Міністерство освіти і науки України
Національний університет “Львівська політехніка”

Нитребич Оксана Олександрівна

УДК 004.052.3

МОДЕЛІ ТА МЕТОДИ ОЦІНЮВАННЯ НАДІЙНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З УРАХУВАННЯМ ЙОГО АРХІТЕКТУРИ

01.05.03 – математичне та програмне забезпечення обчислювальних машин і систем

Автореферат
дисертації на здобуття наукового ступеня
кандидата технічних наук

Львів – 2015

Дисертацією є рукопис.

Робота виконана у Національному університеті “Львівська політехніка” Міністерства освіти і науки України.

Науковий керівник: доктор технічних наук, професор
Федасюк Дмитро Васильович,
Національний університет “Львівська політехніка”,
проректор з науково-педагогічної роботи,
завідувач кафедри програмного забезпечення

Офіційні опоненти: доктор технічних наук, професор
Дивак Микола Петрович,
Тернопільський національний економічний університет,
декан факультету комп’ютерних інформаційних технологій

кандидат технічних наук, доцент
Мельничин Андрій Володимирович,
Львівський національний університет ім. Івана Франка,
доцент кафедри теорії оптимальних процесів

Захист відбудеться “___” березня 2015 р. о _____ годині на засіданні спеціалізованої вченої ради Д 35.052.05 у Національному університеті “Львівська політехніка” (79013, м. Львів, вул. С.Бандери, 12).

З дисертацією можна ознайомитись у науково-технічній бібліотеці Національного університету “Львівська політехніка” (79013, м. Львів, вул. Професорська, 1).

Автореферат розіслано “____” лютого 2015 р.

Учений секретар
спеціалізованої вченої ради,
доктор технічних наук, професор Р.А. Бунь

ЗАГАЛЬНА ХАРАКТЕРИСТИКА РОБОТИ

Актуальність теми. Широке використання програмного забезпечення (ПЗ) у всіх сферах суспільної діяльності висуває високі вимоги до надійності функціонування програмних систем, відмови в роботі яких можуть призвести не тільки до значних фінансових втрат, але й створити загрози людському здоров'ю та життю. Програмні проекти стають дедалі складнішими та багатокомпонентними, тому відомі засоби оцінювання їхньої надійності та тестування є трудомісткими, вимагають великих затрат ресурсів та не забезпечують виявлення усіх дефектів.

Історія розвитку методів та засобів оцінювання надійності ПЗ розпочалася в 60-х роках минулого століття. Низка науковців (А. Ґоель, Дж. Муса, Т. Тейер, В. Ліпаєв, М. Ксіє, К. Тріведі та ін.) працювала над питаннями розроблення й дослідження моделей та методів аналізу надійності програмних систем, що дали б змогу скоротити затрати, необхідні на етапі тестування ПЗ. У зв'язку зі зростанням складності програмних комплексів, ускладненням та розширенням їхнього функціонального призначення зростає потреба в розробленні нових моделей аналізу надійності ПЗ, які враховували б внутрішню структуру програми та взаємодію її компонент.

У відомих дослідженнях моделі, розроблені на основі архітектурного підходу, використовують теорію марковських ланцюгів першого порядку з припущенням, що виконання компонент ПЗ є незалежним. З огляду на складність архітектури сучасних програмних засобів та множину сценаріїв їхнього виконання це припущення не завжди справджується. Тому для розроблення адекватних моделей оцінювання надійності ПЗ, що підвищать ефективність процесу тестування, доцільно розглядати марковські ланцюги вищих порядків, які дають змогу враховувати взаємозалежність виконання компонент програм.

Одним із важливих засобів оцінювання та забезпечення надійності програмних систем є тестування. Під час цього етапу отримують дані про помилки програм, які є вхідними даними практично для всіх типів моделей аналізу надійності програмних засобів. Також на етапі тестування усувають знайдені дефекти ПЗ, а це в цілому підвищує якість та надійність програмних систем. Останнім часом найбільше поширення має ручне тестування, яке є трудомістким процесом та потребує багато часу та фінансових ресурсів, а крім цього, його результати є малоймовірними для оцінювання надійності ПЗ у зв'язку з нерівномірним та недостатнім покриттям коду тестами. Тому автоматизоване формування набору сценаріїв тестування, що забезпечить максимальне покриття коду тестами та мінімальну кількість потрібних для цього тестів, дасть змогу підвищити ефективність процесу тестування: зменшити його тривалість, необхідні людські та економічні затрати.

Отже, розроблення нових засобів оцінювання надійності ПЗ з урахуванням архітектури і взаємозалежності виконання компонент разом з автоматизацією процесу тестування шляхом побудови ефективних тестів

програм належать до важливих завдань у сфері програмної інженерії та становлять актуальну наукову задачу.

Зв'язок роботи з науковими програмами, планами, темами. Дослідження, що становлять матеріал дисертації, безпосередньо пов'язані з науково-дослідним напрямом кафедри програмного забезпечення Національного університету «Львівська політехніка». Результати, викладені в дисертаційній роботі, отримано під час виконання таких держбюджетних науково-дослідних робіт:

- «Розробка лінійних та нелінійних математичних моделей та методів аналізу теплових режимів електронних пристроїв із неоднорідною структурою» (номер державної реєстрації 0110U001121), яку виконували в період з 2009 до 2011 року. Участь автора полягала в розробленні алгоритмів автоматизованої побудови сценаріїв тестування.
- «Розроблення моделей надійності, ризику та безпечності програмно-апаратних технічних систем» (номер державної реєстрації 0113U000000), яку виконують з 2013 року. Автор розробила модель функціонування програмної системи на основі її архітектури з урахуванням значень змінних коду ПЗ.
- «Розроблення моделей, методів та засобів оцінювання та аналізу надійності програмного забезпечення» (номер державної реєстрації 0113U003185), яку виконували в період з 2013 до 2014 року. Автором вдосконалено модель оцінювання надійності програмного продукту та проведено дослідження впливу характеристик ПЗ на кількість виявлених помилок під час процесу тестування.

Мета і задачі дослідження. Метою дисертаційної роботи є підвищення ефективності процесу тестування ПЗ шляхом розроблення нових та вдосконалення відомих засобів оцінювання надійності програмного продукту з використанням моделей, які ґрунтуються на архітектурному підході.

Для досягнення цієї мети в дисертації сформульовано і вирішено такі *задачі*:

1) аналіз відомих засобів забезпечення та оцінювання надійності програм;
2) дослідження та удосконалення моделей оцінювання надійності ПЗ з урахуванням його архітектури та взаємозалежності виконання компонент;
3) розроблення моделі функціонування ПЗ на основі аналізу змінних коду для оцінювання його надійності та покращення ефективності процесу тестування;
4) побудова методу автоматизованого формування набору тестів з використанням розробленої моделі функціонування програм;
5) дослідження залежності кількості виявлених помилок від характеристик програмних систем під час тестування стратегіями «чорної» та «білої» скриньок;
6) розроблення методу оцінювання надійності програмних засобів на основі побудованих моделей;

7) розроблення інформаційного та програмного забезпечення системи оцінювання надійності ПЗ з використанням отриманих теоретичних результатів;
8) верифікація розробленого програмного засобу та інтерпретація отриманих результатів.

Об'єктом дослідження є процес оцінювання надійності програмного забезпечення.

Предметом дослідження є моделі та методи оцінювання надійності багатокомпонентного ПЗ з урахуванням його архітектури та взаємозалежності виконання компонент.

Методи дослідження. Під час розв'язання поставлених задач використано теорію марковських процесів та методи обчислювальної математики (розв'язування систем лінійних алгебричних рівнянь) для оцінювання надійності програмних систем і засоби та методи теорії ймовірностей і математичної статистики – для розроблення методів автоматизованої побудови набору тестів програмних систем та дослідження впливу характеристик програм на кількість виявлених помилок під час процесу тестування; для створення програмного забезпечення застосовано теорію алгоритмів, теорію об'єктно-орієнтованого програмування.

Наукова новизна одержаних результатів полягає у розв'язанні наукового завдання розроблення засобів оцінювання надійності ПЗ на етапі тестування з урахуванням його архітектури та взаємозалежності виконання компонент. При цьому отримано такі наукові результати:

1) удосконалено відому модель оцінювання надійності ПЗ, яка ґрунтується на архітектурному підході, шляхом використанням марковських ланцюгів вищих порядків з дискретним часом, що дає змогу врахувати взаємозалежність виконання компонент програмних систем і таким чином підвищити адекватність моделі та точніше оцінити показники надійності програм;
2) уперше розроблено модель функціонування ПЗ, у якій на відміну від відомих, враховано множину значень змінних коду, що дає можливість підвищити ефективність процесу тестування та забезпечує вхідні дані для моделей оцінювання надійності програмного продукту з використанням архітектурного підходу;
3) уперше розроблено метод генерування сценаріїв тестування на основі побудованої моделі функціонування ПЗ, який дає змогу збільшити значення метрики покриття значень змінних коду та зменшити кількість необхідних для цього тестів;
4) уперше розроблено метод оцінювання надійності ПЗ на основі архітектурного підходу, у якому використано марковські ланцюги вищих порядків та метрику покриття значень змінних коду, що підвищує точність та ефективність процесу оцінювання надійності багатокомпонентних програмних систем.

Практичне значення одержаних результатів дисертаційної роботи зумовлене тим, що вони дають змогу підвищити точність оцінювання надійності програмної системи та ефективність її тестування. Зокрема, практично цінними результатами є:

1) розроблені модель та метод оцінювання надійності ПЗ є придатними до використання на підприємстві для аналізу ймовірності безвідмовної роботи багатокомпонентних програмних систем;
2) упроваджений шаблон проектування ПЗ можна використати під час розроблення архітектури програмних систем із заданими вимогами до надійності;
3) застосування на виробництві розробленого методу генерування сценаріїв тестування на основі моделі функціонування програмного засобу сприятиме зниженню вартості процесу тестування, а також підвищенню ефективності виявлення помилок завдяки повнішому покриттю коду тестами, що зменшить витрати на етапах експлуатації та супроводу;
4) результати дослідження залежностей кількості виявлених помилок від параметрів ПЗ під час процесу тестування дають змогу оцінити вплив різних характеристик програмного продукту на його надійність, а також дати рекомендації розробникам проектів щодо вибору оптимальної ефективної стратегії тестування;
5) на основі розроблених математичних моделей та описаних методів реалізовано програмну систему оцінювання надійності програмного забезпечення «СОН ПЗ» для автоматизованого формування множини тестів програм із максимальним обсягом покриття значень змінних коду та оцінювання надійності програмних систем на основі архітектурного підходу, що придатна для використання в ІТ-компаніях.

Результати дисертаційної роботи застосовано під час оцінювання надійності розроблених програмних систем та генерування набору тестів ПЗ на підприємствах ТзОВ «Едвантіс», ІП «Логіка», а також використано в навчальному процесі Національного університету «Львівська політехніка» під час читання курсу лекцій з дисциплін «Якість програмного забезпечення та тестування» та «Основи теорії надійності програмних систем», що підтверджено відповідними актами.

Особистий внесок здобувача. Усі теоретичні та практичні результати, подані в роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, здобувачу належить: розроблення моделі функціонування програмної системи [1]; розроблення методу оцінювання надійності ПЗ на основі архітектурного підходу [2]; розроблення методів автоматизованого генерування множини тестів програм та дослідження залежностей кількості виявлених під час процесу тестування помилок від характеристик програмного продукту розробленими методами [3, 7]; удосконалення моделі оцінювання надійності ПЗ з урахуванням марковських ланцюгів вищих порядків [4]; проведення аналізу та обґрунтування використання марковських ланцюгів вищих порядків й інформаційних

критеріїв для визначення порядку таких ланцюгів у задачах оцінювання надійності програмного продукту [5, 6, 9–12]; побудова шаблону проектування ПЗ компактного збору даних для марковських ланцюгів вищих порядків [8].

Апробація результатів дисертації. Основні теоретичні положення та практичні результати дисертаційної роботи обговорювали на таких наукових конференціях: VI, VII International Scientific and Technical Conference «Computer Sciences and Information Technologies» «CSIT» (Львів, 2011, 2012); VI International Academic Conference of Young Scientists «Computer Science and Engineering CSE-2013» (Львів, 2013); VIII, X International Conference on Perspective Technologies and Methods in MEMS Design «MEMSTECH» (Поляна, 2012, 2014); XII International Conference The Experience of Designing and Application of CAD Systems in Microelectronics «CADSM» (Поляна, 2013), а також на наукових семінарах кафедри програмного забезпечення Національного університету «Львівська політехніка» (Львів, 2011–2014) та кафедри комп'ютерних наук Тернопільського національного економічного університету (Тернопіль, 2014).

Публікації. Результати дисертації викладено у дванадцяти публікаціях. Чотири статті надруковано в наукових фахових виданнях України [3–6], дві – у міжнародних періодичних виданнях [1, 2], шість тез доповідей опубліковано в матеріалах міжнародних науково-технічних конференцій [7–12].

Структура та обсяг роботи. Дисертація складається зі вступу, чотирьох розділів, висновків, списку використаних джерел (134 найменування) та трьох додатків. Загальний обсяг дисертації – 153 сторінки, із яких 130 сторінок основного тексту. Робота містить 33 рисунки і 15 таблиць.

ОСНОВНИЙ ЗМІСТ РОБОТИ

У **вступі** викладено загальну характеристику дисертаційної роботи, обґрунтовано актуальність теми, сформульовано мету і задачі, методи дослідження, наголошено на науковій новизні роботи, висвітлено практичну цінність проведених науково-технічних досліджень та отриманих результатів. Наведено зміст і структуру дисертації, інформацію про публікації, апробацію і впровадження результатів роботи.

У **першому розділі** проведено огляд літератури та аналіз відомих моделей оцінювання надійності програмних систем.

Наведено класифікацію моделей оцінювання надійності ПЗ, створених на основі архітектурного підходу, тобто моделей, які враховують структуру програмних систем, складену з сукупності компонент (функціональна частина програми, яку можна редагувати та тестувати окремо від інших) та взаємозв'язків між ними. Показано, що в моделях цього типу здебільшого не враховано взаємозалежність виконання компонент, яка нерідко трапляється в реальних програмних продуктах. Крім цього, описано відомі засоби генерування сценаріїв тестування програмних систем на основі різних моделей функціонування програм, подано їхні основні переваги та недоліки.

Обґрунтовано необхідність розроблення нових моделей та засобів оцінювання надійності програм на основі архітектурного підходу, які б адекватніше описували процес аналізу надійності ПЗ із урахуванням його структури та впливу характеристик компонент на систему в цілому, а також умотивовано потребу розроблення нових методів автоматизованого формування тестів ПЗ різними стратегіями, що підвищили б ефективність процесу тестування.

У **другому розділі** описано розроблену модель функціонування ПЗ, у якій уперше використано інформацію про класи еквівалентності змінних коду (класи, що їх виділяють шляхом вибору кожної вхідної вимоги зі специфікації ПЗ і розбиття її на дві або більше груп), на основі якої впроваджено метод автоматизованого формування сценаріїв тестування.

Модель функціонування програмної системи відображає процес її виконання та певний набір характеристик і подається у вигляді зваженого графа, вузлами якого є компоненти програми, а зваженими ребрами – ймовірності передавання управління між ними (рис. 1). Коректна модель функціонування ПЗ, що враховує необхідні характеристики програм, дає змогу точніше здійснювати автоматизовану побудову сценаріїв тестування та оцінювати надійність програмного продукту моделями архітектурного підходу.

Нову модель функціонування ПЗ подано орієнтованим графом $G=\{C, P\}$, де C – множина компонент програмної системи, P – множина переходів між відповідними компонентами. Процес виконання ПЗ зображено за допомогою пройдених шляхів такого графа, кожну вершину якого подано як компоненту програмної системи, що змінює значення множини змінних, а ребра відповідають за послідовності виклику компоненти (рис. 1).

Якщо використати такі позначення:

- V_i^{used} – множина змінних та відповідних класів еквівалентності, які використано в компоненті $C_i\,(i=\overline{1,N})$;
- $V_i^{change}=V_i^{change\_user}\bigcup V_i^{change\_program}$ – множина змінних та відповідних класів еквівалентності, що змінюються в компоненті $C_i\,(i=\overline{1,N})$, відповідно, користувачем та програмно;
- V_i^{fail} – множина змінних та відповідних неправильних класів еквівалентності, що можуть викликати помилку в компоненті $C_i\,(i=\overline{1,N})$;

тоді кожен вузол графа C_i є набором відповідних множин $(V_i^{used}, V_i^{change}, V_i^{fail})$, а також для нього характерний список інцидентних дуг, які, своєю чергою, містять ймовірність p_{ij} передавання контролю до іншої компоненти C_j $(i, j=\overline{1,N})$.

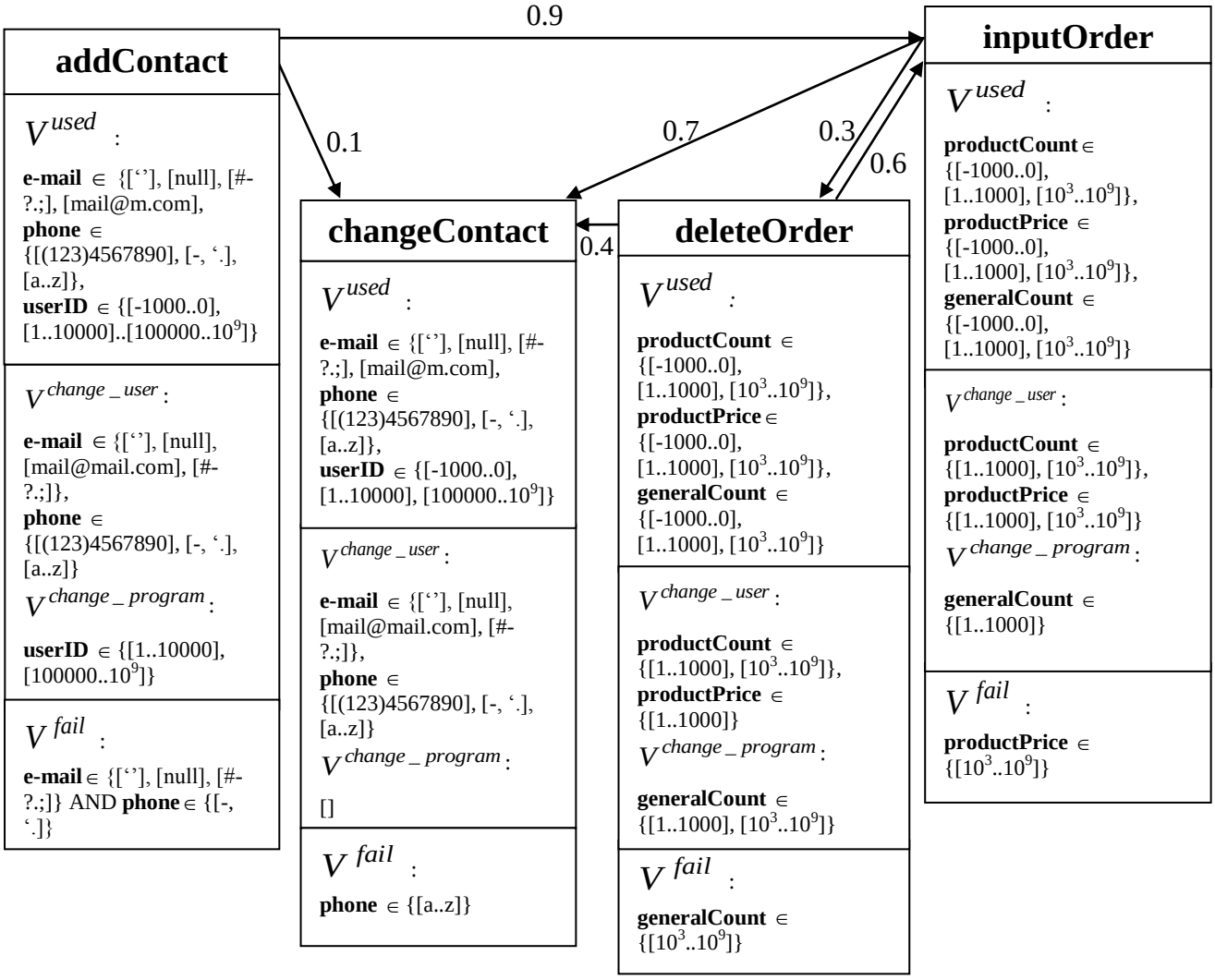

Рис.1. Приклад моделі функціонування програми з компонентами: addContact, changeContact, deleteOrder, inputOrder

У розділі досліджено практичні питання створення такої моделі функціонування ПЗ, проаналізовано характеристики її компонент та отримано числові показники кількості потрібних тестів для повного покриття компоненти залежно від її типу (метод, клас, пакет). Також запропоновано новий спосіб практичної реалізації описаної моделі функціонування ПЗ з поєднанням техніки логування для обчислення ймовірностей передавання управління між компонентами та аналізу коду для визначення набору компонент та їхніх змінних.

За розробленою моделлю кожен сценарій тестування ПЗ складається з декількох кроків $T^k = \{T_0^k, T_1^k, ..., T_n^k\}$, де кожен крок $T_j^k = (C_j^k, V_j^k)$ визначений компонентою C_j^k та множиною змінних V_j^k з відповідними класами еквівалентності, що фактично змінюються у цій компоненті. Змінні набувають своїх значень на кожному кроці тесту внаслідок введення даних користувачем, зчитуванням з бази даних та ін., через що під час виконання такого сценарію

тестування можуть виникнути помилки. На практиці набори тестів формують ітераційно, крок за кроком виконують на ПЗ, враховуючи виявлені помилки під час формування наступного набору тестів.

У розділі використано такі позначення: $F_j(C_i)$ – кількість тестів, під час яких відбулася помилка і які враховують компоненту C_i на *j*-тій ітерації; $Num(C_i)$ – множина номерів усіх компонент, у які можна передати контроль із компоненти C_i; $Com(C_i)$ – множина всіх компонент, у які можна передати контроль із компоненти C_i; N – кількість усіх вершин графа функціонування програмної системи; TS – набір усіх сценаріїв тестування; $Cov(C_i)$ – міра метрики покриття компоненти C_i тестами (кількість виконань компоненти); v_i – змінна компоненти C_i.

Розроблено метод автоматизованого формування тестових сценаріїв стратегією «чорної» скриньки, який складається з двох основних частин: початкове генерування тестів без інформації про помилки та решти *K* наборів тестів, що включають цю інформацію. Основними кроками цього методу є такі:

Крок 1. *Ініціалізація.* Вибір мінімальної кількості α покриття всіх компонент ПЗ та середньої довжини β тестового сценарію з досвіду тестувальника. Множина тестів $TS=\{\}$, номер ітерації тестування $m=0$, номер тесту $k=0$.

Крок 2. Формування сценарію тестування T^k під час початкового генерування набору тестів:

1) $j=0$ – номер кроку такого тесту;
2) випадковим чином вибирають початковий крок T_j^k тесту T^k з імовірністю $p_j^k = 1/N$;
3) з імовірністю $p_{j+1}^k = 1/card(Com(C_j))$ вибирають наступний крок T_{j+1}^k у *k*-му сценарії; $j=j+1$;
4) якщо $j \le \beta$, тоді виконують пункт 3;
5) приєднання сформованого тесту до набору: $TS = TS \bigcup \{T^k\}$.

Крок 3. Перевірка покриття всіх компонент. Якщо $Cov(C_i) < \alpha, i = \overline{1,N}$, тоді $k=k+1$, тобто здійснюють перехід на крок 2.

Крок 4. Виконання всіх розроблених тестів та обчислення $F_m(C_i), i = \overline{1,N}$. Якщо $F_m(C_i) = \varnothing, i = \overline{1,N}$, тоді виконують крок 9, інакше $m=m+1$;

Крок 5. Формування *m*-го набору тестів ($m \ge 1$). $TS=\{\}$, $k=0$.

Крок 6. Формування T^k сценарію тестування для *m*-го набору тестів:

1) $j=0$ – номер кроку такого тесту;

2) випадковим чином вибирають номер першого кроку T_j^k тесту T^k, з імовірністю $p_j^k = \max\limits_{i=1..N} \dfrac{F_{m-1}(C_i)}{\sum\limits_{l=1}^{N} F_{m-1}(C_l)}$;

3) з імовірністю $p_{j+1}^k = \max\limits_{i \in Num(C_j)} \dfrac{F_{m-1}(C_i)}{\sum\limits_{l \in Com(C_j)} F_{m-1}(C_l)}$ обирають наступний крок у T_{j+1}^k сценарії; $j=j+1$;

4) якщо $j \leq \beta$, тоді здійснюють перехід на пункт 3;

5) приєднання побудованого тесту до набору: $TS = TS \bigcup \{T^k\}$.

Крок 7. Перевірка покриття усіх компонент. Якщо $Cov(C_i) < \alpha, i = \overline{1,N}$, тоді $k=k+1$, тобто здійснюють перехід на крок 6.

Крок 8. Перехід на крок 4.

Крок 9. Кінець.

Побудовані послідовності компонент виконують для різних вхідних даних так, щоб кожний вхідний параметр набував принаймні одного значення з кожного класу еквівалентності. Класи еквівалентності для компоненти, яку тестують «чорною» скринькою, будують не з аналізу коду, а відповідно до досвіду тестувальника. При цьому вхідні дані можуть і не набути усіх можливих значень і деякі помилки будуть пропущені, що є недоліком цієї стратегії тестування.

Також у розділі розроблено метод автоматизованого генерування тестів стратегією «білої» скриньки на основі описаної моделі функціонування ПЗ, що дає змогу забезпечити покриття метрики значень змінних коду. Для цього введено міру покриття компоненти $C_i (i<N)$ тесту T, яку визначають співвідношенням

$$\alpha(T, C_i) = V_i^{used} \bigcap (V_1 \bigcup V_2 \bigcup .. \bigcup V_{i-1}). \quad (1)$$

Крім цього, позначено максимальну довжину тестового сценарію для певного ПЗ через β_{max}. Тоді міру метрики значень змінних коду (англ. parameter value coverage – PVC) визначають у вигляді

$$PVC(TS) = \sum_{i=1}^{N} \sum_{v_i \in V_i^{used}} Ind_{v_i} \left(\bigcup_{T \in TS} \alpha(T, C_i) \right), \quad (2)$$

де $Ind_v(C) = \begin{cases} 0, v \notin C; \\ 1, v \in C. \end{cases}$

Також уведено поняття складності виконання тестів

$$CM(TS) = \sum_{i=1}^{card(TS)} \sum_{j=1}^{card(T^i)} V_j^i. \quad (3)$$

Отже, умовами формування множини сценаріїв тестування ПЗ стратегією «білої» скриньки є

$$CM(TS) \to \min$$

з обмеженнями $PVC(TS) = PVC_{\max}(TS) = \sum_{i=1}^{N} card\,(V_i^{used})$. (4)

Описаний у розділі метод генерування тестів стратегією «білої» скриньки складається із таких основних кроків:

Крок 1. *Ініціалізація*. Формують початкову множину тестових сценаріїв, що складаються лише з одного кроку $TS = \{(C_0, V_0^{change}), (C_1, V_1^{change}), ..., (C_N, V_N^{change})\}$. На цьому кроці тестового сценарію перебирають усі можливі параметри, оскільки вони впливатимуть на всі подальші кроки.

Крок 2. *Знаходження повного покриття значень змінних*. Множина тестових сценаріїв, побудована на попередньому кроці, розширюється так: для сценарію $T = ((C_0, V_0), (C_1, V_1), ..., (C_K, V_K))$ з множини *TS* знаходять усі компоненти C_{K+1}, які суміжні з компонентою C_K, і додають новий крок (C_{K+1}, V_{K+1}) до сценарію *T*, де $V_{K+1} = V_{K+1}^{change} \setminus \bigcup_{i=1}^{K} V_i$. Приєднують новий сценарій до множини *TS*, якщо при цьому зростає $PVC(TS)$. Якщо $PVC(TS) = PVC_{\max}(TS)$, то здійснюють перехід на крок 3, в іншому разі повторюють крок 2.

Крок 3. *Мінімізація складності виконання сценарію тестування.* Для кожного сценарію з множини *TS* виконують таку перевірку: якщо в разі вилучення сценарію з множини *TS* величина $PVC(TS)$ не змінюється, то його видаляють, в іншому разі перевіряють кожен крок (C_K, V_K) і видаляють з множини V_K ті змінні, які не впливають на $PVC(TS)$. Це виконують для мінімізації $CM(TS)$.

Крок 4. *Перебір класів еквівалентності змінних.* Сформовані тестові сценарії з множини *TS* виконують для різних вхідних даних так, щоб кожний вхідний параметр набував принаймні одного значення з кожного класу еквівалентності. Для підвищення ефективності тестування необхідно проаналізувати взаємопов'язані змінні та перебрати всі кортежі відповідних значень класів еквівалентності, що є значно складнішим завданням. Інший підхід полягає в більшій дискретизації вузлів графа функціонування ПЗ до рівня програмних виразів та операторів управління, що дає змогу глибоко проаналізувати покриття та взаємний вплив змінних.

Крок 5. *Ітераційне виконання тестових сценаріїв.* Після побудови та виконання тестових сценаріїв знаходять множину компонент $F(C_i)$, у яких виникли помилки. Якщо $F(C_i) = \varnothing$, то алгоритм закінчується, інакше, починаючи з кроку 1, будують нову множину тестових сценаріїв *TS* з новим покриттям у вигляді

$$PVC(TS) = \sum_{C_i \in F(C)} \sum_{v_i \in V_i^{used} \bigcap V_i^{fail}} Ind_{v_i}(\bigcup_{T \in TS} \alpha(T, C_i)). \quad (5)$$

Описаний метод дає змогу створювати множини сценаріїв тестування ПЗ з максимальним покриттям коду за мінімальної складності їхнього виконання, що підвищує ефективність процесу тестування та усунення дефектів програми.

З використанням розробленого методу формування тестів проведено дослідження залежностей кількості виявлених помилок від характеру процесу тестування та різних параметрів моделі функціонування програмного продукту, таких як кількість компонент, кількість змінних у компонентах, зв'язність компонент (рис. 2). На основі проведених досліджень запропоновано рекомендації розробникам програмних проектів стосовно вибору типу стратегії тестування на основі створеної моделі функціонування відповідного ПЗ. Окрім того, якісний вигляд залежностей, представлених на рис. 2 (*а*, *б*), співпадає з відомими з літератури даними, а це підтверджує коректність розробленого методу формування множини тестів.

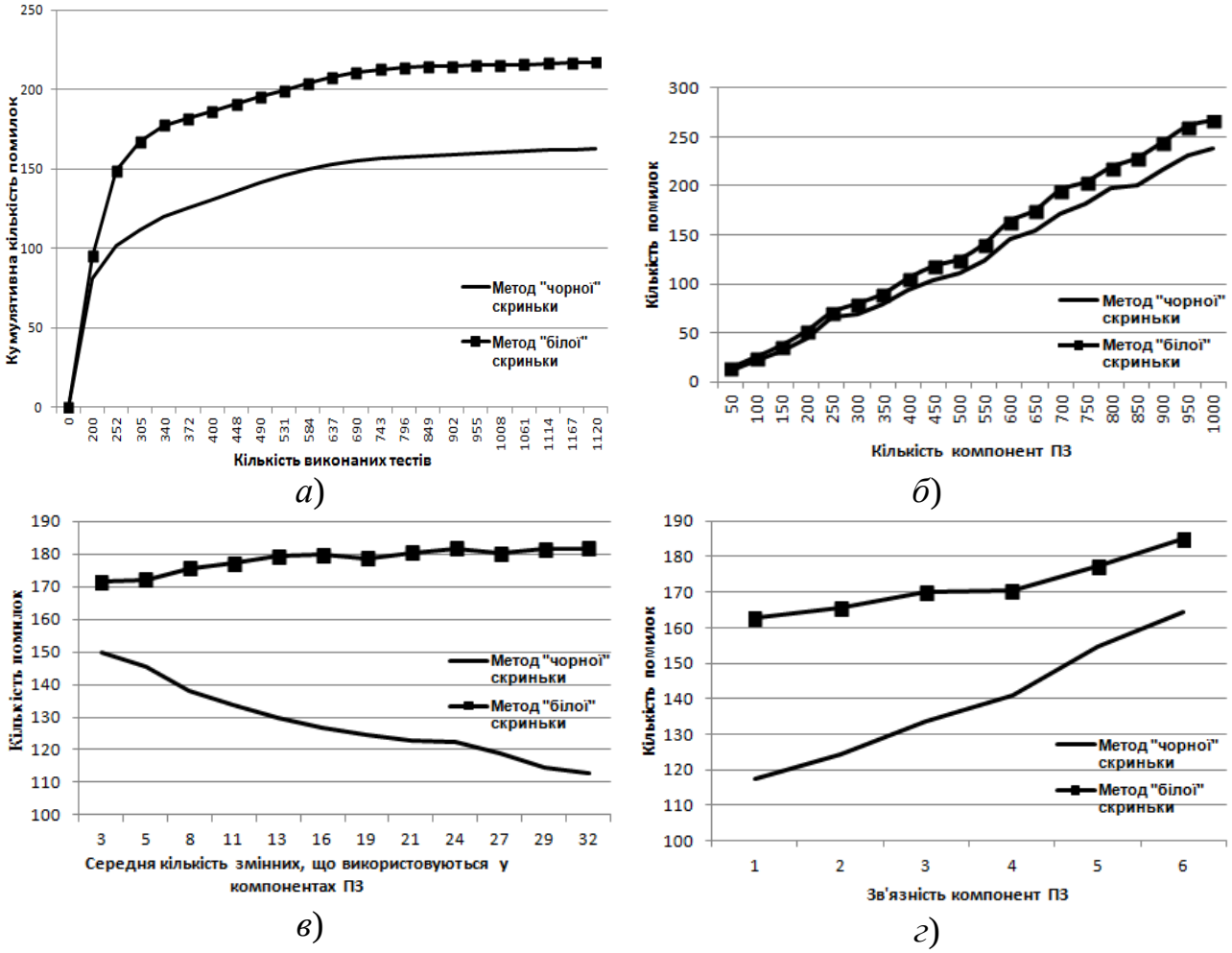

Рис. 2. Графіки залежності кількості помилок виявлених під час тестування від: *а*) кількості виконаних тестів; *б*) кількості компонент; *в*) середньої кількості змінних у компоненті; *г*) зв'язності компонент для стратегій тестування «білою» та «чорною» скриньками

Проведено зіставлення кількості виявлених помилок, отриманих у результаті послідовного тестування спочатку відомими методами «чорної» та «білої» скриньок, а потім з використанням нового розробленого в дисертації методу для п’яти тестових програмних систем. Результати такого дослідження показали, що використання розробленого методу автоматизованого формування тестів дає змогу підвищити ефективність виявлення помилок на 7–10 % (рис. 3).

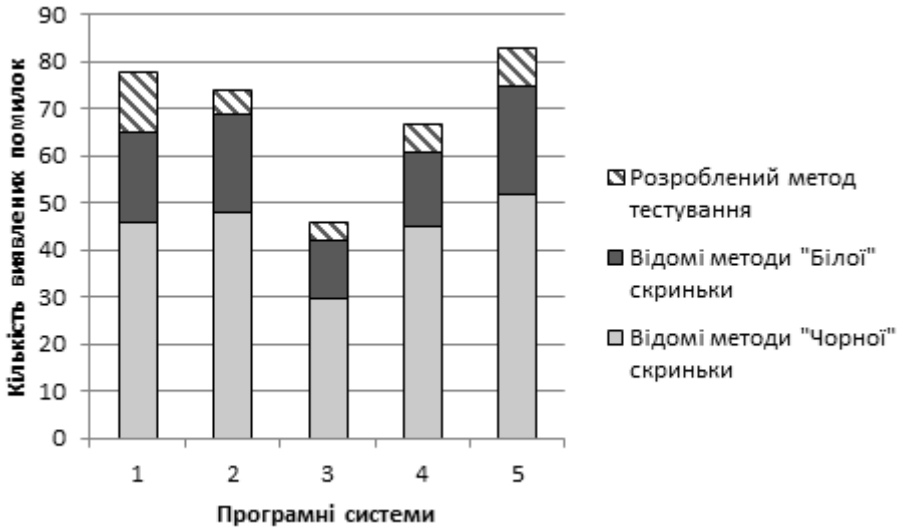

Рис. 3. Діаграма кількості виявлення помилок за допомогою раніше відомих та розробленого методів тестування

У **третьому розділі** для врахування взаємозалежності виконання компонент удосконалено модель оцінювання надійності ПЗ з використанням марковських ланцюгів вищих порядків. У моделі розглянуто поглинальний марковський процес, що передбачає наявність одного чи більше поглинальних станів, тобто станів, з яких не можна перейти в інші.

Розроблена модель оцінювання надійності ПЗ є ієрархічною, тобто спочатку обчислюють параметри архітектури ПЗ на основі моделі функціонування програмної системи з використанням теорії марковських процесів, а далі враховують поведінку помилок кожної з компонент. Згідно з такою моделлю імовірність безвідмовної роботи цілої системи обчислюють за формулою

$$R = \prod_{i=1}^{N} R_i \cdot \tag{6}$$

Своєю чергою, імовірність безвідмовної роботи кожної компоненти з використанням марковських ланцюгів вищих порядків обчислюють співвідношенням

$$R_l = \exp\left(- \int_0^{W_l} \lambda_l(t)dt \right), \tag{7}$$

де $W_l = \sum_{ij..k} V_{ij...kl} t_{ij...kl}$.

Для знаходження $V_{ij..kl}$ – очікуваної кількості виконань компоненти i залежно від виконання попередніх K компонент – розв'язано систему лінійних рівнянь:

$$V_{j..kl} = q_{j..kl} + \sum_{i=1}^{N-1} V_{ij..k}\, p_{j..kl} \,. \qquad (8)$$

Крім цього, на основі моделі функціонування ПЗ обчислено такі числові вхідні параметри моделі оцінювання надійності ПЗ: $p_{ij..kl}$ – імовірність переходу в компоненту l залежно від виконання попередніх K компонент; $q_{ij..kl}$ – початковий імовірнісний вектор; $t_{ij..kl}$ – тривалість виконання компоненти l залежно від виконання попередніх K компонент.

У розділі проведено аналіз використання інформаційних критеріїв (критерії Акаіке та Байєса) для визначення порядку марковського ланцюга та показано випадки їх застосування залежно від обсягу вибірки.

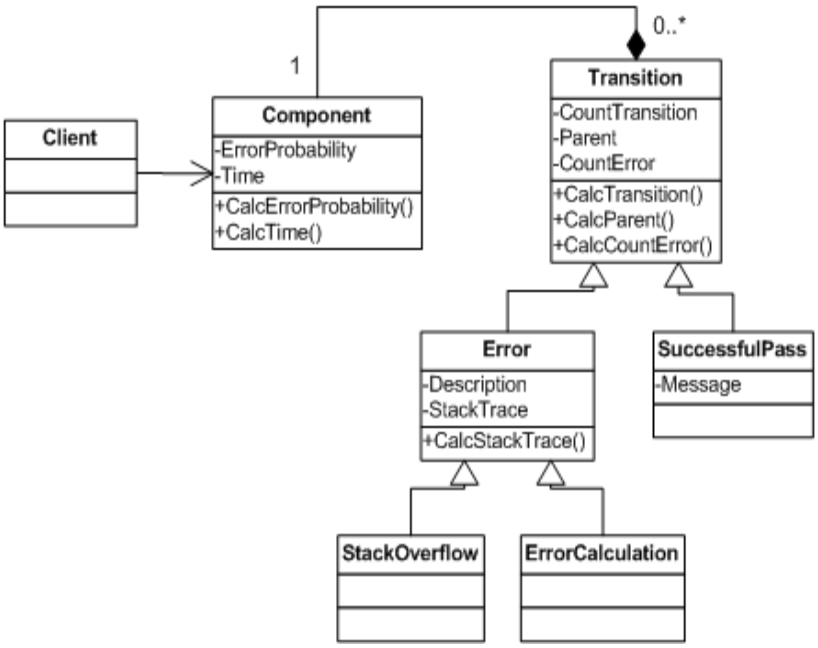

Рис. 4. UML-діаграма шаблону проектування зображення графа марковського ланцюга вищого порядку

Для практичного використання розробленої моделі оцінювання надійності ПЗ, а саме побудови графа потоку керування, розроблено шаблон, який дає змогу створювати послідовності виконання компонент будь-якої довжини. Новий шаблон проектування програмного продукту розроблено на основі шаблону «Компонувальник», який використовують для опису складних систем, з модифікацією для зображення переходів у графі (рис. 4).

Як показано на рис. 4, внутрішні вершини графа містять лічильник помилок і час виконання компоненти, кількість переходів зберігається в класі Transition, листки відображають помилки або успішне виконання тесту. На етапі тестування ПЗ під час виконання нових тестів змінюється лише лічильник переходів, а загальна структура змінюється лише тоді, коли виникають нові послідовності виконання компонент. Отже, шаблон містить тільки необхідні послідовності вищого порядку, детальну інформацію про ймовірності передавання управління між компонентами, інтенсивність виявлення помилок та тривалість виконання кожної компоненти.

У розділі описано метод оцінювання надійності ПЗ на основі розробленої моделі та наведено приклад його застосування, який складається з таких основних етапів:

1. Визначення вхідних параметрів моделі оцінювання надійності ПЗ, що, своєю чергою, має низку підзадач:
   – за допомогою розробленого методу генерування сценаріїв тестування визначають інтенсивності виявлення помилок кожної компоненти з використанням моделей «чорної» скриньки;
   – обчислюють матриці ймовірностей переходів між компонентами ПЗ та тривалість їхнього виконання (з використанням логування, діаграм уніфікованої мови моделювання – UML-діаграм, прихованих марковських моделей);
   – визначають порядок марковського ланцюга (залежно від розміру вибірки даних використовують критерії Акаіке або Байєса).
2. Оцінювання надійності ПЗ з використанням апарату марковських ланцюгів вищих порядків.

З використанням розробленої програмної системи (розділ 4) обчислено ймовірності безвідмовної роботи тестових програмних систем за допомогою моделі оцінювання надійності ПЗ з урахуванням першого та вищого порядку марковського ланцюга (рис. 5) та проведено їхнє порівняння з результатами тестування. Модель оцінювання надійності програм з використанням марковських ланцюгів вищих порядків дає змогу збільшити точність оцінки надійності до 6 %.

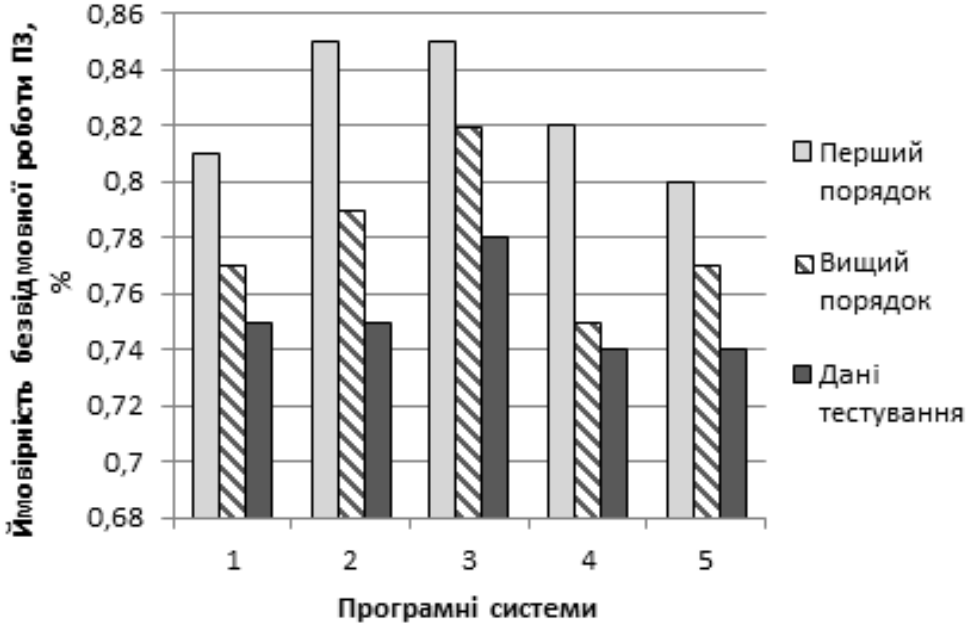

Рис. 5. Діаграма значень імовірності безвідмовної роботи програмних систем, отриманих за допомогою моделей першого та вищого порядків

У **четвертому розділі** описано архітектуру та організацію розробленої програмної системи оцінювання надійності ПЗ «СОН ПЗ» (рис. 6) та особливості її реалізації.

Визначено основні функції незалежних модулів «СОН ПЗ»: модуль для побудови моделі функціонування програми; модуль генерування набору тестів на основі розробленої моделі функціонування програмної системи; модуль для оцінювання надійності програмного продукту з використанням теорії марковських ланцюгів вищих порядків (рис. 7). Описано інформаційну модель створеної системи «СОН ПЗ» у вигляді UML-діаграми класів, яка враховує класи досліджуваних об’єктів та класи вихідних даних, що забезпечує гнучкість використання цієї системи. У розділі також наведено розроблені алгоритми для створення моделі функціонування програмної системи та оцінювання надійності ПЗ, на основі яких здійснюється реалізація «СОН ПЗ».

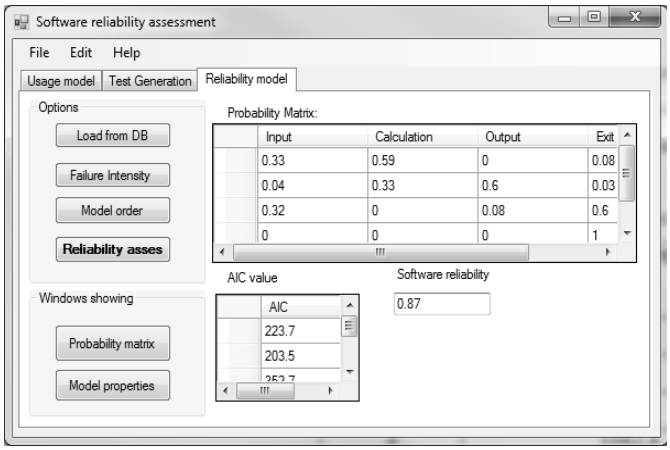

Рис. 6. Вигляд віконного інтерфейсу програмної системи «СОН ПЗ»

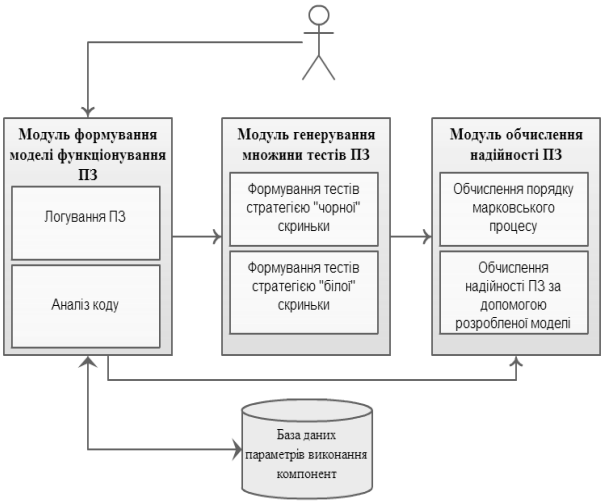

Рис. 7. Функціональна схема «СОН ПЗ»

ОСНОВНІ РЕЗУЛЬТАТИ ТА ВИСНОВКИ

У дисертації розв’язано наукове завдання розроблення математичного й програмного забезпечення для оцінювання надійності програмного продукту з використанням марковських ланцюгів вищих порядків, що дає змогу підвищити ефективність процесу тестування.

Основні наукові та практичні результати дисертаційної роботи:

1. Удосконалено математичну модель оцінювання надійності ПЗ, яка завдяки використанню марковських ланцюгів вищих порядків дає змогу врахувати взаємозалежність виконання компонент програмного продукту, що підвищує адекватність моделі у випадку багатокомпонентної програмної системи зі складними сценаріями виконання.
2. Побудовано нову модель функціонування ПЗ, яка враховує вплив аргументів компонент, глобальних та локальних змінних та їхніх класів еквівалентності і дає змогу обчислити метрику покриття значень змінних коду з використанням даних про сценарії тестування програмної системи. Досліджено практичні аспекти побудови такої моделі функціонування

програмного продукту та сформульовано рекомендації щодо вибору стратегії тестування ПЗ.

3. Розроблено метод автоматизованого генерування набору тестів програмної системи на основі моделі функціонування ПЗ, що забезпечує рівномірне покриття коду тестами, а саме метрику покриття значень змінних коду, та підвищує ефективність процесу тестування на 7–10 %, водночас зменшуючи потрібні для нього часові, фінансові та людські ресурси.

4. Досліджено залежність кількості виявлених помилок від характеру процесу тестування (кількості сценаріїв тестування) та від параметрів ПЗ (кількість компонент, середня кількість змінних у компоненті, складність помилок програмних продуктів, середня зв'язність компонент, складність виявлення помилок) для стратегій тестування «білої» та «чорної» скриньок, що є підставою для практичних рекомендацій розробникам проектів стосовно вибору стратегії тестування програмного продукту.

5. Для підвищення ефективності оцінювання надійності програмних систем упроваджено шаблон проектування ПЗ, призначений для отримання вхідних параметрів моделі оцінювання надійності програм з урахуванням марковських ланцюгів вищих порядків без суттєвих змін структури програм та з низькими вимогами до пам'яті.

6. На основі побудованих моделей розроблено метод оцінювання надійності ПЗ, основними кроками якого є вибір стратегії тестування, отримання достовірних даних для моделей надійності, визначення порядку моделі. Цей метод дає змогу збільшити точність оцінювання показників надійності до 6 %.

7. Розроблено архітектуру та інформаційну модель програмної системи для оцінювання надійності ПЗ. Створена на основі побудованих моделей та розроблених методів програмна система реалізує розроблене математичне забезпечення та дає змогу автоматизувати опрацювання експериментальних даних, оцінювати надійність програмних продуктів, а також здійснювати автоматизоване генерування набору тестів.

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Fedasyuk D. Variable state-based software usage model / D. Fedasyuk, V. Yakovyna, P. Serdyuk, O. Nytrebych // Econtechmod: an international quarterly journal on economics in technology, new technologies and modelling processes. – 2014. – V. 3, N. 2. – P. 15–20.
2. Yakovyna V. Software reliability assessment using high-order Markov chains / V. Yakovyna, D. Fedasyuk, O. Nytrebych, I. Parfenyuk, V. Matselyukh //

International Journal of Engineering Science Invention. – 2014. – V. 3, N. 7. – P. 1–6.

3. Федасюк Д. В. Метод побудови сценаріїв тестування програмного забезпечення на основі аналізу його змінних / Д. В. Федасюк, В. С. Яковина, П. В. Сердюк, О. О. Нитребич // Інформаційні технології та комп'ютерна інженерія. – Вінниця, 2014. – № 2 (30). – С. 50–58.
4. Яковина В. Використання марковських ланцюгів вищого порядку в задачах моделювання надійності програмного забезпечення / В. Яковина, П. Сердюк, О. Нитребич, Д. Федасюк // Комп'ютерні науки та інформаційні технології: Вісник Національного університету «Львівська політехніка». – Львів : Видавництво Національного університету "Львівська політехніка", 2013. – № 771. – С. 209–213.
5. Яковина В. С. Аналіз використання інформаційних критеріїв у моделях оцінки надійності програмного забезпечення / В. С. Яковина, Д. В. Федасюк, О. О. Нитребич // Нові рішення в сучасних технологіях: Вісник Національного технічного університету «ХПІ». Збірник наукових праць. – Харків, 2014. – № 26 (1069). – С. 108–115.
6. Яковина В. Використання інформаційного критерію АКАІКЕ в задачах моделювання надійності програмного забезпечення / В. Яковина, О. Нитребич, Д. Федасюк // Комп'ютерні науки та інформаційні технології: Вісник Національного університету «Львівська політехніка». – Львів : Видавництво Національного університету "Львівська політехніка", 2012. – № 732. – С. 190–192.
7. Fedasyuk D. Algorithm for automated test scenario construction / D. Fedasyuk, V. Yakovyna, P. Serdyuk, O. Nytrebych // Proc. of the Xth Intern. Conf. on Perspective Technologies and Methods in MEMS Design (MEMSTECH'2014), June 22–24, 2014. – Lviv, 2014. – P. 60–62.
8. Yakovyna V. Markov chain dynamic representation model for reliability testing / V. Yakovyna, P. Serdyuk, O. Nytrebych // Proc. of the XIIth Intern. Conf. on the Experience of Designing and Application of CAD Systems in Microelectronics (CADSM'2013), February 19–23, 2013. – Polyana, 2013. – P. 384–386.
9. Yakovyna V. Representation of higher order Markov process through equivalent first order Markov process / V. Yakovyna, O. Nytrebych, D. Fedasyuk // Proc. of the VIth Intern. Conf. of Young Scientists «Computer Science and Engineering 2013» (CSE'2013), November 21–23, 2013. – Lviv, 2013. – P. 216–217.
10. Yakovyna V. Modified high-order software reliability Gokhale model / V. Yakovyna, O. Nytrebych, D. Fedasyuk // Proc. of the VIIth Intern. Scientific and Technical Conference (CSIT'2012), November 20–24, 2012. – Lviv, 2012. – P. 194–196.

11. Yakovyna V. On using AIC in software reliability modeling / V. Yakovyna, D. Fedasyuk, O. Nytrebych // Proc. of the VIIIth Intern. Conf. on Perspective Technologies and Methods in MEMS Design (MEMSTECH'2012), April 18–21, 2012. – Polyana, 2012. – P. 93–95.
12. Yakovyna V. Review of architecture-based software reliability models / V. Yakovyna, O. Nytrebych // Proc. of the VIth Intern. Scientific and Technical Conference on Computer Science and Information Technologies (CSIT'2011), November 16–19, 2011. – Lviv, 2011. – P. 194–196.

АНОТАЦІЇ

Нитребич О.О. Моделі та методи оцінювання надійності програмного забезпечення з урахуванням його архітектури. – На правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 01.05.03 – «Математичне та програмне забезпечення обчислювальних машин і систем». – Національний університет «Львівська політехніка», Міністерство освіти і науки України, Львів, 2014.

Дисертація присвячена оцінюванню надійності програмної системи з урахуванням її архітектури, зокрема вдосконаленню наявних моделей оцінювання надійності програмного забезпечення (ПЗ) та розробленню нових методів автоматизованого формування набору тестів програмних систем для підвищення ефективності їхнього тестування.

У роботі вдосконалено модель оцінювання надійності ПЗ з використанням марковських ланцюгів вищих порядків, що забезпечує врахування взаємозалежності виконання компонент. Для визначення порядку марковського процесу проаналізовано інформаційні критерії. Розроблено новий шаблон проектування ПЗ, що сприяє компактному зборові даних для моделі оцінювання надійності програм. Крім того, на основі вдосконаленої моделі розроблено метод аналізу надійності ПЗ з використанням марковських ланцюгів вищих порядків та описано всі його підзадачі з можливими випадками розв'язання, що дає змогу підвищити точність показників надійності програмних систем до 6 %.

Розв'язано важливу наукову задачу автоматизованого генерування сценаріїв тестування на основі розробленої моделі функціонування програмного продукту, де вперше використано дані про класи еквівалентності змінних коду, а це дозволяє отримати достовірні вхідні дані задачі оцінювання надійності ПЗ. Таке формування тестів враховує метрику покриття значень змінних коду й дає змогу покращити ефективність тестування на 7–10 %. Також досліджено вплив характеристик ПЗ на кількість виявлених помилок під час процесу тестування, що допомагає виробити практичні рекомендації керівникам проектів стосовно вибору стратегії тестування.

Ключові слова: надійність програмного забезпечення, моделі оцінювання надійності програмного забезпечення, марковський ланцюг вищого порядку, генерування сценаріїв тестування, життєвий цикл програмного забезпечення.

Нытрэбыч О.О. Модели и методы оценки надежности программного обеспечения с учетом его архитектуры. – На правах рукописи.

Диссертация на соискание ученой степени кандидата технических наук по специальности 01.05.03 – «Математическое и программное обеспечение вычислительных машин и систем». – Национальный университет «Львивська политэхника», Министерство образования и науки Украины, Львов, 2015.

Диссертация посвящена оцениванию надежности программной системы с учетом её архитектуры, в частности усовершенствованию существующих моделей оценивания надежности программного обеспечения (ПО) и разработке новых методов автоматизированного формирования набора тестов программ для повышения эффективности их тестирования.

В диссертационной работе усовершенствована модель оценивания надежности ПО с использованием марковских цепей высших порядков, что обеспечивает учет взаимозависимости выполнения компонент. Для определения порядка марковского процесса проанализированы информационные критерии. Разработан новый шаблон проектирования ПО, который обеспечивает компактный сбор данных для модели оценивания надежности программ. Кроме того, на основе описанной модели разработан метод анализа надежности ПО с учетом марковских цепей высших порядков и описаны все его подзадачи с возможными случаями решения, что дает возможность повысить точность показателей надежности программных систем до 6 %.

Решена важная научная задача автоматизированного генерирования сценариев тестирования на основе разработанной модели функционирования программного продукта, где впервые использованы данные о классах эквивалентности переменных кода, а это позволяет получить достоверные исходные данные задачи оценивания надежности ПО. Такое формирование тестов учитывает метрику покрытия значений переменных кода и позволяет повысить эффективность тестирования на 7–10 %. Также исследовано влияние характеристик ПО на количество обнаруженных отказов в процессе тестирования, что позволяет выработать практические рекомендации руководителям проектов по выбору стратегии тестирования.

Ключевые слова: надежность программного обеспечения, модели оценивания надежности программного обеспечения, марковская цепь высшего порядка, генерирование сценариев тестирования, жизненный цикл программного обеспечения.

Nytrebych O.O. Models and methods of software reliability assessment considering its architecture. – Manuscript.

Theses for Ph.D degree on technical sciences in specialty 01.05.03 – «Mathematical and software support of computer machines and systems». – Lviv Polytechnic National University, Ministry of Education and Science of Ukraine, Lviv, 2015.

An actual scientific problem of software reliability assessment based on architectural approach and consideration of inter-component execution dependencies, which allows increasing software reliability assessment accuracy and improving quality assurance process, has been solved.

The model for software reliability assessment has been improved using high-order Markov chains. This allows considering dependency of control transfer between components and reflects the execution nature of software with complex architecture. The software reliability assessment method has been developed. It includes next main steps: every component reliability calculation, calculation of component transition probability matrix, estimation of optimal order Markov chain using informational criteria and assessing software reliability with improved model. Software reliability assessments using first and higher order models have been conducted and results compared, showing that usage of Markov chain in software reliability analysis problems improves assessment accuracy to 6%. For quick collection of required parameters of software reliability assessment model a design pattern has been developed. This design pattern usage allows dynamically changing the process order and shortening RAM utilization.

Due to the fact that quality assurance is currently the primary way to ensure software reliability, the method of test cases generation using "black-box" and "white-box" strategies has been developed. Software test cases are based on new software usage model, which considers software variables set and their equivalence classes. Variables values set usage is required to cover the code variables parameters metric. This metric reflects consideration of all variables values and its high value shows that software is reliable. In addition, dependencies between discovered software failures and software parameters (number of components, average component variables number, components connectivity and failures complexity) were researched and giving background for providing practical recommendations to quality assurance leads. Also, efficacy of developed method for software failures search in comparison with known methods has been analyzed. Developed method of test cases generation improves failures discovery efficacy by 7–10%.

Based on developed methods and models the software system for software reliability assessment has been created. It consists of three independent modules: module for software usage model creation, module for test cases generation using developed software usage model, and a module for software reliability assessment using high-order Markov chain theory. Information model of created system has been described using UML class diagrams, and algorithms for such system implementation have been developed.

Main results of this work are already used in industry during software quality assurance, and in educational process.

Key words: software reliability, software reliability assessment model, higher order Markov chain, test case generation, software life cycle.