідно виконувати її декомпозицію на сукупність вузьких, підбираючи для неї ряд характеристичних метрик [5].

У разі підбору метричного забезпечення для властивості програмного забезпечення проводиться вибір та вимірювання деяких простих метрик, за допомогою яких характеризується обрана властивість.

Аналіз останніх досліджень

Одним із найпоширеніших методів підбору метрик для властивостей програмного забезпечення вважається «ціль–питання–метрика» (GQM) [6].

Цей метод базується на постановці цілей та запитань при підборі метричного забезпечення для властивості.

Метод складається з трьох ключових етапів [7]:

– формулювання цілей, які повинні бути досягнуті оцінкою властивості програмного забезпечення;

– формулювання питань для кожної цілі, на які потрібно отримати відповіді, для визначення чи досягнення цілі;

– визначення метрик, 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 у разі використання цього методу є досвід та компетентність експерта [6].

У той самий час цей метод не є формалізованим, що дає певну долю ймовірності неадекватного підбору метрик. Крім цього, метод не дозволяє вибрати метричні забезпечення, які найкраще підходять для характеристики властивості.

Для вирішення поставленої задачі пропонується її формалізувати за допомогою використання предметно-орієнтованого методу побудови залежностей між метриками програмного забезпечення [8].

Цілі статті:

– застосування методу «ціль–питання–метрика» для підбору метричного забезпечення властивості «розуміння»;

– застосування предметно-орієнтованого методу побудови залежностей між метриками програмного забезпечення для альтернативного підбору метричного забезпечення властивості «розуміння» та ранжування підібраних метрик за ступенем вагомості метрик для властивості;

– доведення можливості окремого застосування предметно-орієнтованого методу для підбору метрик та більшої ефективності їх спільного застосування.

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

Підбір метрик для властивості «розуміння» зумовлений широким її використанням. Метрику можна застосовувати при тестуванні, створенні документації, повторному використанні компонентів програмного забезпечення, рефакторингу, в зворотній інженерії [9].

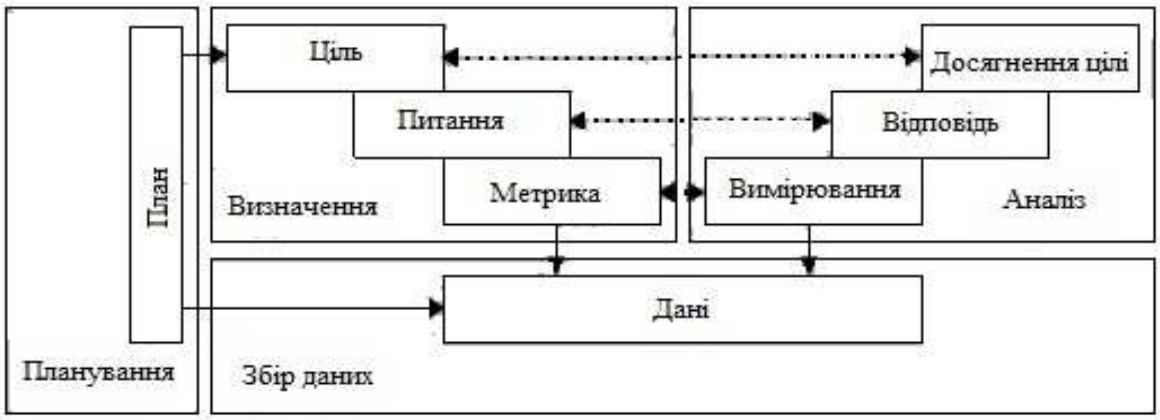

Рис. 1. Метод «ціль–питання–метрика»

Розуміння процесів та продуктів – це перший крок до більш ефективного їх контролю та керування.

Властивість «розуміння» показує, наскільки просто чи складно зрозуміти написану програму. Це поняття включає декілька аспектів:

– зрозумілість програмного коду, що визначається зрозумілістю назв програмних конструкцій;

– простоту візуального сприйняття блоків (структурованості тексту програми);

– коментованість програми;

– зрозумілість зв'язків у програмі (структурованості програми);

– зрозумілість компонентів програми;

– призначення програми.

Необхідність включення до пояснення властивості «розуміння» призначення програми пояснюється складністю розуміння коду, написаного для вирішення вузькоспеціалізованих специфічних задач, зокрема для точних наук (моделювання процесів під час ядерних реакцій, деяких хімічних процесів), що потребує знань предметної області.

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

Під час застосування методу «ціль–питання–метрика» для властивості «розуміння» виділяємо такі цілі:

– оцінювання зрозумілості програмного коду;

– оцінювання зрозумілості програми, що включає структурованість програми, та компонентів;

– оцінювання спеціалізованості програми.

Підбір метрик для всіх трьох цілей складає те метричне забезпечення, яке слід застосовувати для властивості «розуміння».

Перша ціль дає можливість зрозуміти внутрішній устрій окремих компонентів програми, друга – призначення компонентів програми та їх взаємозв'язок, третя – складність програми, пов'язаної зі специфікою вирішення задач.

Постановка питань до цілі деталізує поставлену ціль для властивості, пов'язаної з задачами, які вирішуються використанням властивості [9].

Постановка питань пов'язана з об'єктом дослідження: продуктами, процесами, ресурсами.

Оскільки підбір метрик здійснюється для програмного забезпечення, питання для властивості ставляться тільки для програм.

При цьому поставлені питання повинні тільки частково деталізувати цілі, тобто знаходитися на рівень нижче від цілей та на рівень вище метрик. Тому питаннями до першої цілі є такі:

– наскільки зрозумілі назви програмних конструкцій;

– наскільки добре структурований текст програми;

– наскільки коментована програма.

Питаннями до другої цілі є такі:

– наскільки структурована програма;

– наскільки зрозумілі зв'язки компонентів.

Питання до третьої цілі таке: наскільки спеціалізовані задачі виконує програма. Із питання до третьої цілі видно, що підібрати метрики для неї буде складно. Більш конкретне запитання поставити важко. Тому необхідно провести додатковий аналіз поставленої цілі на нижчому рівні.

Рівень метрик у методі «ціль–питання–метрики» відповідає рівню отримання кількісної інформації про досліджувані величини [9].

Метрики на нижчому прикладному рівні деталізують поставлені раніше питання. Під час підбору слід враховувати прямі вимірювані метрики та вже виведені непрямі.

Непрямі метрики дають більшу достовірність отриманих результатів.

Ураховуючи особливості використовуваних вимірювачів, метриками для першої цілі є:

1) наскільки зрозумілі назви програмних конструкцій:

– осмислені імена змінних у модулях;

– осмислені імена модулів;

2) наскільки добре структурований текст програми:

– модулі, що мають горизонтальні відступи;

– модулі, що мають вертикальні відступи;

3) наскільки коментована програма:

– модулі, які мають коментовані заголовки;

– модулі, які мають коментарі між заголовком та кінцем;

– число некоментованих конструкцій у модулі;

– коментовані рядки модуля.

Метриками для другої цілі є питання:

1) наскільки структурована програма:

– цикломатична складність;

– структурна вкладеність;

– метрики Хольстеда;

2) наскільки зрозумілі зв'язки компонентів:

– зчеплення;

– зв'язаність.

Для третьої цілі визначити більшу специфічність можна за непрямими ознаками: більшим числом коментарів у програмі та наявністю спеціалізованої документації.

У разі 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. Коефіцієнт пропонується визначати експертам із 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. Загальну ієрархію цілей, питань та метрик показано на рис. 2.

Етап збору даних пов'язаний із підготовкою матеріалів до вимірювань. Цей етап може включати як фізичні вимірювання, так і проведення різного роду опитувань [9], де вибирається вид зібраних даних 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 [10].

Деякі прямі метрики були отримані за допомогою засобу, описаного в роботі [11].

Об'єм досліджуваних даних обраний таким чином, щоб їх було достатньо для подальшого аналізу з використанням статистичних методів, деталі вирахування якого описані в роботі [12].

Усі виміряні метрики збережено в базі даних пакету статистичного аналізу для емпіричної інженерії програмного забезпечення, описаного в роботі [12].

Аналіз відповідей на питання є останнім етапом підбору метрик для властивості.

Аналіз передбачає безпосередньо процес вимірювання метрик та формулювання висновків про досягнення цілей.

Аналіз даних проводиться на всіх трьох рівнях вимірювання, відповідь та досягнення цілі – одночасно [9].

Висновки – кінцевий результат етапу, який включає в себе:

– підготовку до отримання результатів;

– організацію процесів;

– безпосередньо отримання результатів;

– формулювання висновків на основі оброблених даних;

– експертне формулювання висновків на основі вимірювань [7].

Але для формулювання висновків за результатами вимірювань необхідні висококваліфіковані експерти, що не завжди є рентабельно для проекту, тому на цьому етапі пропонується застосувати предметно-орієнтований метод побудови залежностей між метриками програмного забезпечення для спрощення [8].

Вимірювання обраних метрик відбувалося за допомогою описаних засобів для вибраних проектів.

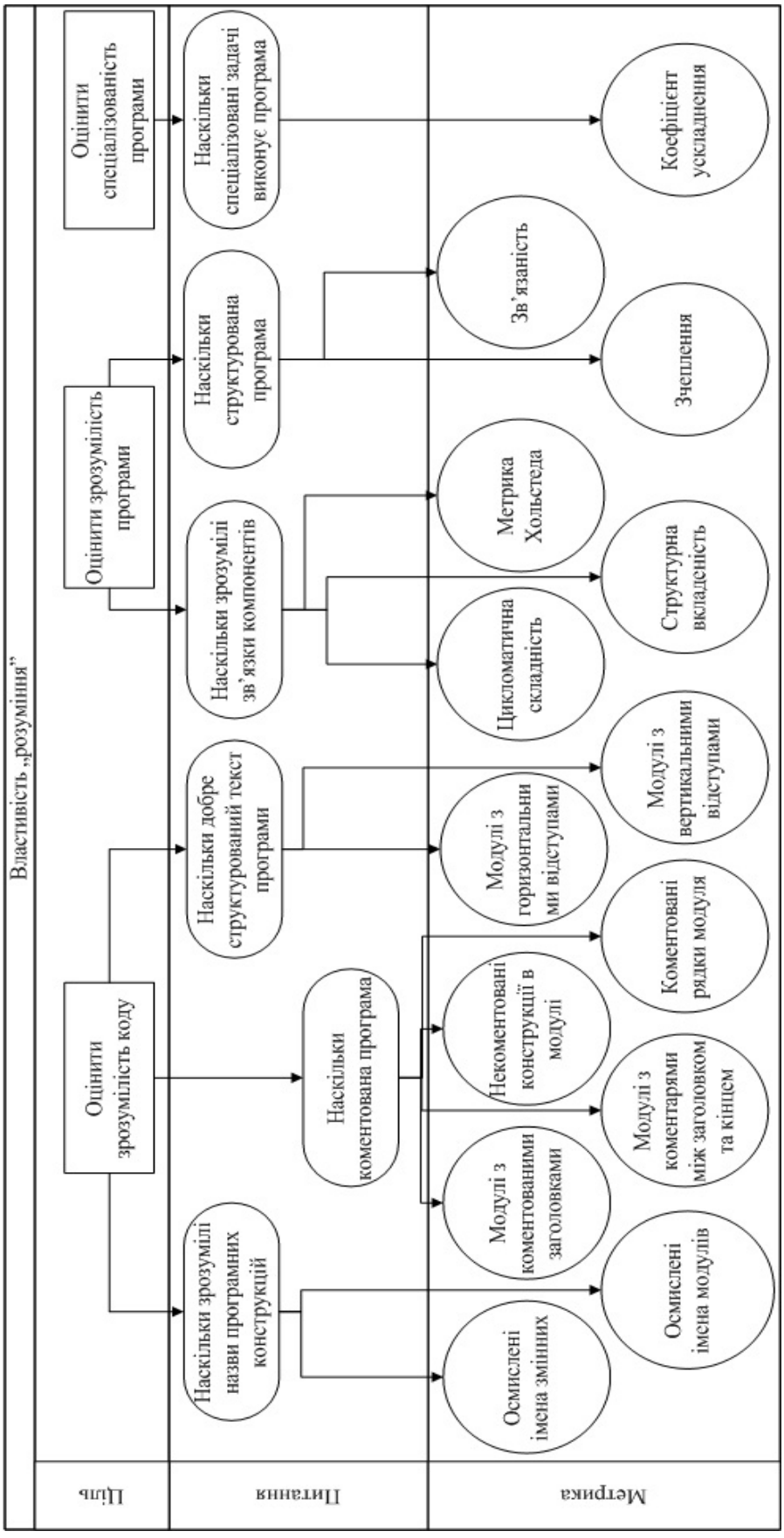

Рис. 2. Структура підібраних цілей, питань та метрик для властивості «розуміння»

Важливими даними у разі застосування запропонованого методу є експертні оцінки властивості «розуміння».

Експертам пропонувалося оцінити запропоноване програмне забезпечення за десятибальною шкалою:

– один у випадку цілком незрозумілої програми;

– десять у випадку повної зрозумілості.

Було отримано більше 100 експертних оцінок експертів різного рівня кваліфікації та досвіду роботи, що достатньо для вирішення даного завдання [13]. Експертні оцінки, як і метрики, заносилися в базу даних та оброблялися засобом, описаним у роботі [12].

У 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, середньоквадратичне відхилення, коефіцієнти асиметрії та ексцесу (табл. 1), на основі яких визначено та підтверджено вагомість метрик для властивості через розрахунок коефіцієнтів кореляції та їх вагомості [14].

Як видно з табл. 1, жодна з метрик та експертні оцінки не мають нормального розподілу. Це ще раз підтверджує виведену в роботі [8] закономірність відсутності нормального розподілу серед метрик. Тому на наступному кроці аналізу проведено парну рангову кореляцію метрик та експертних оцінок із розрахунку коефіцієнтів Спірмена [15]:

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

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

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

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

Ці розрахунки необхідні для визначення вагомості кожної метрики за оцінювання властивості «розуміння». Цією вагомістю і є коефіцієнт кореляції, який у цьому випадку показує ступінь значущості метрики для властивості. Розраховані коефіцієнти наведено в табл. 2.

Проте самі по собі коефіцієнти кореляції не повною мірою демонструють вагомість. Тому, крім коефіцієнтів кореляції, визначено значущість отриманих коефіцієнтів, яка показує, чи є достатнім об'єм отриманих результатів для визначення вагомості метрики для властивості.

Перевірка значущості проводилася з використанням статистичної характеристики t [16]:

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

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

Як видно з табл. 2, метрики MC, MHS, CINT не є значущими, тобто потрібно збільшення об'єму дослідження даних для більш точних результатів.

Після збільшення об'єму даних для цих метрик коефіцієнти стали значущими.

Отримані значення коефіцієнтів кореляції показують, 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».

Для кращого розуміння програма повинна бути більш коментована та більш структурована. Не дуже висока вагомість метрик свідчить про те, що скоріше всього існують інші метрики, які є також вагомими для цієї властивості з ще меншими значеннями вагомості. Важко виділити метрику, яка б не мала ніякого відношення до розуміння програми.

Щодо коефіцієнта ускладнення, введеного раніше, його значення варто брати за 1. Причиною цього є те, що досліджуване програмне забезпечення розроблялося для використання і алгоритми відомі для дослідників.

Формалізований підбір метрик

Використання методу «ціль–питання–метрика» обмежує наявність висококваліфікованих експертів та рентабельність проекту.

Таблиця 1

Початковий та центральний моменти метрик

Позначення метрики	Математичне сподівання	Середньоквадратичне відхилення	Коефіцієнт асиметрії	Коефіцієнт 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»

Таблиця 2

Коефіцієнти кореляції для властивості «розуміння» та прямих метрик

Назва метрики	Коефіцієнт кореляції Спірмена до розширення даних	Значущість до розширення даних	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	+

Тому альтернативою йому може слугувати застосування формалізованого підбору метрик для програмного забезпечення на базі предметно-орієнтованого методу побудови залежностей із самого початку дослідження [17].

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 [17] для властивості. Ті метрики, що не відкинулися, залишається ранжувати та виділити серед них найбільш вагомі.

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

Висновки

1. Не завжди легко підібрати метрики для властивості «розуміння» методом «ціль–питання–метрика» через наявність великої кількості метрик.

2. Під час аналізу даних експертами без допоміжних засобів оцінка властивості відбувається на основі невеликої кількості більш поширених чи відомих метрик, не враховуючи або слабо враховуючи інші метрики.

3. Під час аналізу даних експертами без допоміжних засобів властивості оцінюють з інтуїтивною градацією за вагомістю.

4. Для підвищення точності підбору метрик властивості «розуміння» необхідно провести формалізацію аналізу.

5. Формалізація аналізу дала градацію метрик за їх вагомістю та необхідність уточнення деяких даних.

6. На розуміння програми найбільше впливає структурованість та коментованість.

7. Слабка вагомість багатьох метрик свідчить про існування інших слабо вагомих метрик для властивості «розуміння», які не були включені експертом під час підбору метрик методом «ціль-питання-метрика».

8. Для проведення більш точних розрахунків з використанням менш кваліфікованих експертів необхідно доповнити неформалізований метод «ціль–питання–метрика» деякою формалізацією. Для цього пропонується використати предметно-орієнтований метод побудови залежностей між метриками, який був апробований у цій роботі та в роботі [17].

Підбір метрик був застосований під час створення інформаційної технології супроводження експлуатації авіаційної техніки [18–20].

Література

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

2. Best Practices in Software Measurement: How to use metrics to improve project and process performance / Christof Ebert, Reiner Dumke, Manfred Bundschuh, Andreas Schmietendorf. – Springer-Verlag Berlin Heidelberg, 2005. – 295 p.

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

4. *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.

5. *Дишлевий О.П.* Перевірка адекватності метричних моделей властивостей програмного забезпечення / О.П. Дишлевий // Інженерія програмного забезпечення 2006: Матеріали Всеукр. конф. асп. та студ. – К.: НАУ, 2007. – С. 77–84.

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

7. *Basili V.R.* A method for collection valid software engineering data / V.R. Basili, D.M. Weiss // IEEE Transaction on Software Engineering. – 1984. – No10(6). – P. 728–38.

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

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

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

11. *Сидоров Н.А*. Структура измерителя программ / Н.А. Сидоров, В.А. Хоменко // Проблеми транспорту: зб. наук. пр. – К.: НТУ, 2005. – Вип. 2. – С. 190–195.

12. *Дишлевий О.П.* Пакет статистичного аналізу для емпіричної інженерії програмного забезпечення / О.П. Дишлевий // Наука і молодь. Приклад. сер.: зб. наук. пр. – 2009. – № 9. – С. 104–108.

13. *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.

14. *Вентцель Е.С*. Теория вероятностей: учеб. для вузов / Е.С. Вентцель. – 7-е изд., стер. – М.: Высш. шк., 2001. – 575 с.

15. *Кендалл М.* Статистические выводы и святи / М. Кендалл, А. Стюарт. – М.: Наука, Гл. ред. физ.-мат. лит., 1973. – 899 с.

16. *Айвазян С.А*. 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.

17. *Дишлевий О.П.* Підбір метрик для властивостей програмного забезпечення / О.П. Дишлевий // Проблеми програмування: наук. журн. – 2010. – №2–3. – С. 237–242.

18. *Луцький М.Г.* Підтримка придатності програмного забезпечення при модернізації та створенні авіаційної техніки / М.Г. Луцький // Наук.-техн. конф. «Створення та модернізація озброєння і військової техніки Міністерства оборони України в сучасних умовах», 10–11 вересня 2009 р. – Феодосія, 2010. – С. 20–24.

19. *Луцький М.Г.* Підтримка придатності та продовження експлуатації програмного забезпечення авіаційної техніки / М.Г. Луцький, М.О. Сидоров, Ю.М. Рябокінь // Проблеми програмування. – 2010. – №2–3. – С. 229–236.

20. *Луцкий М.Г.* Программное обеспечение – экологический подход к исследованиям / М.Г. Луцкий, Н.А. Сидоров // Natural and Artificial Intelligence. – ITHEA. – 2010. – Sofia. – Bulgaria. – P. 181–189.

Стаття надійшла до редакції 30.08.10.