Міністерство освіти і науки України
Національний університет "Львівська політехніка"

На правах рукопису

ЯКОВИНА ВІТАЛІЙ СТЕПАНОВИЧ

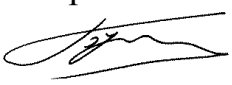
УДК 004.052

МЕТОДИ ТА ЗАСОБИ АНАЛІЗУ НАДІЙНОСТІ ФУНКЦІОНУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З УРАХУВАННЯМ ЕТАПІВ ЖИТТЄВОГО ЦИКЛУ

01.05.03 – математичне та програмне забезпечення
обчислювальних машин і систем

Дисертація на здобуття наукового ступеня доктора технічних наук

Науковий консультант
доктор технічних наук, професор
Федасюк Дмитро Васильович

Ідентичність цього примірника дисертації засвідчую:
Вчений секретар спеціалізованої
вченої ради,
доктор технічних наук, професор

/Р.А. Бунь/

Львів – 2015

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АЗ – апаратне забезпечення

АОП – аспектно-орієнтоване програмування

АФСТ – автоматизоване формування сценаріїв тестування

БД – база даних

ЕОМ – електронна обчислювальна машина

ЖЦ – життєвий цикл

ЛМВП – ланцюг Маркова вищого порядку

ММП – метод максимальної правдоподібності

МНПС – модель надійності з показником складності

НМ – нейронна мережа

НПП – неоднорідний пуассонів процес

ООП – об'єктно-орієнтоване програмування

ПАС – програмно-апаратна система

ПЗ – програмне забезпечення

ПС – програмна система

AIC – (англ. Akaike Information Criterion), інформаційний критерій Akaike

BIC – (англ. Bayesian Information Criterion), байєсівський інформаційний критерій

UML – (англ. Unified Modeling Language), уніфікована мова моделювання

ВСТУП

Актуальність теми. В сучасну епоху постіндустріального суспільства ринок програмного забезпечення (ПЗ) щороку зростає стрімкими темпами. Так, згідно звіту Gartner за січень 2015, у 2014 році ринок промислового ПЗ (Enterprise software) зріс на 5,8% порівняно з попереднім роком (більше ніж будь-яка інша галузь IT індустрії) і становив 317 млрд. доларів США. Зростання цього ринку прогнозується і на наступні періоди незважаючи на сповільнення економічного зростання IT галузі в цілому, так на 2015 рік прогнозувалося зростання ринку промислового ПЗ ще на 5,5% порівняно з попереднім роком до 335 млрд. доларів США. Частка бюджету розроблення ПЗ, витрачена на забезпечення якості (QA), за даними Capgemini, зросла з 18% у 2012 році та 23% у 2013 до 26% у 2014 році з тенденцією до подальшого зростання до 29% у 2017. При цьому загальноприйнятою аксіомою серед розробників ПЗ є правило "1:10:100" – "помилка, вартість виправлення якої становить 1 долар на етапі аналізу вимог чи проектування, коштує 10 доларів на етапі тестування, а виправлення цієї помилки на етапі впровадження коштуватиме 100 доларів".

Більшість сучасної техніки, зокрема – електронні та телекомунікаційні пристрої, є програмно-апаратними системами (ПАС), процес функціонування яких полягає у взаємодії програмних і апаратних засобів. Особливого значення для їхнього проектування, виробництва та експлуатації набуває оцінювання та забезпечення заданих показників їх надійності внаслідок виконання ними все більш відповідальних функцій, пов'язаних із забезпеченням безпеки людини. Із зростанням складності як ПАС загалом, так і ПЗ зокрема, ускладнюється процедура достовірного і точного оцінювання їхніх показників надійності. Одна з основних суперечностей у розвитку сучасної техніки, описана ще Б.В. Гнєденком у 1960-х роках, полягає у тому, що "з одного боку зростаюча складність систем призводить до зниження їх надійності, а з іншого боку, висуваються жорсткіші вимоги до надійної роботи цих систем"; і ця суперечність тільки посилилась з розвитком сучасних ПАС. Отже, сучасний етап розвитку

ПАС характеризується чітко вираженою суперечністю між відповідальністю та складністю відповідного ПЗ – з одного боку, та методами і засобами оцінювання та прогнозування його надійності – з іншого. Усунути цю суперечність можна шляхом підвищення ступеня адекватності моделей надійності ПЗ на усіх етапах його життєвого циклу (ЖЦ), а також урахування впливу архітектури та складності ПЗ на процес оцінювання показників його надійності. При цьому на кожному етапі ЖЦ має використовуватись найбільш адекватна модель надійності ПЗ за рахунок уточнення та збільшення ступеня деталізації інформації про об'єкт дослідження.

Побудові математичних моделей та розвитку методів аналізу надійності як ПАС загалом, так і ПЗ зокрема, присвячено багато праць вітчизняних і закордонних вчених. Починаючи від перших моделей надійності ПЗ, розроблених Z. Jelinski та P. Moranda у 1972 році, опубліковано значну кількість наукових праць, у яких досліджено моделі надійності різних типів, серед яких можна виділити моделі на основі неоднорідного пуассонового процесу J.D. Musa, A.L. Goel та K. Okumoto, S. Yamada та M. Ohba, узагальнені моделі пуассонового процесу A. Goel, A.L. Goel та K. Okumoto, компонентні моделі B. Littlewood, M.L. Shooman, S. Gokhale, R.C. Cheung та ін. У роботах В.С. Харченка розроблено методологію побудови гарантоздатних систем відповідального призначення, узагальнену класифікацію імовірнісних моделей надійності ПЗ та метод вибору оптимальної моделі на основі аналізу їх припущень. Д.А. Маєвський у своїх працях розвиває теорію динаміки програмних систем та детерміністичні моделі надійності ПЗ. Проблемам тестування, інженерії та оцінювання якості та надійності ПЗ присвячено ряд робіт П.І. Андона та К.М. Лавріщевої. Моделі оцінювання та засоби підвищення надійності ПЗ протягом усього ЖЦ розглядаються в роботах М.В. Дідковської та Ю.О. Тимошенко. Проблеми підвищення ефективності діагностування комп'ютерних систем на етапі експлуатації досліджуються у роботах О.В. Поморової. M. Lyu розробив концепцію інженерії ПЗ. Проблеми зв'язку

моделей надійності ПЗ з процесами прийняття рішень при його виробництві розглядаються в роботах H. Pham.

Всі розглянуті моделі надійності ПЗ в якості вхідних даних використовують результати тестування ПЗ, через що значно збільшують вартість виправлення помилки. Крім того, питання зв'язку етапів ЖЦ ПЗ, його складності функціонування та архітектури з надійністю залишаються малодослідженими. Тому вдосконалення наявних і побудова нових математичних моделей надійності ПЗ, які враховували б його складність, архітектуру та етапи ЖЦ, а також розроблення відповідних методів і засобів аналізу надійності функціонування ПЗ, є актуальною науково-прикладною проблемою, результати реалізації якої дадуть можливість підвищити достовірність оцінювання показників надійності сучасного ПЗ, чому і присвячена дана дисертаційна робота.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконана в межах наукового напряму "Технології та засоби розробки програмних продуктів і систем", визначеного Кабінетом Міністрів України в переліку пріоритетних тематичних напрямів наукових досліджень і науково-технічних розробок на період до 2015 року, відповідно до пріоритетного тематичного напряму Національного університету "Львівська політехніка" "Нові інтелектуальні, комп’ютерні, радіоелектронні, інфокомунікаційні вимірювальні технології, системи, пристрої та бортові системи космічних апаратів" і наукового напрямку кафедри програмного забезпечення "Програмне та математичне забезпечення автоматизованих систем". Результати, викладені в дисертаційній роботі, отримано під час виконання таких науково-дослідних робіт:

- грант Національного університету "Львівська політехніка" для молодих вчених "Використання нейронних мереж для задачі криптоаналізу алгоритму симетричного шифрування DES" (номер держреєстрації 0108U004258), який виконували у 2008 році; дисертантом розроблено нейромережеві моделі прогнозування часових рядів та архітектуру і принципи функціонування програмного засобу прогнозування часових рядів.

- "Розробка методів та засобів розподілення обчислень в задачах теплового проектування електронних пристроїв нового покоління" (номер державної реєстрації 0108U000331), яка виконувалась у 2008–2009 роках; дисертантом розроблено архітектуру, алгоритм функціонування та інформаційне забезпечення програмного засобу оцінювання показників надійності ПЗ.
- "Розроблення моделей, методів та алгоритмів для автоматизованої оцінки показників надійності радіоелектронних та електромеханічних пристроїв і систем" (номер державної реєстрації 0110U001098), яка виконувалась у 2010–2012 роках; дисертантом розроблено модель і метод оцінювання показників надійності ПЗ з урахуванням його складності на основі неоднорідного пуассонового процесу, засоби підтримки прийняття рішень при створенні ПЗ на основі критерію достатності процесу тестування та удосконаленої процедури прогнозування кількості помилок ПЗ.
- "Розроблення моделей, методів та засобів оцінювання та аналізу надійності програмного забезпечення" (номер державної реєстрації 0113U003185), яка виконувалась у 2013–2014 роках; дисертантом розроблено модель надійності ПЗ на основі марковського процесу вищого порядку з дискретним часом, методи автоматизованого формування сценаріїв тестування ПЗ та рекомендації щодо вибору стратегії тестування при створенні програмних засобів.
- "Розроблення моделей надійності, ризику та безпечності програмно-апаратних технічних систем" (номер державної реєстрації 0113U001371), яка виконувалась у 2013–2015 роках; дисертантом розроблено моделі надійності ПЗ на основі марковського процесу вищого порядку з неперервним часом, узагальнений метод аналізу надійності ПЗ з урахуванням його складності, структури та етапу життєвого циклу, підходи до аналізу надійності програмно-апаратних систем з урахуванням відмов та збоїв ПЗ.
- проект за програмою Єврокомісії Tempus "Національна освітня інфраструктура удосконалення інноваційної та підприємницької діяльності

IT-студентів" (530576-TEMPUS-1-2012-1-SE-TEMPUS-SMHES), який виконується з 2012 року; дисертантом розроблено програмні засоби оцінювання та прогнозування відмов ПЗ, здійснено тестування та аналіз надійності програмного засобу "Віртуальне інноваційне середовище".

Мета і задачі дослідження. *Метою* роботи є підвищення достовірності оцінювання показників надійності функціонування програмного забезпечення на різних етапах його життєвого циклу за рахунок використання адекватних моделей, методів і засобів аналізу.

Для досягнення поставленої мети необхідно вирішити такі основні *задачі*:

1) аналіз відомих підходів до оцінювання та прогнозування показників надійності ПЗ, встановлення їх переваг і недоліків, які дадуть змогу визначити резерви підвищення їх достовірності при оцінюванні сучасного складного та багатокомпонентного ПЗ;
2) розроблення моделей і методів оцінювання показників надійності ПЗ з урахуванням його складності за результатами процесу тестування;
3) розроблення моделей і методів оцінювання показників надійності складних програмних систем на основі даних про поведінку надійності складових, з урахуванням їх архітектури та складності функціонування;
4) створення методу аналізу надійності функціонування ПЗ на основі розроблених моделей та методів з урахуванням складності, архітектури та етапів життєвого циклу ПЗ;
5) розроблення засобів підтримки прийняття рішень на етапах тестування та експлуатації ПЗ із урахуванням вимог до його надійності;
6) розроблення методів автоматизованого формування сценаріїв тестування ПЗ та рекомендацій щодо вибору стратегії тестування при створенні ПЗ;
7) удосконалення непараметричних моделей прогнозування показників надійності ПЗ на етапах його тестування та експлуатації на основі нейронних мереж;

8) розроблення інформаційного забезпечення та програмних засобів для оцінювання та прогнозування показників надійності ПЗ на основі отриманих теоретичних результатів;
9) верифікація розробленого математичного та програмного забезпечення.

Об'єктом дослідження є аналіз надійності функціонування ПЗ.

Предметом дослідження є моделі та методи оцінювання показників надійності функціонування ПЗ з урахуванням його архітектури, складності та процесів життєвого циклу.

Методи дослідження. Для вирішення поставлених у дисертаційній роботі завдань використано такі методи дослідження: методи теорії ймовірностей та математичної статистики – для побудови і аналізу моделей надійності функціонування ПЗ типу "чорної скриньки", методів автоматизованого формування сценаріїв тестування ПЗ, створення засобів підтримки прийняття рішень у процесі розроблення ПЗ; теорію марковських процесів – для побудови компонентних моделей надійності ПЗ; методи обчислювальної математики – для оцінювання показників надійності ПЗ та розв'язування систем рівнянь Колмогорова – Чепмена та системи рівнянь, отриманої за методом максимальної правдоподібності; методи теорії графів – для створення методу подання марковського процесу вищого порядку у вигляді еквівалентного процесу першого порядку; методи системного аналізу – для розроблення засобів аналізу надійності функціонування ПЗ на ранніх етапах ЖЦ; методи штучного інтелекту – для створення засобів прогнозування відмов ПЗ на основі штучних нейронних мереж (НМ); теорію алгоритмів, аспектно- та об'єктно-орієнтовану парадигми – для розроблення програмних засобів. Для опрацювання експериментальних даних використано сучасні програмні засоби обчислення (пакети MathCad та Statistica).

Наукова новизна одержаних результатів. Наукова новизна результатів дисертаційної роботи полягає у вирішенні науково-прикладної проблеми створення моделей, методів та засобів аналізу надійності функціонування ПЗ з урахуванням впливу етапів його ЖЦ, архітектури та складності, які підвищують

достовірність оцінювання показників його надійності. При цьому отримано такі нові наукові результати:

вперше:

1) побудовано модель надійності функціонування програмних систем у вигляді ланцюга Маркова вищого порядку з неперервним часом та модель надійності програмної системи з урахуванням недосконалої інтеграції її модулів та взаємозалежності їх виконання, які, на відміну від наявних моделей, дають можливість більш достовірно оцінювати показники надійності ПЗ;
2) розроблено метод аналізу надійності функціонування ПЗ, який, на відміну від існуючих, враховує його складність, архітектуру та етапи життєвого циклу, що підвищує достовірність оцінювання показників надійності ПЗ;
3) розроблено метод подання марковського процесу вищого порядку у вигляді еквівалентного процесу першого порядку з віртуальними станами для програмних систем, в яких існує взаємозалежність виконання модулів, що дає можливість застосувати наявні засоби автоматизованої побудови моделей надійності функціонування складних програмних систем;
4) розроблено метод прогнозування кількості відмов програмного забезпечення з використанням регресійного аналізу, який дає можливість підвищити точність прогнозування кількості невиявлених помилок в програмному забезпеченні, або ж зменшити тривалість процесу тестування зі збереженням точності прогнозу;

удосконалено:

5) процес аналізу вимог до ПЗ, який за рахунок використання методу аналізу ієрархій дає можливість отримати оцінки параметрів моделей надійності функціонування ПЗ на ранніх етапах його життєвого циклу;
6) нейромережеві моделі прогнозування надійності функціонування ПЗ, зокрема, встановлено оптимальні конфігурації нейронної мережі Елмана та мережі на основі радіально-базисних функцій, що дає можливість прогнозувати відмови ПЗ різного ступеня складності з високою точністю;

отримали подальший розвиток:

7) модель надійності функціонування ПЗ з показником складності, зокрема, встановлено зв'язок значень цього показника зі складністю ПЗ, що дало можливість підвищити достовірність оцінювання показників надійності ПЗ;
8) методи аналізу надійності програмно-апаратних систем, зокрема, встановлено, що введення в модель надійності функціонування ПЗ показника його складності, за рахунок узагальненого характеру моделі та урахування складності ПЗ, підвищує достовірність оцінювання показників надійності програмно-апаратних систем.

Практичне значення одержаних результатів. Практичне значення дисертаційної роботи полягає у розробленні обчислювальних алгоритмів і програмних засобів, які забезпечують достовірне оцінювання показників надійності ПЗ на різних етапах його життєвого циклу, зокрема:

1) узагальнений метод аналізу надійності ПЗ дає змогу визначити шляхи використання розроблених моделей надійності в практиці програмної інженерії залежно від складності розроблюваних програмних засобів і характеристик процесів ЖЦ;
2) розроблені моделі надійності ПЗ надають вхідні дані розробникам ПАС у вигляді відповідних показників надійності ПЗ, що дає змогу здійснювати надійнісне проектування сучасних ПАС, зокрема телекомунікаційних і радіоелектронних;
3) розроблена модель надійності ПЗ з показником складності, а також метод оцінювання та прогнозування надійності на її основі придатні для використання на практиці для оцінювання показників надійності модулів складних програмних систем, або програмних засобів без урахування їх внутрішньої структури на основі результатів тестування;
4) розроблені моделі надійності ПЗ на основі ланцюгів Маркова вищого порядку придатні для оцінювання показників надійності складних програмних систем на основі їх архітектури та показників надійності їх модулів;

5) критерій достатності процесу тестування та метод прогнозування кількості невиявлених помилок ПЗ дають можливість керівникам програмних проектів підвищити ефективність процесу тестування ПЗ шляхом використання числових метрик для планування розподілу ресурсів протягом цього етапу ЖЦ, а також підвищити достовірність оцінювання показників надійності ПЗ;
6) засоби автоматизованого формування сценаріїв тестування дають можливість підвищити ефективність процесу тестування ПЗ за рахунок рівномірного покриття коду програми тестами та обґрунтованого вибору стратегії тестування, що зменшує часові, фінансові та людські ресурси на етапі тестування ПЗ;
7) розроблені програмні засоби для оцінювання показників надійності ПЗ та прогнозування його відмов можуть бути використані в компаніях з розроблення та експлуатації ПЗ та ПАС відповідального призначення, що підтверджується відповідними актами.

Результати роботи та розроблені програмні засоби пройшли дослідне випробування та впровадження на підприємствах ДП НДІ "Система", ТзОВ "Сайпресс Семікондактор Україна", ТзОВ науково-виробнича фірма "Промтехносервіс Україна", ТзОВ "Логіка ЛТД", ТОВ "СЕА Електротехніка", ПП "Лінк Ап Студіо", а також впроваджені у навчальному процесі Національного університету "Львівська політехніка" в курсах лекцій з дисциплін "Якість програмного забезпечення та тестування", "Аналіз вимог до програмного забезпечення" та "Основи теорії надійності програмних систем", що підтверджено відповідними актами.

Особистий внесок здобувача. Усі наукові результати дисертаційної роботи отримані здобувачем особисто. У друкованих працях, опублікованих у співавторстві здобувачеві належать: підрозділи 2.3.2, 2.3.3, 6.1, 6.2.1, 6.3 [96]; розділ 3, підрозділи 1.3, 2.3.1, 2.4, 4.1, 4.3 [125]; побудова моделей надійності програмних систем вищого порядку з неперервним часом [57, 152, 154, 161, 164, 167]; обґрунтування вибору критерію структурної оптимізації [150, 151, 160]; структура та формалізація моделі надійності ПЗ з показником складності,

визначення діапазонів індексу складності ПЗ, дослідження впливу моделей надійності ПЗ на показники надійності ПАС [38, 66, 128, 132–135]; розроблення методів аналізу надійності ПЗ [129–131, 157, 159, 172, 199, 205–207]; проведення експериментальних досліджень з верифікації моделей та підвищення ефективності процесу тестування [153, 156, 158, 173, 198]; обробка статистичного матеріалу результатів тестування ПЗ [201, 202, 204, 260]; побудова нейромережевих моделей надійності ПЗ [256–259, 261]; розроблення алгоритмів і програмних засобів аналізу надійності ПЗ [155, 163, 203, 252–255].

Апробація результатів дисертації. Основні теоретичні положення та практичні результати дисертаційного дослідження доповідались на таких наукових конференціях: Міжнародній конференції молодих вчених "Комп'ютерні науки та інженерія" CSE (Львів, 2011, 2013); Міжнародній науково-технічній конференції "Комп'ютерні науки та інформаційні технології" CSIT (Львів, 2010–2012, 2014); Міжнародній конференції "Інтернет – Освіта – Наука" ІОН (Вінниця, 2008); Міжнародній конференції "Досвід розробки та застосування приладо-технологічних САПР в мікроелектроніці" CADSM (Львів–Поляна, 2009, 2011, 2013); Всеукраїнській науково-практичній конференції "Сучасні інформаційні технології в економіці, менеджменті та освіті" СІТЕМ (Львів, 2011, 2012); Міжнародній конференції "Перспективні технології і методи проектування МЕМС" MEMSTECH (Поляна, 2012, 2013; Львів, 2014); Міжнародній науковій конференції "Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту" ISDMCI (Євпаторія, 2012, 2013); International Conference on Inductive Modelling ICIM (Київ, 2013); Міжнародній конференції "Сучасні проблеми радіоелектроніки, телекомунікацій, комп'ютерної інженерії" TCSET (Львів–Славське, 2014); International Conference on ICT in Education, Research and Industrial Applications: Integration, Harmonization and Knowledge Transfer ICTERI (Львів, 2015); Міжнародній науково-технічній конференції "Електротехнічні та комп'ютерні системи: теорія і практика" (Одеса, 2015).

Публікації. За результатами дисертаційного дослідження опубліковано 55 наукових праць (з них 7 одноосібних), зокрема 2 монографії, 2 статті у закордонних виданнях, 4 статті у фахових виданнях України, що внесені до міжнародних наукометричних баз даних, 19 статей у інших фахових виданнях України, 2 статті у періодичних виданнях України, що не віднесені до переліку фахових, 26 публікацій у матеріалах наукових конференцій.

Структура та обсяг роботи. Дисертаційна робота складається зі вступу, п'яти розділів, висновків, списку використаних джерел з 283 найменувань на 31 сторінці та 4 додатків на 16 сторінках. Загальний обсяг дисертації – 325 сторінок, з них 278 сторінок основного тексту. Робота містить 91 рисунок і 24 таблиці.

РОЗДІЛ 1. СТАН НАЯВНИХ ПІДХОДІВ ДО ОЦІНЮВАННЯ НАДІЙНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В цьому розділі здійснено огляд та аналіз сучасного стану моделей, методів та засобів аналізу надійності ПЗ та тенденцій їх розвитку. Розглянуто поняття надійності ПЗ як групи властивостей, в яку входять безвідмовність, стійкість до аномалій, відновлюваність, точність та реактивність. Наведено визначення та причини відмов ПЗ. Здійснено порівняльний аналіз надійності програмних та апаратних засобів, показано найбільш суттєві відмінності та задачі аналізу надійності АЗ та ПЗ. Здійснено огляд та аналіз моделей надійності ПЗ, при цьому окрему увагу приділено моделям на основі НПП та моделям на основі компонентного підходу; показано основні обмеження та недоліки існуючих моделей та можливі шляхи їх усунення. Проаналізовано сучасний стан моделей і засобів визначення оптимального часу введення програмного продукту в експлуатацію, що на сьогоднішній день, є практично єдиними засобами підтримки прийняття рішень при розробленні ПЗ з урахуванням вимог до його надійності. Розділ завершується постановкою задачі дисертаційної роботи.

1.1. Поняття надійності ПЗ та її характеристики

Згідно ДСТУ 2844-94 програмою називають послідовність інструкцій, які може виконати ЕОМ, а програмним засобом є "взаємопов'язана сукупність програм, процедур, правил, документації та даних, що належить до функціонування обчислювальної системи" [1]. Надійністю функціонування ПЗ (reliability) називається "група властивостей ПЗ, що обумовлює спроможність ПЗ зберігати працездатність і перетворювати вихідні дані в шуканий результат в заданих умовах за встановлений період часу" [1, 2]. В цю групу властивостей, згідно [2] входять безвідмовність (maturity), яка характеризується частотою відмов через помилки та недосконалості ПЗ; стійкість до аномалій (fault tolerance), яка обумовлює здатність ПЗ виконувати свої функції в аномальних умовах під якими розуміють збої і відмови технічних засобів, помилки операторів та помилки у вихідних даних; відновлюваність (recoverability), яка

обумовлює можливість відновлювати рівень якості функціональності і дані після відмов; точність (accuracy), що характеризується подібністю результатів обробки даних до істинних, специфікованих або теоретично правильних значень; та реактивність (responsibility), яка характеризується здатністю своєчасно перетворювати вхідні дані (запити) в шуканий результат. Усі ці властивості ПЗ становлять одну з груп, які визначають якість ПЗ разом з функціональністю, зручністю використання, раціональністю, супроводжуваністю та переносимістю [2]. При цьому, згідно [1] під якістю ПЗ розуміють сукупність його властивостей щодо їх придатності задовольняти встановлені чи передбачені потреби відповідно до призначення ПЗ. Далі в цій роботі надійність ПЗ буде вживатись в термінах і визначеннях ДСТУ 2844-94.

Поняття відмови є фундаментальним поняттям теорії надійності і може бути означене як подія, після виникнення якої характеристики технічного об’єкту (параметри) виходять за допустимі межі [3]. Згідно [1] відмовою ПЗ є подія, під час якої проявляється непрацездатність ПЗ. Ознаки непрацездатності встановлюються в нормативно-технічній документації ПЗ. При цьому невідповідність результатів виконання ПЗ і сподіваних результатів може бути наслідком дефектів апаратних засобів чи наслідком помилок у ПЗ. Тут під помилкою ПЗ розуміють запис елемента програми чи тексту програмної документації, використання яких призводить чи може призвести до неправильного результату [1]. Таким чином, на відміну від АЗ, відмови ПЗ можуть бути спричинені неправильним алгоритмом, некоректною реалізацією алгоритму ("запис елемента програми"), некоректністю програмної документації, що матиме наслідком неправильні дії користувача. При цьому відмови ПЗ можуть залежати від даних, які в поточний момент обробляє програма. Крім того, відмови ПЗ можуть бути спричинені збоями АЗ внаслідок впливу зовнішніх факторів (іонізуючого випромінювання, температури тощо), і, в деяких випадках, можуть бути усунені шляхом перезавантаження ПЗ [4].

В загальному, за своїм характером в першому наближенні відмови поділяють на раптові та поступові [3, 5]. Поступові відмови виникають при

поступовій зміні параметрів, що визначають якість виробу (в основному в результаті старіння чи зношування), коли ці параметри виходять за межі встановлених допусків. Такі відмови не характерні для ПЗ внаслідок його природи. Раптові відмови визначаються різкою зміною параметрів, що визначають якість виробу. Крім того, для обчислювальної техніки характерними є збої, тобто відмови, що самоусуваються [5].

Оскільки, як уже зазначалось, надійність ПЗ визначається як група його властивостей, для кількісного оцінювання цих властивостей використовують критерії і показники надійності [3]. Критерієм називають ознаку (мірило), за якою оцінюють надійність. Показником надійності називають чисельне значення критерія. Показники задають в технічних вимогах на виріб, розраховують в процесі проектування, оцінюють в процесі випробування та експлуатації технічного об'єкту.

Математичний апарат теорії надійності розглядає відмову елемента як випадкову подію, а час ξ до її появи – як випадкову величину. Основною характеристикою надійності елементу будь-якої системи, в тому числі невідновлюваної, є функція розподілу тривалості його безвідмовної роботи $F(t) = P(\xi < t)$, визначена при $t \geq 0$. На її основі, в загальному, можуть бути отримані такі показники надійності [3] для невідновлюваних систем:

- $P(t)$ – ймовірність безвідмовної роботи протягом часу t;
- $Q(t) = 1 - P(t)$ ймовірність відмови протягом часу t;
- T_1 – середня тривалість безвідмовної роботи (середня тривалість напрацювання до відмови);
- $f(t)$ – густина розподілу тривалості безвідмовної роботи;
- $\lambda(t)$ – інтенсивність відмов в момент часу t;

Інтенсивністю відмов називається відношення густини розподілу до імовірності безвідмовної роботи об'єкта (1.1). Статистично інтенсивність відмов – це відношення кількості зразків техніки, що відмовили в одиницю часу до середньої кількості зразків, які справно працюють на інтервалі $[t, t + \Delta t]$.

Інтенсивність відмов $\lambda(t)$ є основним показником надійності елементів складних систем [3].

Між показниками надійності існують наступні залежності [3]:

$$\lambda(t) = \frac{f(t)}{P(t)}, \tag{1.1}$$

$$P(t) = e^{-\int_0^t \lambda(t)dt}, \tag{1.2}$$

$$P(t) = \int_t^\infty f(t)dt, \tag{1.3}$$

$$T_1 = \int_0^\infty P(t)dt = \int_0^\infty tf(t)dt. \tag{1.4}$$

Показниками надійності відновлюваних елементів та систем, якими в загальному є програмні засоби, можуть бути також показники надійності невідновлюваних елементів [3]. Це має місце в тих випадках, коли система, до складу якої входить елемент, не підлягає ремонту за умовами її роботи. Надійність відновлюваних об'єктів оцінюють наступними показниками [3]:

- T – середня тривалість роботи між відмовами (середнє напрацювання на відмову);
- $T_\text{В}$ – середня тривалість відновлення;
- $\omega(t)$ – параметр потоку відмов;
- $\text{К}_\Г(t)$ – функція готовності: ймовірність того, що система справна в момент t ;
- $\text{К}_\П(t)$ – функція простою: ймовірність того, що в момент t система несправна і відновлюється;
- $\text{К}_\Г$ – коефіцієнт готовності: ймовірність того, що система буде справною при тривалій експлуатації (стаціонарний режим);
- $\text{К}_\П$ – коефіцієнт простою: ймовірність того, що система буде несправною при тривалій експлуатації.

Параметром потоку відмов $\omega(t)$ називають похідну (швидкість зміни) середньої кількості відмов об'єкта в момент t. Статистично параметр потоку відмов визначають як відношення кількості зразків техніки, що відмовили в одиницю часу до кількості зразків, що поставили на випробування, за умови, що зразки, які відмовили, заміняють справними чи відремонтованими [3]. Параметр потоку відмов володіє наступними властивостями [3]:

- у випадку експоненційного закону тривалості роботи об'єкта з параметром λ і миттєвого відновлення $\omega(t) = \lambda$;
- при миттєвому відновленні границя, до якої прямує параметр потоку відмов при $t \to \infty$, дорівнює величині, оберненій до середньої тривалості безвідмовної роботи, тобто $\lim\limits_{t\to\infty} \omega(t) = \frac{1}{T}$;
- при миттєвому відновленні параметр потоку відмов і густина розподілу часу до відмови пов'язані наступним інтегральним рівнянням Вольтерра другого роду:

$$\omega(t) = f(t) + \int_0^t \omega(\tau) f(t-\tau) d\tau. \qquad (1.5)$$

Рівняння (1.5) встановлює залежність між показниками надійності відновлюваної та невідновлюваної техніки. Воно дає змогу за статистичними даними про відмови відновлюваної техніки в процесі її експлуатації визначити показники надійності невідновлюваної техніки [3].

1.2. Порівняльна характеристика надійності апаратних та програмних засобів

З початком дослідження надійності ПЗ, теоретики і практики в галузі надійності обговорювали питання надійності ПЗ у порівнянні з надійністю апаратного забезпечення з точки зору подібності, відмінностей, технік моделювання і т.д. [6, 7]. Оскільки основи технік моделювання надійності ПЗ запозичені з теорії надійності, розробленої для складних технічних систем за останні 40 років, порівняння надійності програмного та апаратного забезпечення

може допомогти у використанні цих методик і в програмно-апаратних системах. В табл. 1.1 наведено порівняльну характеристику надійності програмного та апаратного забезпечення [8, 9].

Матеріал обладнання зношується з часом, тому календарний час є загальноприйнятим аргументом функції надійності апаратного забезпечення. Натомість у випадку програмного забезпечення відмова ніколи відбудеться, якщо програма не використовується. Тому в контексті надійності ПЗ, більш адекватно інтерпретувати час як навантаження на ПЗ, або обсяг робіт, виконаних ПЗ. Тут слід зауважити про важливість урахування метрики покриття тестами програмного коду та шляхів виконання програми: статистику відмов ПЗ слід досліджувати при виконанні певної вибірки робіт/завдань з різними вхідними даними. Виконання будь-якої кількості завдань з однаковими вхідними даними (якщо вони перший раз не призвели до відмови) не призведе до відмови ПЗ. Так, згідно [10], кількість та характер відмов, які стосуються ПЗ, є наслідком його внутрішніх помилок та залежать від умов його застосування. В якості аргументу функції надійності ПЗ можуть використовуватись наступні одиниці часу [8]:

- час виконання (англ. execution time) – процесорний час; час, протягом якого процесор зайнятий;
- час роботи (англ. operation time) – час, протягом якого використовується програмне забезпечення;
- календарний час – показник, який використовується для програмного забезпечення, що працює 24 години на добу;
- запуск (прогін) програми (англ. run) – робота, представлена процесору;
- інструкція (англ. instruction) – кількість виконаних інструкцій коду;
- шлях (англ. path) – послідовність виконання вхідних даних.

Моделі на основі часу виконання, часу роботи, календарного часу та виконаних інструкцій належать до моделей в часовій області. Моделі на основі прогону програми і шляху належать до моделей в області вхідних даних [8].

Таблиця 1.1.

Порівняння надійності апаратного та програмного забезпечення.

Надійність ПЗ	**Надійність АЗ**
Без урахування еволюції програми, інтенсивність відмов статистично не зростає.	Інтенсивність відмов має вигляд U-подібної кривої. Ділянка початкового напрацювання подібна до етапу відлагодження програми.
Відмова не відбувається, якщо ПЗ не експлуатується.	Старіння матеріалу може спричинити відмову навіть якщо система не експлуатується.
Предметом дослідження є механізм відмов.	Механізм відмов можна трактувати як чорну скриньку.
Процесорний час і прогін програми – два найбільш популярні аргументи функції надійності.	Календарний час є загальноприйнятим аргументом функції надійності.
Більшість моделей аналітично виводяться з припущень. Акцент ставиться на розробці моделі, поясненні припущень моделі, фізичному змісту параметрів.	Дані про відмови описують деяким розподілом. Вибір відповідного розподілу базується на аналізі даних про відмови та досвіді дослідника. Акцент ставиться на аналіз даних про відмови.
Відмови спричинені некоректною логікою, неадекватним вибором операторів мови програмування, чи некоректними вхідними даними. Це схоже на помилку в проектуванні складної апаратної системи.	Відмови спричинені зношуванням матеріалу, випадковими відмовами, помилками в проектуванні, неправильним використанням, навколишнім середовищем тощо.

Надійність ПЗ може бути покращена шляхом збільшення зусиль на тестування та виправлення виявлених помилок. Надійність, як правило, постійно змінюється протягом етапу тестування через внесення помилок в новому коді або видалення існуючих помилок.	Надійність АЗ можна покращити за рахунок надійнісного проектування (на етапах системотехнічного і схемотехнічного проектування), удосконалення технологічного процесу та матеріалів, прискореним випробуванням об'єкту.
"Ремонт" ПЗ вимагає встановлення нової версії усього продукту чи його частини.	Відновлення АЗ вимагає відновлення початкових, оригінальних умов та стану чи заміни елементів системи на нові.
Модулі ПЗ рідко бувають стандартизовані.	Модулі АЗ зазвичай є стандартизованими.

Згідно [10] особливостями і відмінностями ПЗ (в сенсі надійності) від традиційних технічних систем є:

- не для всіх видів програм можна застосувати поняття і методи теорії надійності – їх можна використовувати тільки до програмних засобів, що функціонують у реальному часі і безпосередньо взаємодіють із зовнішнім середовищем;
- при розробленні та оцінюванні якості програмних модулів до них не можна застосувати поняття надійності функціонування, якщо при обробці інформації вони не використовують значення реального часу і не взаємодіють безпосередньо з зовнішнім середовищем;
- домінуючими факторами, що визначають надійність програм, є помилки при проектуванні і розробці, а фізичне руйнування програмних модулів при зовнішніх впливах має другорядне значення;

- відносно рідкісне руйнування програмних модулів і необхідність їхньої фізичної заміни, приводить до принципової зміни понять помилки і відмови, і до поділу їх за тривалістю відновлення відносно деякого припустимого часу простою для функціонування інформаційної системи;
- для підвищення надійності програмних комплексів особливе значення мають методи автоматичного скорочення тривалості відновлення і перетворення відмов у короткочасні збої, шляхом введення в програмні засоби часової, програмної та інформаційної надлишковості;
- непередбачуваність місця, часу і ймовірності прояву помилок, а також їхнє нечасте виявлення при реальній експлуатації досить надійних програмних засобів, не дає можливості ефективно використовувати традиційні методи апріорного розрахунку показників надійності складних систем, орієнтовані на стабільні, вимірювані значення надійності складових модулів;
- традиційні методи форсованих випробувань надійності систем шляхом фізичного впливу на їхні модулі не можна застосувати до програмних засобів, їх варто замінити на методи форсованого впливу інформаційних потоків зовнішнього середовища.

Аналіз надійності АЗ в загальному полягає у розв'язанні двох задач:

1) визначення зі статистичних даних значення показників надійності елементів, і зокрема інтенсивності відмов $\lambda(t)$;
2) визначення показників надійності технічних систем (головним чином функції та коефіцієнту готовності) на основі відомостей про їх структуру, поведінку та значення інтенсивностей відмов елементів системи.

Для вирішення першої задачі використовують переважно добре опрацьовані методи математичної статистики [3, 5], а для другої використовують

і розробляють такі методи аналізу надійності, засновані на використанні теорії ймовірностей, логіко-імовірнісні методи, топологічні методи, методи, засновані на теорії марковських процесів, методи статистичного моделювання тощо [3].

Окремою важливою задачею, що стоїть перед розробниками складних технічних систем, є надійнісне проектування, при якому вимоги до надійності системи ураховують ще на етапі її системотехнічного проектування [11].

З іншого боку потреба розробників у систематичних, відтворюваних інженерних підходах при створенні надійних програмних продуктів привела до появи концепції інженерії надійності ПЗ (англ. Software Reliability Engineering) [12, 13], яка визначається як кількісне дослідження поведінки функціонування ПАС по відношенню до вимог стосовно надійності [12]. Відповідно, інженерія надійності ПЗ включає [12]:

1) вимірювання надійності ПЗ, яке включає оцінювання та прогнозування на основі моделей надійності ПЗ;
2) атрибути та метрики проектування ПЗ, процесу розробки ПЗ, архітектури системи, середовище функціонування ПЗ та їх вплив на надійність;
3) застосування цих знань під час специфікації та проектування архітектури ПЗ, процесу розробки, тестування, впровадження, експлуатації та супроводу.

Розроблення методів і засобів інженерії надійності ПЗ є актуальною задачею в галузі комп'ютерної інженерії, програмної інженерії та суміжних дисциплін на даний момент та найближче десятиліття [12, 13].

Підсумовуючи, можна зазначити, що у зв'язку з подальшим розвитком індустрії програмного забезпечення та технологічної бази його розробки, ключові причини появи помилок в ПЗ виявляються на більш ранніх етапах життєвого циклу [14], а тому зростає увага до розроблення та застосування моделей надійності ПЗ на усіх етапах ЖЦ [15–17], що виражається у розвитку інженерії надійності ПЗ [13, 18] та засобів оцінювання надійності ПАС [19, 20].

1.3. Класифікація моделей надійності ПЗ

Протягом останніх десятиліть для оцінки якості та надійності ПЗ було запропоновано та досліджено значну кількість аналітичних моделей. Кожна модель базується на певних припущеннях про процес розробки та середовище функціонування ПЗ. Середовище функціонування відрізняється від середовища тестування програми, а також може змінюватися для кожного окремого програмного засобу, процесу розробки життєвого циклу, а також залежно від технічних можливостей команди проекту [21]. Таким чином, для розробників і користувачів програмних продуктів важливо бути знайомими з усіма відповідними моделями з метою прийняття обґрунтованих рішень стосовно якості ПЗ.

У дослідженнях надійності апаратного забезпечення, механізм появи відмов часто розглядають як чорну скриньку, а предметом дослідження є статистичні характеристики і параметри відмов (постійних і тимчасових) [3, 8]. При розгляді надійності ПЗ дослідників та інженерів насамперед цікавить механізм відмови [8, 9]. Переважна більшість моделей надійності ПЗ є аналітичними моделями, які отримані з припущень про механізми появи відмов. У таких дослідженнях основна увага приділяється припущенням, на основі яких розробляється модель, та інтерпретації її параметрів.

З метою створення адекватної моделі надійності ПЗ та у процесі прийняття обґрунтованих рішень на основі такої моделі, необхідне глибоке розуміння процесів, методологій та технологій створення, тестування ПЗ, етапів та механізмів внесення помилок, їх типів, факторів зовнішнього середовища, які дозволяють верифікувати правильність припущень та обмежень моделі, визначити область адекватності моделі в даному середовищі користувача [8–10, 22, 23].

Подібно до надійності апаратури, надійність ПЗ на часовому інтервалі характеризується ймовірністю безвідмовної роботи протягом певного періоду часу за певних умов [3, 9]. В результаті виконання програми, стан входу

перетворюється на стан виходу. Таким чином програма може розглядатись як функція *f*, яка перетворює простір входу в простір виходу, де простір входу – це множина всіх станів входу, а простір виходу – множина всіх станів виходу. Стан входу можна визначити як сукупність вхідних змінних чи типові команди/транзакції над програмою [8].

Програма створюється для виконання певних функцій. Відмовою ПЗ можна вважати ситуацію, коли фактичні вихідні дані відрізняються від очікуваних вихідних даних [8, 9]. Слід зауважити, що визначення відмови є різним для кожного програмного продукту і повинно бути чітко визначене у специфікації [8]. Так, наприклад, час відгуку програми, який становить 30 с., може вважатись відмовою для системи управління повітряним транспортом, але є допустимим для системи бронювання авіаквитків. Дійсно, згідно [1] відмовою ПЗ є "подія, під час якої проявляється непрацездатність ПЗ", а "ознаки непрацездатності встановлюються в нормативно-технічній документації ПЗ".

Помилка у випадку програмного забезпечення – це некоректна логіка, некоректна команда чи невідповідна команда, яка під час виконання може спричинити відмову [9, 23]. Згідно [1] помилка ПЗ, це "запис елемента програми чи тексту програмної документації, використання яких призводить чи може призвести до неправильного результату". Іншими словами, помилка – це джерело відмов, а відмови – це реалізація помилок. Коли відбувається відмова, їй відповідає деяка помилка в програмі, але наявність помилки може не спричиняти відмови і програма ніколи не вийде з ладу поки помилкові записи (оператори, команди, виклики функцій тощо) не будуть виконані. Таким чином, на відміну від апаратного забезпечення, статистика відмов ПЗ повинна враховувати сценарії використання та покриття програмного коду тестами, в іншому випадку аналіз надійності ПЗ може призвести до некоректних результатів, навіть з використанням адекватного математичного апарату.

Життєвий цикл (ЖЦ) програмного забезпечення в загальному зазвичай поділяють на такі етапи [24]: специфікація вимог, проектування, кодування, тестування, експлуатація і супровід. Етап проектування може включати

попереднє проектування і деталізоване проектування. Етап тестування може включати модульне, інтеграційне та регресійне тестування, тестування в умовах експлуатації. Етап супроводу може включати один або два підцикли, кожен з яких має всі етапи стадії розробки. Згідно ДСТУ 3918-1999 (ISO/IEC 12207:1995) [24] основними процесами життєвого циклу ПЗ є: процес замовлення, процес постачання, процес розробки, процес експлуатації та процес супроводу. В свою чергу процес розробки ПЗ включає наступні дії: реалізацію процесу, аналіз системних вимог, проектування архітектури системи, аналіз вимог до ПЗ, проектування архітектури ПЗ, розробку детального проекту ПЗ, кодування і тестування ПЗ, інтеграцію ПЗ, кваліфікаційне тестування ПЗ, системну інтеграцію, кваліфікаційне тестування системи, встановлення ПЗ, та забезпечення приймання програмного забезпечення [24].

На ранніх стадіях ЖЦ ПЗ потрібна модель прогнозування надійності, оскільки дані про відмови відсутні. Моделі такого типу призначені для передбачення кількості помилок в програмі перед тестуванням, і в деяких літературних джерелах [25] відносяться до детерміністичних (статичних) моделей надійності ПЗ. На етапі тестування надійність ПЗ покращується завдяки відлагодженню. Модель зростання надійності потрібна для оцінки поточного рівня надійності, часу і ресурсів, потрібних для досягнення заданого рівня надійності. Впродовж цього етапу оцінка надійності базується на аналізі даних відмов. Моделі такого типу відносять [25] до імовірнісних (динамічних) моделей надійності ПЗ. Після введення програми в експлуатацію при визначенні її надійності повинні враховуватись додавання нових модулів, усунення старих модулів, усунення виявлених помилок, поєднання нового коду з попередньо написаним кодом, зміна середовища користувача, зміна апаратного забезпечення тощо. На цьому етапі потрібна еволюційна модель надійності [8].

Моделі, в основі яких лежить підрахунок відмов (динамічні моделі) припускають, що, концептуально, в програмі наявна скінченна кількість помилок. Враховуючи, що кількість помилок є цілим числом, динамічні моделі обчислюють кількість початкових несправностей на етапі відлагодження

програми і кількість помилок, що залишились під час чи вкінці етапу відлагодження. Моделі підрахунку відмов використовують інтенсивність відмов як основну характеристику появи відмови. Залежно від типу, моделі припускають, що інтенсивність відмов кожної помилки є або сталою функцією часу відлагодження, або випадковою змінною із заданим законом розподілу [8–10, 22, 23, 25, 26]. Як тільки інтенсивність відмов, пов'язана з помилками певного типу визначена, інтенсивність відмов програми в цілому обчислюється як добуток кількості помилок, що залишились в програмі, на інтенсивність відмов, породжених помилкою кожного типу [8].

Під час етапу відлагодження кількість помилок, що залишились, змінюється. Один із способів моделювання процесу відмов є представлення кількості помилок, що залишились, як стохастичного процесу [8, 9, 26]. Переважна більшість моделей припускає ідеальне відлагодження – імовірність усунення помилки становить 100%, під час усунення помилки не вводиться нових, таким чином кількість помилок, що залишилась в програмі є незростаючою функцією тривалості відлагодження [8, 26]. В моделях, що припускають неідеальне відлагодження (помилки не усуваються, вводяться, чи не змінюються під час кожного відлагодження), кількість помилок, що залишилися, може зростати чи зменшуватись. Такий процес підрахунку помилок можна представити біноміальною моделлю, моделлю Пуассона, складним пуассоновим процесом, чи марковським процесом [9, 23, 25–32].

Моделі надійності ПЗ можуть бути розділені на два великі класи – детерміністичні (статичні) та імовірнісні (динамічні) [8, 9, 23, 25]. Разом з тим слід зауважити, що подальший поділ моделей за типами в кожному класі дещо відрізняється в різних працях. Так в [25] імовірнісні моделі поділяються на моделі на основі висівання помилок, інтенсивності відмов, апроксимації залежностей, неоднорідного пуассонового процесу, зростання надійності, на основі марковської структури. У [8, 9] загальний поділ зберігається, однак додаються моделі на основі архітектурного підходу, області вхідних даних, Баєсівські та уніфіковані моделі. Натомість у [26] динамічні моделі поділяють на

моделі на основі часу між відмовами, кількості відмов, висівання помилок та області вхідних даних. Крім того в багатьох роботах [15, 33–39] моделі на основі архітектурного підходу виділяють в окремий клас з подальшим поділом на типи, оскільки ці моделі, на відміну від попередніх, враховують архітектуру ПЗ, а не тільки статистику відмов. Класифікація моделей надійності програмного забезпечення за [25] наведена на рис. 1.1.

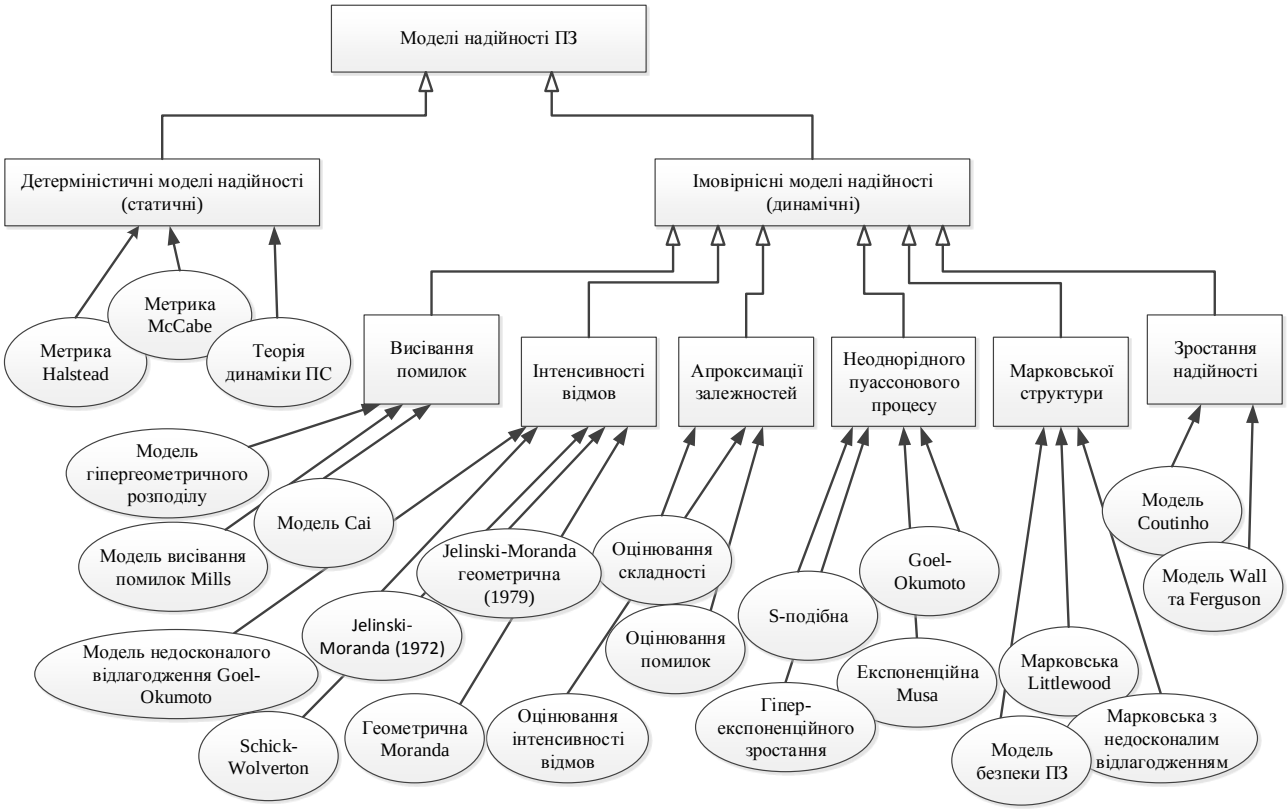


Рис. 1.1. Класифікація моделей надійності програмного забезпечення [25].

Детерміністичні моделі.

Моделі цього класу використовуються для дослідження:

1) елементів програми шляхом підрахунку кількості операторів, операндів та інструкцій;
2) потоку управління програми шляхом підрахунку розгалужень та трасування шляхів виконання;
3) потоку даних програми шляхом вивчення спільного використання даних та передачі даних.

Вимірювання продуктивності детерміністичного типу отримується з аналізу тексту вихідних кодів програми, і не включає жодної випадкової події чи величини. До детерміністичного класу відносять дві моделі [8, 25]: модель ПЗ Halstead, модель цикломатичної складності McCabe та модель на основі метрик складності [39]. У загальному ці моделі представляють кількісний підхід до вимірювання комп'ютерного програмного забезпечення. Модель ПЗ Halstead [40] використовується для оцінювання кількості помилок в програмі [8, 25], тоді як модель цикломатичної складності McCabe [41] використовується для визначення верхньої границі кількості тестів програми [8, 25]. Зауважимо, що обидві ці моделі є статичними за своєю природою, тобто вважають процеси в програмній системі незмінними з часом, а її надійність є виключно функцією метрик програмного засобу.

Принципово іншим підходом до вирішення проблем, притаманних сучасному етапу розвитку теорії надійності програмних засобів, є теорія динаміки програмних систем [27, 28, 42], основи якої були розроблені в [27, 28] як принципово новий детермінований підхід до визначення показників надійності з урахуванням впливу вторинних помилок. Динаміка програмних систем відрізняється від існуючої теорії надійності ПЗ тим, що вона базується на загальній теорії динаміки систем, а не на теорії ймовірностей, і розглядає процеси появи помилок в програмній системі не як випадковий процес, а як результат впливу детермінованих потоків помилок. Таким чином, ця теорія теж породжує детерміністичні моделі надійності, які, однак, не є статичними, на відміну від моделей Halstead та McCabe. Результати верифікації цієї моделі [42] на прикладі п'ятдесяти різноманітних програмних систем показали, що модель динаміки програмних систем на сьогоднішній день є однією з кращих за точністю оцінювання надійності ПЗ, та показує однаково точні результати для різних програмних засобів, а тому може вважатись універсальною моделлю [42].

Імовірнісні моделі.

Моделі цього класу представляють появу відмов та усунення помилок як випадкові події. Як зазначалось вище, в рамках цього класу можливий різний

поділ на типи, тому тут наведемо класифікацію за [8] як найбільш широку. Згідно [8] імовірнісні моделі можуть бути поділені на різні типи: (а) висівання помилок, (б) інтенсивності відмов, (в) апроксимації залежностей, (г) зростання надійності, (д) архітектурного підходу, (е) області вхідних даних, (ж) шляху виконання програми, (з) неоднорідного пуассонового процесу, (и) марковські, (і) баєсівські та уніфіковані.

Слід, однак, зазначити, що запропонована у [8, 25] множина класифікаційних ознак імовірнісних моделей не є достатньо повною і ортогональною. Натомість існують класифікаційні схеми, які мають розгалужену ієрархію ознак, і схеми матричного типу, описані у [43, 44]. Так, у [44] описано п'ять класифікаційних ознак моделей надійності ПЗ:

1) *час моделі* – визначається тим, який час застосовується в моделі: процесорний чи календарний;
2) *категорія* – визначається кількістю помилок у ПЗ (обмеженою чи необмеженою), які можуть бути виявлені за необмежений час;
3) *тип* – визначається розподілом кількості випадкових відмов у часі: найчастіше використовують розподіл Пуассона і біноміальний розподіл;
4) *клас* – визначається формою функції, що використовується для оцінки параметра потоку відмов (застосовується тільки для категорії "обмежених" моделей);
5) *сім'я* – визначається так само, як і клас, але тільки для "необмежених" моделей.

Узагальнену класифікацію імовірнісних моделей надійності ПЗ згідно наведених класифікаційних ознак зображено на рис. 1.2. На цьому рисунку не виділено ознаку "час моделі", оскільки вона відноситься, скоріше, до методики вимірювання часу, а не до методики побудови моделей [44].

<table>
<tr><td rowspan="2">Категорія</td><td colspan="2">Сім'я</td><td rowspan="2">Тип</td></tr>
<tr><td>Експоненціальні</td><td>Гамма-вейбулівські</td></tr>
<tr><td rowspan="2">Необмежені</td><td>Геометричні</td><td rowspan="2">Вейбулівські</td><td rowspan="3">Пуассонові</td></tr>
<tr><td>Логарифмічні</td></tr>
<tr><td rowspan="3">Обмежені</td><td>Неоднорідного пуассонового процесу</td><td rowspan="2">S-подібні</td></tr>
<tr><td>Гіперекспоненціальні</td></tr>
<tr><td>Деевтрофікаційні</td><td>Вейбулівські</td><td>Біноміальні</td></tr>
<tr><td rowspan="2">Категорія</td><td>Експоненціальні</td><td>Гамма-вейбулівські</td><td rowspan="2">Тип</td></tr>
<tr><td colspan="2">Клас</td></tr>
</table>

Рис. 1.2. Фасеточно-ієрархічна класифікація моделей надійності ПЗ [44].

Моделі на основі висівання помилок.

Основним підходом цього класу моделей є "висівання" відомої кількості помилок в програму, яка, вважається, має невідому кількість власних помилок. Після цього програма тестується і підраховується виявлена кількість висіяних та власних помилок. На основі цього отримують оцінку кількості помилок в програмі до висівання, яка використовується для отримання оцінок надійності ПЗ та інших відповідних характеристик. Найбільш популярною та найбільш фундаментальною моделлю цього класу є гіпергеометрична модель Mills [45].

Моделі на основі інтенсивності відмов.

Група моделей на основі інтенсивності відмов використовується для дослідження функціональних залежностей інтенсивності відмов на помилку (певного типу) та інтенсивності відмов програми протягом певного часового інтервалу. Оскільки середня тривалість між відмовами є величиною, оберненою до інтенсивності відмов при експоненційному законі розподілу випадкової величини, моделі на основі інтервалів часу між відмовами [26] також належать до цієї категорії. Найбільш поширеними моделями цього типу є деевтрофікаційна модель Jelinski та Moranda [46], модель Schick та Wolverton [47], геометрична деевтрофікаційна модель Jelinski та Moranda [48], геометрична пуассонова модель Moranda [49], модель недосконалого відлагодження Goel та Okumoto [50] та ін.

Моделі на основі апроксимації залежностей.

Група моделей на основі апроксимації залежностей використовує регресійний аналіз для дослідження взаємозв'язку між складністю ПЗ та кількістю помилок в програмі, кількістю змін, інтенсивністю відмов чи інтервалом часу між відмовами. Для виконання такої задачі використовують як параметричні, так і непараметричні методи. Моделі цього типу використовують лінійну та нелінійну регресію або аналіз часових рядів для встановлення функціонального взаємозв'язку між залежними та незалежними змінними, в якості яких можуть виступати кількість помилок та інтервали часу між відмовами, кількість змінених на етапі підтримки модулів, навички програміста, розмір програми і т. ін. відповідно. Цей тип включає наступні моделі [8, 9]: оцінювання помилок, оцінювання змін, оцінювання інтервалів часу між відмовами, оцінювання інтенсивності відмов.

Моделі зростання надійності.

Група моделей зростання надійності має на меті вимірювання та прогнозування покращення показників надійності ПЗ під час процесу відлагодження програмного засобу. Для опису такого покращення використовують функцію зростання. Аргументом такої функції може бути час, кількість тестових випадків або ітерацій тестування, залежною змінною може бути імовірність безвідмовної роботи, інтенсивність відмов чи загальна кількість виявлених помилок [8, 25, 51]. До моделей цього типу належать: модель зростання Duane [52], модель зростання Weibull [53], вейбулівська модель Wagoner [54], модель логістичної функції зростання [55] та ін.

Моделі на основі архітектурного підходу.

Група моделей на основі структури програми розглядає програму як мережу чи граф надійності. Вузлами такої мережі є модулі або підпрограми, а напрямлені дуги відображають послідовність виконання програми по модулям. Отримавши оцінку надійності кожного вузла, надійність переходів між вузлами, матрицю ймовірностей переходів в мережі, та припускаючи незалежність відмов кожного вузла, можна отримати надійність програми в цілому як розв'язок

проблеми надійності мережі. Моделями, які входять до цієї групи є модель Littlewood з марковською структурою [49], користувацька марковська модель Cheung [56] та ін. [15, 33–37, 57].

Моделі на основі області вхідних даних.

Ця група моделей використовує запуск програми (опрацювання вхідного стану) в якості аргументу функції надійності на противагу часу в моделях на основі часової області. Надійність оцінюється як відношення успішних запусків програми до загальної кількості запусків. Основна увага приділяється розподілу імовірності вхідних станів або операційному профілю програми. Через складність отримання такого розподілу, область вхідних даних розділяють на набір класів еквівалентності, кожен з яких зазвичай пов'язують з одним зі шляхів виконання програми. Оцінку надійності ПЗ отримують на основі помилок, виявлених шляхом фізичного чи символьного виконання тестових прикладів взятих з області вхідних даних. До моделей цього класу відносяться моделі Nelson [58] та Ramamoorthy–Bastani [59], в якій автори розглядали надійність критичного ПЗ, ПЗ реального часу та ПЗ автоматизованого управління.

Моделі на основі шляху виконання програми.

Моделі цього типу оцінюють надійність програмного забезпечення на основі імовірності виконання логічного шляху програми та імовірності виконання некоректного шляху. Такі моделі подібні до моделей на основі області вхідних даних, оскільки кожен вхідний стан відповідає шляху виконання програми. Моделлю, яка належить до цієї групи [8] є декомпозиційна модель Shooman [60].

Моделі на основі неоднорідного пуассонового процесу.

Група моделей на основі неоднорідного пуассонового процесу (НПП) надає аналітичні засоби для опису поведінки відмов ПЗ під час тестування. Головною проблемою моделей цього типу є визначення вигляду функції математичного сподівання кумулятивної кількості відмов, що спостерігались до деякого моменту часу. До цієї групи належить значна кількість моделей, зокрема експоненційна модель Musa [61], модель неоднорідного пуассонового процесу

Goel та Okumoto [62], модель S-подібного зростання [63], гіперекспонеційного зростання [64], узагальнена модель негомогенного пуасонівського процесу Тимошенко–Дідковської [65], узагальнена модель пуассонового процесу з показником складності [66] тощо.

Марковські моделі надійності ПЗ.

Група марковських моделей є найбільш загальним засобом для представлення процесу відмов ПЗ. Кількість залишкових помилок моделюється як стохастичний процес підрахунку. У випадку використання марковського ланцюга з дискретними станами і неперервним часом, стан процесу відображає кількість залишкових помилок, а тривалість між відмовами відповідає часу переходу з одного стану в інший. Якщо припустити, що інтенсивність відмов програми пропорційна кількості залишкових помилок, то можна використати дві моделі – процесу лінійної загибелі та лінійного породження та загибелі. Перша припускає, що кількість залишкових помилок є монотонно не зростаючою функцією часу, тоді як остання дозволяє враховувати помилки, внесені під час відлагодження програми. При розгляді нестаціонарної марковської моделі, модель стає дуже широкою і узагальнює багато існуючих моделей. Властивість не стаціонарності інтенсивності відмов також може моделювати припущення про неоднакову інтенсивність відмов від кожної помилки.

В цих моделях припускають, що відмови модулів незалежні одна від одної. Це припущення виглядає адекватним на модульному рівні, оскільки модулі можуть проектуватись, реалізовуватись та тестуватись незалежно. Однак це припущення може не справджуватись на рівні системи в цілому [25].

До моделей цієї групи належать [8, 9, 25, 37, 67, 68] модель лінійної загибелі з ідеальним відлагодженням, лінійної загибелі з недосконалим відлагодженням, нестаціонарної лінійної загибелі з ідеальним відлагодженням, нестаціонарної лінійної загибелі і породження.

Баєсівські та уніфіковані моделі надійності ПЗ.

Група баєсівських та уніфікованих моделей припускає апріорне встановлення розподілу інтенсивності відмов. Ці моделі використовуються тоді,

коли інженер з надійності ПЗ має значний досвід і глибоке розуміння причин відмов, а відмови є нечастою подією [8]. Крім ланцюгів Маркова з дискретними станами і неперервним часом, двома іншими узагальненими моделями є статистики експоненційного порядку [69, 70] та шокові моделі [71–73].

Разом з тим, слід зазначити, що розвиток теорії надійності стосовно програмних засобів, мережевих технологій та web-систем йде недостатньо ефективно. Це, на думку авторів [74] спричинено кількома причинами, такими як те, що класична теорія надійності не може описати і дати адекватні оцінки об'єктам, працездатність яких порушується не тільки внаслідок відмов фізичної природи, а й через помилки проектування, інформаційні впливи та ін., при розробці таких систем зберігається орієнтація на якісні, а не кількісні інструментарії оцінки тощо. Така ситуація в теорії надійності комп'ютерних систем склалась в останні десятиліття незважаючи на те, що все більшу частку в причинах відмов комп'ютерних систем складають помилки програмних засобів, перевантаження мереж та несанкціоновані інформаційні впливи [74]. Відображенням пошуку шляхів подолання цієї кризи стала поява і наповнення новим змістом терміну "гарантоздатність" (англ. dependability). Методології побудови таких систем, присвячені, наприклад, роботи [74, 75].

1.4. Моделі надійності ПЗ на основі неоднорідного пуассонового процесу

З початку 1970-х, коли почались інтенсивні дослідження в галузі надійності програмного забезпечення, було запропоновано багато моделей надійності ПЗ. В основу більшості таких моделей покладено розподіл Пуассона, параметри якого мають різний вигляд для різних моделей, оскільки використання такого розподілу випадкових величин добре зарекомендувало себе в багатьох областях, де предметом дослідження є в кількість випадкових незалежних подій [26].

До таких моделей відносяться моделі: Brooks [76], Shooman [77], La Padula [78, 79], Jelinski–Moranda [46, 80], Schick–Wolverton [81], Musa [61, 73, 82, 83],

Schneidewind [84], Goel–Okumoto [62], S-подібна модель зростання надійності Yamada [63] та ін.

В загальному випадку моделі надійності ПЗ на основі неоднорідного пуассонового процесу (НПП) можна поділити на чотири великих класи [23]:

- експоненційні моделі на основі НПП, які описують експоненційне зростання надійності ПЗ;
- S-подібні моделі на основі НПП, в яких крива зростання надійності ПЗ має S-подібну форму, що означає, що ця крива перетинає експоненційну криву знизу, і такий перетин є єдиним;
- моделі на основі НПП з урахуванням недосконалого відлагодження (англ. Imperfect Degugging Models), які враховують, що при видаленні помилки з програмного коду можуть бути внесені нові помилки;
- S-подібні моделі на основі НПП з урахуванням недосконалого відлагодження.

Основним припущенням моделей на основі неоднорідного пуассонового процесу є те, що $M(t)$, кількість відмов ПЗ на інтервалі часу $(0, t]$, відповідає розподілу Пуассона, а $\{M(t), t \geq 0\}$ відповідає НПП. Нехай $\mu(t) = E[M(t)]$ – математичне сподівання $M(t)$ або середня функція пуассонового процесу. Різні моделі цього класу відрізняються виглядом функції $\mu(t)$. Кількість відмов ПЗ, що відбулись до часу t визначається розподілом:

$$Pr\{M(t) = k\} = \frac{\big(\mu(t)\big)^k}{k!} e^{-\mu(t)}, k \geq 0, \tag{1.6}$$

За визначенням, математичне сподівання кумулятивної кількості відмов $\mu(t)$ може бути виражене через параметр потоку відмов ПЗ як [23, 85]:

$$\mu(t) = \int_0^t \lambda(s) ds,$$

$$\lambda(t) = \frac{d\mu(t)}{dt}. \tag{1.7}$$

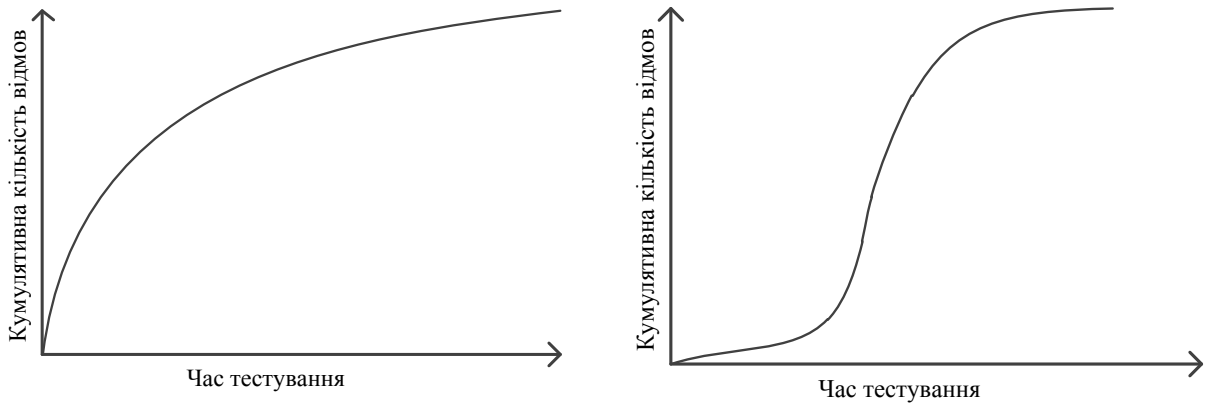
Імовірність безвідмовної роботи ПЗ (в англомовних джерелах – функція надійності, reliability function) запишеться як [23]:

$$R(t) = e^{-\mu(t)} = e^{-\int_0^t \lambda(s)ds}. \quad (1.8)$$

Моделі на основі неоднорідного пуассонового процесу дуже подібні до моделей на основі однорідного пуассонового процесу, різниця між ними полягає у тому, що в перших математичне сподівання кількості відмов є функцією часу.

Широке розповсюдження моделей надійності ПЗ на основі НПП, серед іншого, пов'язано з: можливістю обчислення очікуваної кількості відмов на будь-який момент часу завдяки явному вигляду функції математичного сподівання; легкістю отримання точкових оцінок параметрів моделі з використанням методу максимальної правдоподібності (ММП) чи найменших квадратів. Іншою важливою перевагою моделей цього типу є їх властивість суперпозиції та трансформації часу. Декілька моделей на основі неоднорідного пуассонового процесу можуть бути одночасно задіяні шляхом додаванням відповідних функцій математичного сподівання. Параметр потоку відмов такого процесу також буде сумою параметрів потоків відмов задіяних процесів.

Як уже зазначалось, деякі моделі на основі НПП описують експоненційне зростання надійності, тоді як інші демонструють S-подібне зростання, залежно від природи явища зростання надійності протягом тестування [14, 25]. Відповідно до цієї характеристики моделі відносять до категорії експоненційних чи S-подібних відповідно. Появу S-подібної кривої відносять за рахунок різних факторів і в літературі існує достатня кількість моделей на основі такої форми. Якісно залежність кумулятивної кількості відмов від часу для моделей обох категорій зображена на рис. 1.2.



(а) *(б)*

Рис. 1.2. Якісне зображення залежностей кумулятивної кількості відмов для експоненційних (а) та S-подібних (б) моделей надійності.

У одній з перших робіт Goel та Okumoto [62] запропонували модель із залежною від часу інтенсивністю відмов на основі неоднорідного пуассонового процесу (яка до тепер залишається однією з найбільш популярних моделей надійності ПЗ) у наступній формі:

$$\mu(t) = a(1 - e^{-bt}), \tag{1.9}$$

$$\lambda(t) \equiv \mu'(t) = abe^{-bt}, \tag{1.10}$$

де $\mu(t)$ – математичне сподівання пуассонового процесу (1.6), яке має зміст очікуваної кількості відмов до моменту часу t, $\lambda(t)$ – параметр потоку відмов; параметр a має зміст очікуваної загальної кількості відмов ПЗ, а параметр b – швидкість виявлення помилок, або інтенсивність відмов однієї помилки (англ. fault detection rate per fault [26], failure intensity of a fault [23]).

Пізніше Goel модифікував цю модель [26] увівши параметр якості тестування c для відображення поведінки реального ПЗ, в якому параметр потоку відмов на початкових етапах зростає, а потім спадає:

$$\mu(t) = a\left(1 - e^{-bt^c}\right), \tag{1.11}$$

$$\lambda(t) \equiv \mu'(t) = abct^{c-1}e^{-bt^c}. \tag{1.12}$$

Зауважимо, що оригінальна модель неоднорідного пуассонового процесу Goel–Okumoto була побудована для опису поведінки надійності ПЗ у випадку

неперервного часу. Однак, вона може бути успішно застосована для дискретних часових інтервалів безпосередньо шляхом обмеження значень часу (напр. $t_1, t_2, \dots, t_k$) до додатних цілих чисел. Таким чином часовий аргумент може виражати кількість тестових випадків, які використовувались в процесі тестування ПЗ.

У логарифмічній моделі пуассонового часу виконання Musa та Okumoto [73] математичне сподівання кумулятивної кількості відмов до часу t описується наступним виразом:

$$\mu(t) = \frac{1}{\theta} ln(\lambda_0 \theta t + 1), \tag{1.13}$$

де λ_0 та θ мають зміст початкового параметру потоку відмов та зниження швидкості нормалізованої інтенсивності відмов на одну помилку, відповідно. Ця модель є дуже подібною до геометричної деєутрофікаційної моделі Moranda [80] і може розглядатись як неперервна версія такої моделі.

Експоненційні моделі можуть бути для спрощення процесу моделювання зведені до єдиного набору рівнянь [14, 86], які представляють сукупність важливих моделей з експоненційною функцією інтенсивності відмов. Загальним припущенням залишається те, що інтенсивність відмов пропорційна кількості залишкових помилок, залишається постійною в період між відмовами та зменшується на постійну величину при вилученні помилки із системи.

У інших випадках, коли була потреба у описі зростання надійності S-подібною кривою, використовували деякі існуючі моделі надійності апаратного забезпечення з подібним виглядом кривої [87]. Літературні дані пов'язують S-подібне зростання з різними причинами: взаємозалежністю дефектів ПЗ [63, 70], меншою ефективністю тестування на початковому етапі, порівняно з наступним [25] та ін. Так, S-подібна модель зростання надійності [63, 70] має наступний вигляд математичного сподівання пуассонового процесу (1.6) та параметра потоку відмов:

$$\mu(t) = \alpha\left(1 - (1 + \beta t)e^{-\beta t}\right), \tag{1.14}$$

$$\lambda(t) = \alpha\beta^2 t\, e^{-\beta t}. \quad (1.15)$$

На даний момент розроблено декілька моделей з S-подібною формою кривої зростання надійності ПЗ, які враховують різні причини такої форми кривої [9, 23, 25, 71, 88, 89]. Крім того недавно була представлена "узагальнена модель негомогенного пуасонівського процесу" [65]. Для підвищення точності цієї моделі автори пропонують використовувати наступні вирази для математичного сподівання кумулятивної кількості відмов та, відповідно, параметру потоку відмов, у яких введено додатковий параметр n для оцінки величини проекту:

$$\mu(t) = \alpha\left(n! - \sum_{i=0}^{n} \frac{n!\,\beta^{n-i}}{(n-i)!} t^{n-i} e^{-\beta t}\right), \quad (1.16)$$

$$\lambda(t) = \alpha\beta^{n+1} t^n e^{-\beta t}, \quad (1.17)$$

а вибір параметру n залежить від процесу проведення тестування з наступними рекомендованими значеннями [65]:

- $n = 0$ – для невеликого проекту, в якому розробник є одночасно і тестером (моделі Musa, Goel–Okumoto та їм подібні);
- $n = 1$ – для середнього проекту, в якому тестування і проектування ПЗ виробляється різними людьми з однієї робочої групи (S-подібна модель);
- $n = 2$ – для великого проекту, в якому групи тестування і розробки ПЗ працюють над проектом паралельно;
- $n = 3$ – для дуже великого проекту, в якому відділи тестування і розробки незалежні.

Підсумовуючи, можна зазначити, що усі вищеописані моделі базуються на таких основних припущеннях [45, 46, 58, 59, 61–63, 73, 76–84, 87, 90–95]:

- Відмови ПЗ відбуваються під час його експлуатації та спричинені помилками, що залишаються в системі.
- Кількість помилок на момент t, $M(t)$ відповідає пуассоновому процесу з функцією математичного сподівання $\mu(t)$. При цьому

очікувана кількість помилок, які буде виявлено за проміжок часу $t + \Delta t$ пропорційна кількості помилок, що залишалися у системі на момент t. Також вважається, що $\mu(t)$ – обмежена, не спадаюча функція часу: $\lim_{t\to\infty} \mu(t) = N < \infty$. Отже, ці моделі належать до категорії скінчених функцій.

- Кількості помилок $(f_1, f_2, \ldots, f_n)$, виявлених на відповідних часових інтервалах $[(t_0 = 0; t_1), (t_1; t_2), \ldots, (t_{n-1}; t_n)]$ незалежні для будь-якої скінченої кількості часових проміжків $t_1 < t_2 < \cdots < t_n$.
- Імовірності виявлення будь-якої помилки на будь-якій ітерації є однаковими.
- При виявленні відмови, з програмного коду гарантовано видаляється тільки одна помилка без уведення нових (справедливо тільки для моделей, що не враховують недосконале відлагодження).

Аналітичний вигляд функцій параметру потоку відмов $\lambda(t)$ та функції кумулятивної кількості відмов $\mu(t)$ основних моделей надійності ПЗ на основі НПП наведено в табл. 1.2.

Таблиця 1.2

Функції інтенсивності відмов основних моделей надійності ПЗ.

Модель	**Параметр потоку відмов**	**Кумулятивна кількість відмов**
Goel–Okumoto	$\lambda(t) = Nb\exp(-bt)$	$\mu(t) = N(1 - \exp(-bt))$
Schneidewind	$\lambda(t) = \alpha_0 \exp(-\beta t)$	$\mu(t) = \frac{\alpha_0}{\beta}(1 - \exp(-\beta t))$
Musa	$\lambda(t) = \beta_0\beta_1 \exp(-\beta_1 t)$	$\mu(t) = \beta_0(1 - \exp(-\beta_1 t))$
S-подібна	$\lambda(t) = \alpha\beta^2 t \exp(-\beta t)$	$\mu(t) = \alpha(1 - (1 + \beta t)\exp(-\beta t))$
Тимошенко–Дідковської	$\lambda(t) = \alpha\beta^{n+1}t^n \exp(-\beta t)$	$\mu(t) = \alpha\left(n! - \sum_{i=0}^{n} \frac{n!\,\beta^{n-i}}{(n-i)!} t^{n-i}\exp(-\beta t)\right)$

Неважко помітити, що параметри моделей, наведених в табл. 1.2, пов'язані такими співвідношеннями: $N = \alpha;\ \beta_0 = \alpha;\ b = \beta;\ \beta_1 = \beta;\ \alpha_0 = \alpha\beta$, де α – загальна кількість помилок, які були виявлені в системі від початку спостереження, β – швидкість зміни функції параметру потоку відмов.

Таким чином, можна зробити висновок, що основним недоліком відомих моделей на основі НПП є те, що внаслідок припущень і спрощень, вони не достатньо адекватно відображають процес тестування, а результати, отримані при їх застосуванні не завжди збігаються з отриманими на практиці [65, 90]. Крім того, моделі [46, 61–63, 73, 77–84, 90] не враховують величину та складність програмного продукту. До недоліків моделі [65] можна віднести відсутність формалізації параметру складності проекту n, його апріорне встановлення, незалежно від реальних експериментальних даних тестування ПЗ; значення параметру n з діапазону цілих чисел, що не дасть змоги описувати поведінку реального ПЗ, яке за своєю величиною і складністю має скоріше неперервний ніж дискретний характер. Крім того, оскільки значення n не є параметром розподілу, модель [65] фактично не є однією взаємопов'язаною моделлю, а набором з чотирьох моделей, що відрізняються значенням n. Отже, сучасні тенденції аналізу надійності ПЗ [96] полягають, з одного боку, у підвищенні ступеня адекватності класичних моделей надійності [51, 97], які розглядають ПЗ як чорну скриньку, зокрема моделей на основі НПП [14, 98], що надають аналітичні вирази для опису поведінки відмов ПЗ та легко впроваджуються в процеси індустрії ПЗ, незважаючи на дискусії з цього питання [42, 90, 99].

1.5. Моделі надійності ПЗ на основі компонентного підходу

У зв'язку зі зростаючою складністю програмних продуктів моделі на основі НПП, які і переважна більшість моделей, описаних в підрозділі 1.3, перестали описувати поведінку надійності ПЗ із достатнім для практичних цілей ступенем адекватності. Тому, починаючи з 90-х років минулого століття все більшого поширення набувають моделі типу "білої скриньки", які описують надійність програмного продукту беручи до уваги його внутрішню будову. У моделях

такого типу надійність системи є функцією надійності її модулів (логічна частина ПЗ, яку можна тестувати та реалізовувати незалежно від інших, наприклад, файл). Особливої популярності такі моделі набули в останнє десятиліття, адже створення переважної більшості сучасних складних багатокомпонентних програмних продуктів відбувається на основі об’єктно-орієнтованої парадигми, тобто коли програма розглядається як сукупність паралельно існуючих сутностей (об'єктів), які взаємодіють між собою [100]. Кожен об'єкт виконує певні операції та характеризується деякою поведінкою. Найвідомішими представниками цієї парадигми є C++, Objective C, C#, Java, тому все більша увага приділяється прогнозуванню надійності ПЗ, створеного з використанням об’єктно-орієнтованого програмування (ООП). На даний момент розроблено декілька підходів для вирішення даної проблеми та запропоновано моделі оцінювання надійності багатокомпонентної програмної системи [15, 33, 35, 49, 56, 60, 69, 101–111].

У свою чергу компонентні моделі, або моделі типу "білої скриньки" поділяються на адитивні моделі, моделі розроблені на основі шляхів виконання ПЗ та на основі архітектурного підходу (рис. 1.3) [15, 35].

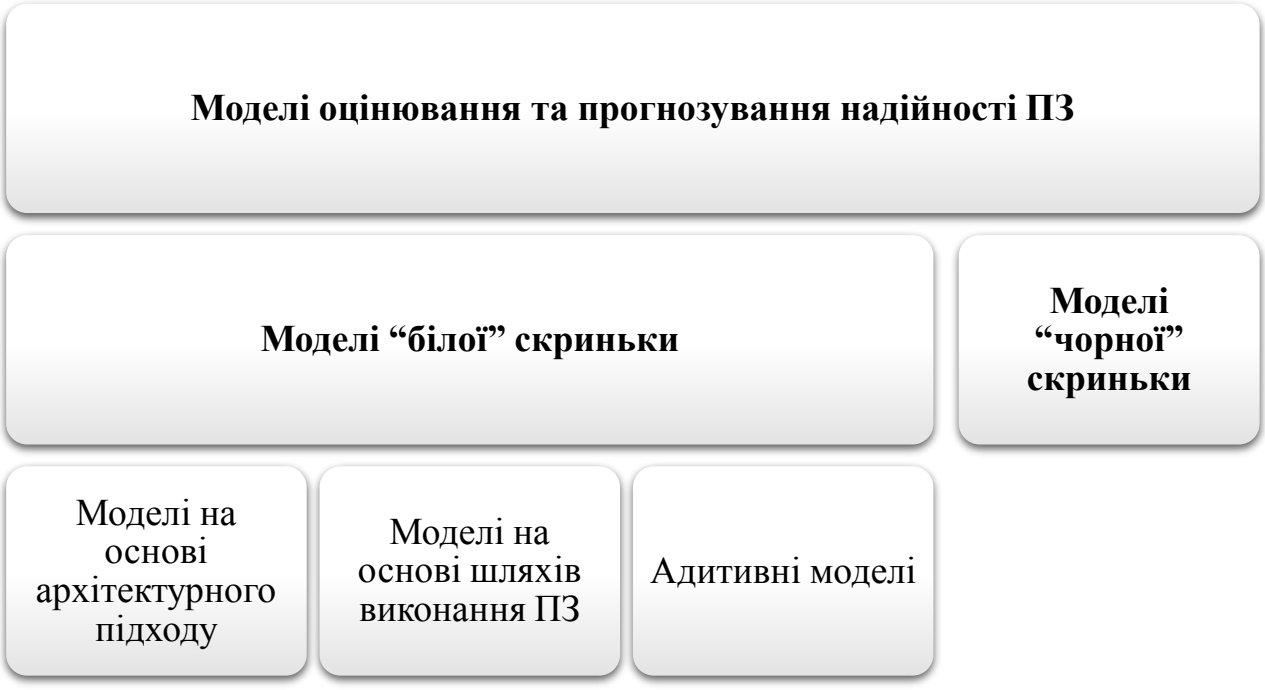


Рис. 1.3. Класифікація моделей оцінювання та прогнозування надійності ПЗ.

1.5.1. Адитивні моделі.

У цьому класі моделей не враховується цілком архітектура ПЗ, а для визначення надійності системи використовується лише надійність кожного модуля програмного продукту. Під надійністю модуля $R(t)$ в цьому розділі розуміється ймовірність безвідмовної роботи, згідно [9]: $R(t) = Pr\{T > t\}$, де T є випадковою величиною, яка означає тривалість між відмовами або тривалість до відмови.

Основні припущення, які допускаються в адитивних моделях:

- надійність всієї системи є сумою надійностей її модулів;
- надійність кожного модуля описується негомогенним пуассоновим процесом.

Найвідомішою моделлю даного класу є *модель Xie та Wohlin* [101]. У відповідності до цієї моделі система складається з n модулів, для кожної з яких відома інтенсивність відмов $\lambda_i(t)$, $i = \overline{1,n}$. Тоді інтенсивність відмов системи $\lambda_s(t)$ обчислюється за формулою

$$\lambda_s(t) = \sum_{i=1}^{n} \lambda_i(t). \tag{1.18}$$

Недоліком даної моделі є неврахування різного впливу кожного модуля на надійність всієї системи та неодночасного початку виконання кожного модуля.

1.5.2. Моделі, побудовані на основі шляхів виконання ПЗ.

Для прогнозування надійності програмного продукту даний підхід використовує інформацію про шляхи виконання (послідовність виконання модулів, потік управління) програми, отриману переважно емпіричним шляхом (тестуванням).

Найбільш поширеними моделями цього класу є моделі Shooman [60] і Krishnamurthy–Mathur [100].

Модель Shooman є однією з перших моделей цього класу. Основні припущення, які допускаються в моделі:

- вважається, що модель базується на основі інформації про шляхи виконання програми та про їх частоту виконання f_i, $i = \overline{1,m}$;
- відома ймовірність відмов q_i кожного шляху i, $i = \overline{1,m}$.

Ймовірність відмови програмної системи в цій моделі описується виразом

$$q_0 = \lim_{N\to\infty} \frac{n_f}{N} = \sum_{i=1}^{m} f_i q_i. \quad (1.19)$$

де n_f – загальна кількість помилок, N – кількість виконаних тестів.

Krishnamurthy ma Mathur [102] уперше запропонували визначати надійність системи як середнє значення надійності кожного зі шляхів її виконання.

Основними припущеннями, які лежать в основі цієї моделі, є:

- система складається з m модулів;
- послідовність виконання модулів для кожного шляху спостерігається за рахунок трасування модулів при виконанні тесту;
- для кожного модуля задана його надійність R_m.

Вважається, що програма P тестується множиною тестів TS. Надійність шляху пройденого при виконанні тесту $tc \in TS$ визначається співвідношенням

$$R_{tc} = \prod_{\forall m \in M(P,tc)} R_m, \quad (1.20)$$

де $M(P, tc)$ – множина послідовності модулів, що виконуються в програмі P під час проходження тесту tc. Тоді надійність системи в цілому виражається наступним чином

$$R = \frac{\sum_{\forall tc \in TS} R_{tc}}{card(TS)}. \quad (1.21)$$

Недоліком даного класу моделей є обмеженість кількості шляхів, отриманих експериментально, та послідовності виконання модулів кожного шляху.

1.5.3. Моделі на основі архітектурного підходу.

У моделях даного типу використовують граф потоку керування для опису архітектури ПЗ. Вважається, що передача контролю між модулями характеризується властивістю Маркова, тобто майбутня поведінка системи не залежить від її функціонування в минулому, а залежить лише від теперішнього стану (перебування в певному стані означає виконання відповідного модуля). У даному підході архітектуру ПЗ можна змоделювати як ланцюг Маркова з дискретним часом, неперервним часом та напівмарковським процесом. Пізніше кожна з моделей класифікується як поглинальна (містить поглинальний стан – стан, з якого система вийти не може) та непоглинальна (не містить поглинальних станів).

Крім цього дані моделі можуть ще поділятись на композиційні та ієрархічні [103]. До композиційних відносяться моделі, які одночасно комбінують архітектуру програмного продукту та характер його відмов для визначення надійності ПЗ. В ієрархічних моделях спочатку обчислюються параметри архітектурної моделі, а потім враховується поведінка відмов системи при прогнозуванні надійності ПЗ.

Моделі, побудовані на основі ланцюга Маркова з дискретним часом. У моделях, що відносяться до даного класу, взаємодію модулів можна описати у вигляді ланцюга Маркова з дискретним часом. Тобто передача контролю між модулями відбувається у визначені проміжки часу, отже, не залежить від кроку. У моделях цього класу вважають відомими:

- вектор ймовірностей перебування системи в різних станах в початковий момент часу;
- матриця ймовірностей переходів між модулями (станами марковського процесу);
- інтенсивність відмов кожного модуля.

Однією з найвідоміших моделей цього класу є *модель Gokhale* [104], у якій вперше для опису архітектури ПЗ та визначення надійності кожного модуля

використовується набір регресійних тестів. За допомогою аналізу покриття коду тестами, використовуючи автоматизований інструмент аналізу ATAC [105], визначається ймовірність переходів між модулями та тривалість перебування в кожному модулі (сума очікуваного часу виконання кожного блоку, з яких складається програмний модуль). Дана модель належить до ієрархічних, тобто спочатку обчислюються параметри моделі функціонування ПЗ, пізніше враховується поведінка відмов модулів для оцінювання надійності.

Основними припущеннями моделі Gokhale є [104]:

- зв'язок між модулями описується дискретним ланцюгом Маркова з одним поглинальним станом;
- тривалість виконання поглинального стану дорівнює нулеві;
- характер помилок кожного модуля описується негомогенним пуассоновим процесом, в якому інтенсивність відмов кожного модуля $\lambda_i(t)$ визначається шляхом тестування з використанням інструменту ATAC;
- відомі ймовірності передачі управління між i-тим та j-тим модулями програми – p_{ij};
- відома ймовірність q_i виклику першим модуля i, $i = \overline{1,N}$.

Надійність кожного модуля обчислюється як:

$$R_i(t) = \exp\left(-\int_0^{V_i t_i} \lambda_i(\tau) d\tau\right), \tag{1.22}$$

де $V_i t_i$ – це очікувана тривалість виконання модуля i. Автор [104] вважає критичними ті модулі, що мають максимальне значення $V_i t_i$.

Для знаходження V_i – очікуваної кількості відвідувань модуля i $\left(i = \overline{1,N}\right)$ потрібно розв'язати систему лінійних алгебраїчних рівнянь

$$V_i = q_i + \sum_{j=1}^{N-1} V_j p_{ji}. \tag{1.23}$$

Надійність системи в цілому обчислюється як:

$$R = \prod_{i=1}^{N} R_i. \tag{1.24}$$

Модель Kubat [106] розглядає випадок, коли ПЗ складається з N модулів, що можуть виконувати k завдань і навпаки, кожне завдання може виконуватись декількома модулями.

Базові припущення, які допускаються в цій моделі:

- зв'язок між модулями описується ланцюгом Маркова з дискретним часом;
- відома ймовірність $q_i(k)$ того, що завдання k буде викликати першим модуль i $\left(i = \overline{1,N}\right)$;
- задано ймовірність $p_{ij}(k)$ переходу від модуля i до модуля j при виконанні завдання k;
- тривалість виконання модулю i під час розв'язання завдання k характеризується густиною ймовірності $g_i(k,t)$, $\left(i = \overline{1,N}\right)$;
- відома інтенсивність відмов кожного модуля λ_i $\left(i = \overline{1,N}\right)$.

Надійність модуля i під час виконання завдання k обчислюється за формулою

$$R_i(k) = \int_0^{\infty} e^{-\lambda_i t} g_i(k,t)dt. \tag{1.25}$$

Очікувана кількість виконання модулю i під час розв'язання завдання k, позначається $V_i(k)$ і є розв'язком системи рівнянь

$$V_i(k) = q_i(k) + \sum_{j=1}^{N-1} V_j(k)p_{ji}(k)\,, \left(i = \overline{1,N}\right). \tag{1.26}$$

Отже, ймовірність того, що не відбудеться відмови під час виконання завдання k апроксимується виразом

$$R(k) \approx \prod_{i=1}^{N} [R_i(k)]^{V_i(k)}. \tag{1.27}$$

Інтенсивність відмов всієї системи обчислюється згідно

$$\lambda_s = \sum_{k=1}^{K} r_k[1 - R(k)], \tag{1.28}$$

де r_k – відсоток виконання завдання k.

Модель Cheung [56] є однією з найвідоміших і достатньо вивчених моделей. У даній моделі обчислюється надійність ПЗ з урахуванням надійності кожного модуля.

Основні припущення, на яких ґрунтується дана модель:

- зв'язок між модулями зображується ланцюгом Маркова з дискретним часом з одним вхідним та одним вихідним станами;
- відмови модулів є незалежними;
- відома надійність кожного модуля R_i.

Два поглинальні стани C та F додаються до даної моделі, що відповідно означають успішне виконання ПЗ та його відмову. Матриця ймовірностей P модифікується у матрицю $\hat{P}$, а ймовірності передавання управління p_{ij} – в $R_i p_{ij}$, що виражають ймовірність переходу від модуля i $\left(i = \overline{1, N}\right)$ до модуля j $\left(j = \overline{1, N}\right)$ без помилок. Надійність програми – це ймовірність перебування системи в стані C. Нехай Q – матриця, отримана вилученням з матриці $\hat{P}$ рядків та стовпців, що відповідають поглинальним станам C та F. $Q^k(1, N)$ – ймовірність перебування у кінцевому стані N із врахуванням початкового стану 1 через k переходів $\left(k = \overline{0 \ldots \infty}\right)$. У [56] отримано наступний вираз

$$S = I + Q + Q^2 + Q^3 + \cdots = \sum_{k=0}^{\infty} Q^k = (I - Q)^{-1}, \tag{1.29}$$

за допомогою якої обчислюється надійність всієї системи:

$$R = S(1, N)R_N. \tag{1.30}$$

Дана модель є добре дослідженою. Для неї отримано коефіцієнти матриці переходів для послідовного та паралельного виконання модулів, а також для випадку, коли виконання одного модулю може бути замінене виконанням іншого

та якщо після виконання наступного модуля не можна повернутись до попереднього [107]. Крім цього, для моделі Cheung проведено аналіз її невизначеності (тобто аналіз перевірки точності результату).

Моделі, побудовані на основі ланцюга Маркова з неперервним часом. Моделі цього класу виражають взаємодію модулів у вигляді ланцюга Маркова з неперервним часом, тобто, тривалість перебування в модулі експоненційно розподілений. До цього класу відносяться моделі Laprie [108] та Littlewood [49].

Модель Laprie є частковим випадком моделі Littlewood. Базові припущення, які допускаються в цій моделі:

- зв'язок між модулями описується ланцюгом Маркова з неперервним часом;
- задано $1/\mu_i$ – параметр середнього часу виконання модуля i $\left(i = \overline{1,N}\right)$;
- відома матриця імовірностей переходів між модулями p_{ij} $\left(i,j = \overline{1,N}\right)$;
- для кожного модуля відома інтенсивність відмов λ_i $\left(i = \overline{1,N}\right)$.

Система моделюється $N+1$ станом, де $(N+1)$-й стан є поглинальним. Розглянемо матрицю $B = \left[b_{ij}\right]$, елементи якої визначаються таким співвідношенням

$$\begin{gathered} b_{ii} = -(\mu_i + \lambda_i); \\ b_{ij} = p_{ij}\mu_i;\ i \neq j;\ i,j = \overline{1,N} \end{gathered} \tag{1.31}$$

Матриця B розкладається на суму двох матриць:

- матриця B', що відповідає процесу виконання ПЗ, діагональні елементи якої дорівнюють $-\mu_i$, недіагональні $-p_{ij}\mu_i$ $\left(i,j = \overline{1,N}\right)$;
- діагональна матриця B'', що відповідає процесу появи відмов ПЗ, діагональні елементи якої дорівнюють $-\lambda_i$ $\left(i = \overline{1,N}\right)$.

Вважається, що інтенсивність відмов набагато менша, ніж інтенсивність роботи програми, тобто $\lambda_i \ll \mu_i$ $\left(i = \overline{1,N}\right)$.

Інтенсивність відмов усієї системи обчислюється за формулою

$$\lambda_s = \sum_{i=1}^{N} \pi_i \lambda_i, \tag{1.32}$$

де $\pi = [\pi_i]$ – вектор стаціонарного розподілу, що є розв'язком системи рівнянь $\pi B' = 0$.

Фізичний зміст π_i – пропорція безвідмовного часу виконання модуля i, а $\pi_i \lambda_i$ – інтенсивність відмов *i*-того модуля $\left(i = \overline{1, N}\right)$.

Laprie та Kanoun [109] пізніше зосередили свою увагу на моделюванні надійності системи, що охоплює апаратні засоби в сукупності з ПЗ.

Моделі, побудовані на основі напівмарківського ланцюга. Даний клас моделей описує взаємодію модулів напівмарківським процесом, тобто, тривалість перебування в певному модулі має функцію розподілу, що залежить від стану, в якому перебуває система, а також, і від стану, в який вона перейде.

Модель Littlewood [49] є однією з перших і водночас найзагальніших моделей. У своїх перших роботах автор моделював систему ланцюгом Маркова з неперервним часом [69], а пізніше – напівмарківським ланцюгом. Основні припущення, які допускаються в моделі:

- задано матрицю ймовірностей переходів між станами $P = [p_{ij}]$ $\left(i, j = \overline{1, N}\right)$;
- обчислено інтенсивність відмов кожного модуля λ_i $\left(i = \overline{1, N}\right)$ (поведінка відмов описується пуассоновим процесом);
- обчислено ймовірність ν_{ij} появи помилки при передаванні управління від модуля i до модуля j;
- відома функція розподілу часу перебування в певному модулі $F_{ij}(t)$ з середнім часом m_{ij} $\left(i, j = \overline{1, N}\right)$.

Інтенсивність відмов програмної системи в цій моделі обчислюється наступним чином:

$$\lambda_s = \sum_i \lambda_i a_i + \sum_{i,j} b_{ij} \nu_{ij}, \tag{1.33}$$

де $a_i = {\pi_i \sum_j p_{ij} m_{ij}} \Big/ {\sum_i \pi_i \sum_j p_{ij} m_{ij}}$ – частина часу, проведеного в модулі i, $b_{ij} = {\pi_i p_{ij}} \Big/ {\sum_i \pi_i \sum_j p_{ij} m_{ij}}$ – частота передавання управління між модулями i та j, а π_i – елементи вектора стаціонарного розподілу $\left(i = \overline{1, N}\right)$.

Одним із суттєвих недоліків відомих в літературі моделей класу "білої" скриньки є складність їх практичного використання в індустрії програмного забезпечення. Багато параметрів у цих моделях вважаються наперед відомими, але не описані способи отримання їх значень для реальних програмних продуктів, що для багатьох параметрів є нетривіальною задачею.

Разом з тим, моделі оцінювання надійності ПЗ на основі архітектурного підходу [34, 35, 49, 56, 103, 104, 107–109, 112–115] у більшості випадків використовують теорію класичних марковських процесів, припускаючи незалежність виконання модулів програмної системи, що є спрощеним описом реального процесу виконання ПЗ, де імовірність переходу в наступний стан (наприклад передачі потоку управління до іншого модуля програми) може залежати не тільки від поточного стану, а й від передісторії потрапляння в цей стан. У [116] для усунення такого спрощення запропоновано використовувати математичний апарат ланцюгів Маркова вищого порядку (ЛМВП), який дає змогу точніше аналізувати поведінку програмного продукту.

1.6. Моделі і методи визначення політики оптимального введення ПЗ в експлуатацію

Якість програмної системи зазвичай залежить від того, скільки часу займає тестування та які методології тестування застосовуються. З одного боку, чим більше часу люди витратять на тестування, тим більше помилок буде виправлено, і тим більш надійним буде програмне забезпечення; однак, вартість тестування ПЗ також зросте. З іншого боку, якщо на тестування витрачено дуже мало часу, вартість ПЗ може бути зменшена, але користувачі ризикують

отримати ненадійне ПЗ [117, 118]. Це також підвищить вартість програмного продукту під час експлуатації, адже значно дорожче виправити помилку на етапі експлуатації, ніж на етапі тестування. Отже, важливою проблемою інженерії програмного забезпечення є визначення оптимального моменту для припинення тестування і випуску (релізу) продукту.

У [9] наведено огляд і обговорення декількох узагальнених моделей затрат на розробку програмного продукту, що базуються на функціях надійності ПЗ (імовірності безвідмовної роботи), що задаються на основі НПП. Використання таких моделей дає можливість ефективно спланувати ресурси для забезпечення вчасного та ефективного уведення в експлуатацію програмного продукту; визначити чи програмний продукт є достатньо надійним для релізу; отримати менеджерам чи розробникам підтримку в процесі прийняття рішень про перехід від поточного етапу тестування до випуску продукту. Все це забезпечується за рахунок визначення оптимальних політик тестування та випуску програмних систем на основі моделей затрат. Окрім вартостей, що входять до традиційних моделей затрат, моделі затрат, що розглядаються у [9], містять у собі такі параметри як вартість тестування, вартість відлагодження під час тестування, вартість відлагодження під час гарантійного терміну, та вартість ризиків програмної відмови. Ці моделі можуть бути використані для реалістичної оцінки повної вартості ПЗ для таких галузей, як телекомунікації, системи масового обслуговування, та вбудованих систем реального часу.

В моделях затрат, що розглядаються в цьому підрозділі використовуються наступні позначення та основні припущення: $m(T)$ – математичне сподівання помилок, що виявлені до моменту часу T; a – загальна кількість програмних помилок, які врешті решт можуть бути виявлені; b – показник експоненти; x – задане напрацювання (тривалість роботи); $R(x|T)$ – імовірність безвідмовної роботи ПЗ протягом часу T для заданого напрацювання x; T – час релізу програми; C_1 – вартість тестування за одиницю часу; C_2 – вартість усунення помилки за одиницю часу на етапі тестування; $E(T)$ – очікувана вартість

програмної системи на момент часу T; $N(T)$ – кількість помилок, виявлених до моменту часу T; Y – час виправлення (усунення) помилки на етапі тестування; μ_y – математичне сподівання часу виправлення помилки на етапі тестування, $E(Y)$.

Загальні припущення моделей затрат [9]:

(1) Вартість тестування є пропорційною до часу тестування.

(2) Вартість усунення помилок на етапі тестування пропорційна до сумарного часу усунення помилок, виявлених до кінця етапу тестування.

(3) Вартість усунення кожної помилки на етапі тестування підлягає усіченому експоненційному розподілу.

(4) Вартість ризику пов'язана з надійністю на кожен момент релізу.

Нехай час усунення помилки є випадковою величиною Y. Врахувавши припущення (3), отримують формулу розподілу густини ймовірності Y [9]:

$$s(y) = \frac{\lambda e^{-\lambda y}}{\int_0^{T_0} \lambda e^{-\lambda z}\, dz}, \qquad 0 \le y \le T_0$$

де T_0 – максимальний час усунення помилки. Очікуваний час виправлення кожної помилки становить:

$$\mu_y = E(Y) = \int_0^{T_0} y s(y) dy = \int_0^{T_0} \frac{y\lambda e^{-\lambda y}}{\int_0^{T_0} \lambda e^{-\lambda z} dz} dy.$$

Після спрощень можна отримати:

$$\mu_y = \frac{1 - (\lambda T_0 + 1)e^{-\lambda T_0}}{\lambda(1 - e^{-\lambda T_0})}. \tag{1.34}$$

1.6.1. Модель вартості ПЗ з фактором ризику

Розглядається модель вартості, що включає рівень ризику та час виправлення помилок [9]. На основі цієї моделі визначено оптимальні політики релізу, що мінімізують очікувану загальну вартість ПЗ. Без втрати загальності, для цієї моделі вартості в якості функції надійності було використано модель

НПП Goel–Okumoto [62], для якої імовірність безвідмовної роботи ПЗ визначається за формулою:

$$R(x|T) = e^{-[m(T+x)-m(T)]} = e^{-a[e^{-bT}-e^{-b(T+x)}]} \quad (1.35)$$

Очікувана вартість програмної системи, $E(T)$, визначається як: (1) вартість виконання тестування; (2) вартість виправлення помилок під час тестування; і (3) вартість ризиків відмови програми.

Вартість виконання тестування задається як:

$$E_1(T) = C_1 T \quad (1.36)$$

Очікуваний загальний час виправлення усіх $N(T)$помилок становить:

$$E\left[\sum_{i=1}^{N(T)} Y_i\right] = E[N(T)]E[Y_i] = m(T)\mu_y,$$

де μ_y визначається згідно (1.34). Отже, очікувана вартість виправлення всіх помилок, виявлених протягом часу T, може бути записана як:

$$E_2(T) = C_2 E\left[\sum_{i=1}^{N(T)} Y_i\right] = C_2 m(T)\mu_y. \quad (1.37)$$

Вартість ризику програмного збою після релізу ПЗ становить:

$$E_3(T) = C_3[1 - R(x|T)], \quad (1.38)$$

де C_3 – вартість (збитки) програмного збою.

Таким чином, очікувана загальна вартість ПЗ може бути виражена як [198]:

$$E(T) = C_1(T) + C_2 m(T)\mu_y + C_3[1 - R(x|T)]. \quad (1.39)$$

У [119] було введено такі позначення

$$f(T) = \lambda(T)\left[C_3(1 - e^{-bx})R(x|T) - C_2\mu_y\right], \quad (1.40)$$

$$g(T) = C_3(1 - e^{-bx})R(x|T)[1 - ae^{-bT}(1 - e^{-bx})],$$

і доведено теорему, яка, за відомими значеннями C_1, C_2, C_3, x та μ_y, визначає оптимальне значення часу релізу програми, яке мінімізує очікувану вартість ПЗ.

1.6.2. Модель вартості з тестовим покриттям

Іншою моделлю, описаною у [9] є модель вартості ПЗ з урахуванням покриттям тестами. Окрім традиційних складових вартості, таких як вартість тестування та вартість усунення помилок, в цю модель включено вартість ризику відмови програми в частинах, непокритих тестуванням. Описуються оптимальні політики релізу, що мінімізують очікувану загальну вартість програмного продукту в залежності від вимог надійності.

Ця модель вартості розроблялась на основі наступних припущень:

(1)–(3) Ці припущення ідентичні пунктам (1)–(3) загальних припущень, описаних на початку підрозділу 1.6.

(4) Функція кумулятивної кількість відмов з урахуванням покриття тестами задана моделлю PZ-покриття [120].

(5) Існує вартість ризику, пов'язана з неповним покриттям тестами. Постачальник ПЗ повинен платити відшкодування кожному замовнику за потенційні відмови в непокритому тестами коді ПЗ.

(6) Надійність ПЗ на момент релізу повинна задовольняти вимоги замовників.

Дана модель визначає очікувану вартість програмної системи $E(T)$ як:

1) Вартість тестування $E_1(T)$
2) Вартість виправлення помилок під час тестування $E_2(T)$
3) Вартість ризику потенційних відмов в непокритому тестами коді $E_3(T)$.

Обидві функції $E_1(T)$і $E_2(T)$ задаються аналогічно виразам (1.36) та (1.37) відповідно. Нехай існує D замовників. Постачальник повинен заплатити кожному з них певну суму грошей, C_3 , за потенційний ризик відмов в непокритому тестами коді. Загальна вартість ризику, пов'язаного з тестовим покриттям, $E_3(T)$, може бути подана наступним чином:

$$E_3(T) = C_3 D\big(1 - c(T)\big). \tag{1.41}$$

Тому очікувана загальна вартість ПЗ, $E(T)$, матиме вигляд:

$$E(T) = C_1(T) + C_2 m(T)\mu_y + C_3 D[1 - c(T)], \tag{1.42}$$

де μ_y задане рівністю (1.34). $m(T)$ – кумулятивна кількість відмов, $c(T)$ – функція покриття тестами, які задаються згідно моделі PZ-покриття [120].

У [9] визначено оптимальний час релізу, що мінімізує очікувану вартість ПЗ з урахуванням заданого рівня надійності R_0. Оптимізаційна задача може бути сформульована наступним чином:

$$\begin{cases} E(T) \to min, \\ R(x|T) \geqslant R_0, \end{cases}$$

де $E(T)$ задана рівністю (1.42). У [120] доведено теорему, в якій визначено оптимальне значення T, яке мінімізує очікувану загальну вартість $E(T)$ з урахуванням вимоги до надійності ПЗ R_0.

У [9] ця теорема була застосована для визначення оптимального часу релізу програмного забезпечення на основі результатів тестування програмного продукту AT&T System T, який є мережевою системою управління, що отримує дані телеметрії, такі як сигнали тривоги, інформацію про продуктивність обладнання та діагностичні повідомлення, і пересилає їх оператору для подальших дій. Дані про відмови цієї системи були опубліковані в [121].

1.6.3. Узагальнена модель вартості ПЗ

У [9] описано узагальнену модель вартості, яка крім факторів вартості, описаних у підрозділі 1.6, враховує вартість виправлення помилок, виявлених під час гарантійного терміну, та ризик відмови програми.

Тут використовуються наступні позначення: C_0 – організаційні витрати на тестування ПЗ; C_3 – вартість виправлення помилки за одиницю часу на етапі експлуатації; C_4 – втрати від відмови програми; W – змінна часу на виправлення помилки під час гарантійного періоду на етапі експлуатації; μ_W – очікуваний час виправлення помилки під час гарантійного періоду на етапі експлуатації, який є математичним сподіванням $E(W)$; T_W – гарантійний період; α – рівень зменшення вартості тестування.

Припущеннями моделі є:

(1)–(4) Ці припущення ідентичні пунктам (1)–(4) загальних припущень (див. підрозділ 1.6).

(5) Існують організаційні витрати на початку процесу розробки ПЗ.

(6) Вартість тестування є степеневою функцією від часу тестування. Це означає, що на початку тестування вартість зростає з високою швидкістю, яка пізніше сповільнюється.

(7) Час усунення кожної помилки під час гарантійного періоду підлягає усіченому експоненційному розподілу.

(8) Вартість виправлення помилок під час гарантійного періоду пропорційна до загального часу виправлення всіх помилок, виявлених протягом інтервалу (T, T_W).

З припущення 7, функція густини усіченого експоненційного розподілу часу виправлення помилки під час гарантійного періоду:

$$q(w) = \frac{\lambda_w e^{-\lambda_w w}}{\int_0^{T_0} \lambda_w \, e^{-\lambda_w x dx}}, 0 \leqslant w \leqslant T_0. \tag{1.43}$$

Таким чином, очікуваний час усунення помилки під час гарантійного періоду:

$$\mu_w = \frac{1 - (\lambda_w T_0 + 1) e^{-\lambda_w T_0}}{\lambda_w (1 - e^{-\lambda_w T_0})}. \tag{1.44}$$

Очікувана вартість ПЗ складається з організаційних витрат, вартості тестування, вартості виправлення помилок під час тестування та гарантійного періоду, та величини ризику, пов'язаного з релізом ПЗ в момент часу T. Таким чином, очікувана загальна вартість ПЗ $E(T)$ може бути записана наступним чином [122]:

$$\begin{aligned} E(T) = C_0 + C_1 T^{\alpha} + C_2 m(T) \mu_y + C_3 \mu_w [m(T + T_w) - m(T)] \\ + C_4 [1 - R(x|T)], \end{aligned} \tag{1.45}$$

де $0 \leqslant \alpha \leqslant 1$.

У [122] доведено теорему про визначення оптимального часу релізу ПЗ T^*, який мінімізує очікувану загальну вартість ПЗ.

1.6.4. Модель вартості з декількома типами помилок

У [9] розглядається модель вартості ПЗ з декількома типами помилок, що приводять до відмов ПЗ, побудована згідно з наступними припущеннями:

(1) Вартість виявлення та виправлення помилки під час розробки є нижчою, ніж на етапі експлуатації.

(2) Вартість видалення певного типу помилок є незмінною на етапі відлагодження.

(3) Вартість видалення певного типу помилок є незмінною під час експлуатації.

(4) Вартість видалення критичних помилок є вищою, ніж значних помилок, а вартість видалення значних помилок є вищою, ніж незначних помилок.

(5) На етапі відлагодження додається певна незмінна вартість.

Для опису цієї моделі введено наступні позначення: T – час релізу програми; C_{i1} – вартість виправлення помилки i-того типу на етапі тестування; C_{i2} – вартість виправлення помилки i-того типу на етапі експлуатації; C_3 – вартість тестування за одиницю часу; $E(T)$ – очікувана вартість ПЗ; R_0 – заданий рівень надійності ПЗ; T_r – час відлагодження, необхідний для отримання мінімальної вартості ПЗ за умови обмеження на рівень надійності; T_e – час відлагодження, необхідний для отримання мінімальної вартості ПЗ за умови обмеження на кількість помилок, що залишаються; T_{rel} – час відлагодження, необхідний для досягнення максимального рівня надійності за умови обмеження на вартість ПЗ.

Нехай тривалість життєвого циклу ПЗ є випадковою величиною t, а $g(t)$ – функція густини ймовірності t. Очікувана вартість ПЗ включає вартість виправлення помилок у процесі життєвого циклу ПЗ, починаючи з етапу тестування. $E(T)$може бути записана наступним чином [123]:

$$E(T) = \int_0^T \left[C_3 t + \sum_{i=1}^{3} C_{i1} m_i(t)\right] g(t)dt + \int_T^{\infty} \left[C_3 t + \sum_{i=1}^{3} C_{i1} m_i(T) + \sum_{i=1}^{3} C_{i2}\big(m_i(t) - m_i(T)\big)\right] g(t)dt, \quad (1.46)$$

де $m_i(t)$ згідно [123] задається як:

$$m_i(t) = \frac{ap_i}{1-\beta_i}\left(1 - e^{-(1-\beta_i)b_i t}\right).$$

Функція $E(T)$ показує вартість тестування за одиницю часу і вартість виправлення помилок під час тестування, якщо життєвий цикл ПЗ не є більшим, ніж час релізу. З іншого боку, якщо життєвий цикл ПЗ більший, ніж час релізу, то необхідно ввести додатковий фактор вартості, наприклад, вартість виправлення помилок на етапі експлуатації. Для визначення оптимального часу релізу, який мінімізує $E(T)$, в [9] введено позначення

$$h(T) = \sum_{i=1}^{3} (C_{i2} - C_{i1})\lambda_i(T). \quad (1.47)$$

У [9] показано, що існує оптимальний час тестування, T^*, який мінімізує $E(T)$:

Якщо $h(0) \leqslant C_3$, то $T^* = 0$, інакше $T^* = h^{-1}(C_3)$.

Ця теорема показує, що якщо $h(0) \geqslant C_3$, то час тестування, який мінімізує вартість ПЗ, уже досягнутий. Таким чином, граничні витрати на подальше тестування — це зростаюча функція, і оскільки кожен наступний тест чи відлагодження збільшує вартість ПЗ, як наслідок потрібно припинити тестування і випустити ПЗ на ринок. Однак, якщо час тестування, який мінімізує вартість ПЗ, не був досягнутий, і граничні витрати на подальше відлагодження є спадаючою функцією, тестування повинно продовжуватись до часу T^*, такого що $h(T^*) = C_3$.

Хоча оптимальні політики релізу є теоретично обґрунтованими [9], на практиці мінімізація вартості ПЗ не завжди є єдиною ціллю. У [9] наведено

оптимальні політики релізу, які мінімізують очікувану вартість ПЗ, враховуючи різні накладені умови, такі як:

- оптимізація вартості за умови обмеження по надійності;
- оптимізація вартості за умови обмеження на кількість помилок, що залишились;
- оптимізація надійності ПЗ за умови обмеження вартості.

1.6.5. Модель збільшення прибутку з урахуванням випадкових сценаріїв використання

У [9] розглядається модель збільшення прибутку за умов випадкового середовища використання програмного продукту, яка враховує не тільки час, необхідний для виправлення помилок під час внутрішнього тестування, вартість усунення помилок під час бета-тестування, вартість ризику через програмні відмови, але й вигоди від надійного виконання програмного забезпечення як під час бета-тестування, так і на етапі експлуатації.

Узагальнена модель вартості ПЗ (див. підрозділ 1.6.3) враховує як вартість гарантії, так і вартість штрафу після випуску програмного забезпечення, які перекриваються одна-одною. Модель збільшення прибутку з урахуванням випадкових сценаріїв використання заснована на дослідженні [124], і у [9] має назву модель вартості T-P. Модель вартості T-P не враховує вартість гарантії, але натомість враховує подібну концепцію – вартість витрат, пов'язаних з бета-тестуванням, яке проводиться в польових умовах, тобто задіяні сценарії використання наближені до сценаріїв реальної експлуатації продукту. В цьому випадку вартість бета-тестування і вартість штрафу після випуску ПЗ не перекриваються.

На рис. 1.4 зображено процес розробки ПЗ, який враховується в цій моделі вартості: внутрішнє тестування, бета-тестування і експлуатацію, при цьому бета-тестування і експлуатація проводяться в польових умовах, в яких сценарії використання ПЗ зазвичай досить сильно відрізняються від сценаріїв, згідно яких проводилось внутрішнє тестування.

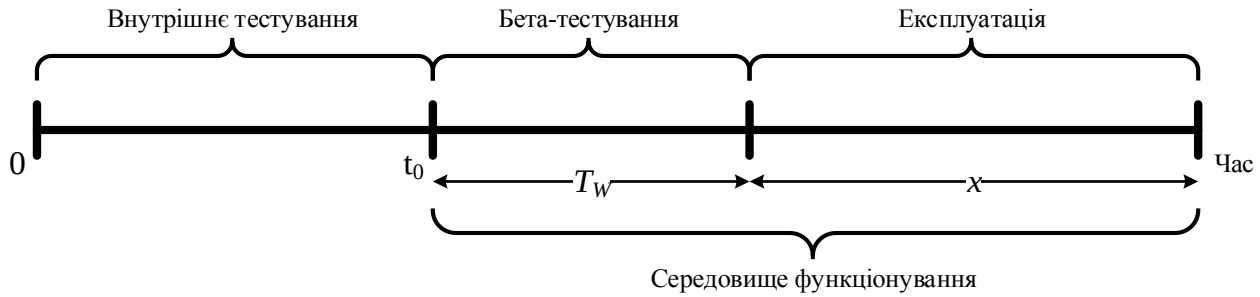


Рис. 1.4. Модель збільшення прибутку ПЗ.

В цій моделі враховуються наступні фактори вартості:

1. Існують сталі організаційні витрати (пускові затрати) на початку внутрішнього тестування.
2. Вартість тестування лінійно залежить від часу внутрішнього тестування.
3. Вартість усунення відмов під час періоду внутрішнього тестування пропорційна загальному часу, необхідному для усунення всіх відмов, виявлених протягом цього періоду.
4. Вартість усунення відмов в період бета-тестування пропорційна загальному часу усунення всіх відмов, виявлених на проміжку $[T, T + T_W]$.
5. Усунення відмови займає певний час, і вважається, що час, необхідний для усунення кожної відмови відповідає усіченому експоненційному розподілу.
6. Існують штрафи за відмови програмного забезпечення після його офіційного випуску, тобто після етапу бета-тестування.
7. Компанії, які розробляють програмне забезпечення, отримують економічну вигоду від безвідмовного виконання їх програмного забезпечення під час бета-тестування та експлуатації.

Очікуваний час, необхідний для усунення кожної відмови під час внутрішнього тестування μ_y задається рівнянням (1.34). Аналогічно, очікуваний

час, необхідний для усунення відмови μ_W на стадії бета-тестування задається рівнянням (1.44).

Очікуваний чистий прибуток від процесу розробки програмного забезпечення $E(T)$ визначається як економічний дохід від надійного програмного забезпечення, що перевищує очікувану загальну вартість розробки програмного забезпечення [124]: $E(T) = \text{Очікуваний дохід від надійності} - (\text{Загальна вартість розробки} + \text{Вартість ризику})$.

В [9] доведено теорему, яка визначає оптимальне значення T^*, яке максимізує очікуваний чистий дохід від процесу розробки програмного забезпечення $E(T)$. Більш детальний огляд зазначених моделей оптимального релізу ПЗ наведено в [9, 125].

Підсумовуючи, можна зазначити, що окремою важливою задачею, яка постає перед розробниками моделей надійності ПЗ будь-якого типу є прийняття рішень про випуск програмного продукту [14], яке полягає в переході з етапу розробки до етапу експлуатації ПЗ. При цьому приходиться долати протиріччя та знаходити компроміс між раннім виходом на ринок з отриманням усіх переваг мінімальної конкуренції, та відкладанням релізу з метою підвищення якості продукту та покращення функціональності. Незважаючи на численні дослідження [126–128] це питання все ще залишається відкритим, і не існує достатньо універсального критерію, який може бути застосований до усіх типів вхідних даних.

1.7. Висновки до розділу 1

Таким чином, вирішення актуальної проблеми підвищення надійності сучасних ПАС вимагає розроблення моделей, методів та засобів аналізу надійності програмних засобів, які є складовими таких систем. В свою чергу, розв'язання протиріччя між складністю сучасного ПЗ з одного боку та високими вимогами до його надійності – з іншого, вимагає розв'язання наступних науково-практичних задач:

- розроблення моделей і методів оцінювання показників надійності ПЗ за даними їх тестування, причому такі моделі, для підвищення ступеня їх адекватності, повинні враховувати складність програмних засобів;
- розроблення моделей і методів оцінювання показників надійності складних програмних систем на основі даних про поведінку надійності їх складових, причому внаслідок складності таких ПС, ці моделі і методи повинні враховувати взаємозалежність потоку управління програми;
- побудова узагальненого методу аналізу надійності ПЗ з урахуванням його складності на різних етапах ЖЦ;
- розроблення засобів підтримки прийняття рішень стосовно етапів тестування та випуску в експлуатацію програмних продуктів із урахуванням вимог до їх надійності.

Як видно з наведеного огляду літератури, перша задача може бути вирішена шляхом побудови моделей надійності ПЗ на основі НПП з урахуванням складності програмних засобів. Для вирішення другої задачі доцільним є використання ЛМВП та розроблення відповідних методів аналізу надійності ПЗ. Вирішення задачі підвищення ефективності розроблення програмних продуктів з урахуванням вимог до їх надійності слід вирішувати шляхом побудови моделей зрілості ПЗ або ж використанням критеріїв та процедур, які дадуть змогу отримати кількісні оцінки для прийняття рішення щодо випуску ПЗ в експлуатацію, в тому числі за рахунок прогнозування показників надійності ПЗ. Важливим прикладним аспектом оцінювання показників надійності ПЗ за результатами його випробувань та підвищення його якості, є створення методів автоматизованого формування сценаріїв тестування ПЗ з урахуванням метрик покриття коду.

РОЗДІЛ 2. ОЦІНЮВАННЯ ПОКАЗНИКІВ НАДІЙНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ РЕЗУЛЬТАТІВ ТЕСТУВАННЯ

В цьому розділі розглянуто модель надійності ПЗ на основі НПП, яка включає як параметр динамічний показник складності програмного засобу. Ця модель використовується для оцінювання таких показників надійності ПЗ, як загальна кількість помилок в коді, параметр потоку відмов, інтенсивність відмов, імовірність безвідмовної роботи тощо, за результатами його випробування (тестування). Досліджено поведінку параметрів моделі та встановлено діапазони показника складності ПЗ. Наведено результати верифікації даної моделі на основі емпіричних даних та її порівняння з S-подібною моделлю та моделлю Goel–Okumoto, як найбільш поширеними моделями на основі НПП. Показано, що МНПС, за рахунок уведення динамічного показника складності, відноситься до класу узагальнених моделей на основі НПП і дозволяє описувати поведінку показників надійності з більшим ступенем адекватності порівняно з відомими моделями на основі НПП. Наведено результати впливу вибору моделі надійності ПЗ на оцінювання показників надійності відмовостійкої ПАС, і показано, що МНПС дає змогу підвищити точність оцінювання інтенсивності відмов ПЗ на три порядки, та підвищити точність оцінювання функції та коефіцієнта готовності ПАС. На основі МНПС розроблено метод оцінювання показників надійності ПЗ з урахуванням його складності на основі результатів тестування та проведено його верифікацію з використанням емпіричних даних тестування промислового програмного засобу.

Основні результати і висновки, викладені в цьому розділі, опубліковані в працях [38, 66, 129–135].

2.1. Узагальнена модель надійності ПЗ на основі НПП з показником складності

Як уже зазначалось в розділі 1, існуючі моделі надійності ПЗ на основі НПП недостатньо адекватно описують поведінку реального ПЗ внаслідок зростання складності останнього. З метою усунення цих недоліків, підвищення

точності оцінювання та аналізу поведінки показників надійності ПЗ на основі даних статистичних випробувань, якими у випадку ПЗ зазвичай є результати тестування програми, в [66, 129, 131, 136] було розроблено та описано модель надійності ПЗ на основі НПП з динамічним показником складності, який є параметром моделі та визначається на основі експериментальних даних і набуває значення додатного числа.

Як і переважна більшість моделей на основі НПП (див. підрозділ 1.4), модель надійності з показником складності (МНПС) будувалась на основі наступних припущень [66, 129].

1. *Кількість помилок, виявлених протягом кожного інтервалу часу що не перетинаються, є незалежною;*

Кількості помилок, виявлених на відповідних часових інтервалах є незалежними для будь-якої скінченої кількості часових проміжків. Кількість помилок на заданий момент часу відповідає пуассоновому розподілу. Також вважається, що кумулятивна кількість помилок є обмеженою, не спадною функцією часу.

Це припущення означає, що помилки ПЗ є непов'язані між собою, при цьому одній відмові, отриманій під час тестування, відповідає одна помилка в коді програми. Крім того, загальна кількість помилок в програмній системі є скінченною величиною. Останнє припущення видається достатньо коректним, в той час як на рахунок першого існують різні думки [95]. Дослідження, описані в працях [75, 95, 105] дають можливість стверджувати, що на пізніх етапах тестування, коли з системи видалена достатня кількість помилок, це припущення є цілком справедливим, в той час як до висновків моделей цього класу, отриманих на початкових етапах тестування, слід ставитись з обережністю і підтверджувати їх статистичною перевіркою гіпотези про належність експериментальних даних до розподілу Пуассона із заданим ступенем достовірності.

2. Виявлені помилки виправляються, час на виправлення не враховується;

В моделях, що базуються на цьому припущенні, береться за основу, що програмна система проходить через процес виправлення помилок, тобто в процесі тестування виявляють нові помилки. Таким чином це припущення принаймні є допустимим для багатьох ситуацій в тестуванні промислових програмних продуктів. Інколи, процес тестування продовжується без видалення виявленої помилки. В цьому випадку можна припустити, що подальший процес виявлення помилки відбувається в такий спосіб, наче ця помилка була усунена. Якщо, не зважаючи на це, помилка залишається в тій частині коду програми, яка надалі проходить тестування, це припущення буде справджуватись тільки тоді, коли ця помилка буде вилучена до такого тестування або ж при створенні нових тестових наборів, які оминають цю помилку.

3. При виправленні виявленої помилки не вноситься нових помилок;

Зміст цього припущення полягає в твердженні, що процес виявлення помилок, який моделюється, відповідає монотонному шаблону. Це означає, що після видалення помилок з системи, в ній залишається менше помилок, ніж було до того. В загальному випадку це може не відповідати дійсності, оскільки під час виправлення виявленої помилки можуть бути змінені інші ділянки програми з уведенням нових помилок в систему. Тому дане припущення в загальному вважається обмежувальним припущенням стосовно моделей надійності ПЗ. Єдиний спосіб задовольнити цю умову полягає у ретельному контролі процесу виправлення, так щоб при усуненні помилки не було введено жодної нової. Якщо ж, однак, введені під час виправлення помилки складають невелику частку в їх загальній кількості, то відхилення від цього припущення матиме мінімальний практичний вплив на результати використання моделі.

4. Інтенсивність виявлення помилок спадає з часом;

Таке припущення означає, що ПЗ покращується в статистичному сенсі в процесі тестування. Це припущення є прийнятним в більшості випадків і може бути обґрунтоване таким чином. В процесі тестування з програми вилучаються помилки. Вони видаляються перед продовженням процесу тестування, або ж

вони не видаляються, а тестування охоплює інші частини програми. В першому випадку подальша інтенсивність виявлення помилок зменшується в явному вигляді. В іншому випадку інтенсивність виявлення помилок (стосовно цілого програмного продукту) зменшується неявним чином, адже в подальшому тестується все менша частина коду програми.

5. *Імовірності виявлення будь-якої помилки на будь-якій ітерації є однаковими;*

Це припущення означає, що кожна помилка, яка залишилась в програмі, має однакову ймовірність бути виявленою в заданому інтервалі тестування між відмовами. Це припущення є доволі правдоподібним, якщо тестові набори вибираються так, щоб забезпечити однакову ймовірність виконання усіх частин програмного коду. Однак, якщо один зі шляхів виконання програми тестується більш ретельно ніж інші, в ньому буде виявлено більше помилок ніж в останніх. Помилки, які містяться в не реалізовуваних, або жодного разу не тестованих частинах коду програми очевидно будуть мати низьку або ж нульову ймовірність виявлення.

6. *Інтенсивність відмов є функцією кількості помилок, що залишились в програмі;*

Це припущення означає, що усі помилки, які залишились в коді програми з однаковою імовірністю виявляться при експлуатації системи, і використовується для оцінювання надійності ПЗ на основі кількості залишкових помилок. Крім того, це припущення говорить, що зовнішні фактори не впливають на інтенсивність відмов ПЗ. Якщо використання програми однорідне, то це припущення є однозначно коректним. Якщо ж деякі частини ПЗ будуть виконуватись з більшою імовірністю (частотою) ніж інші, це припущення не справджуватиметься. Разом з тим, надійність системи може бути перерахована з урахуванням інформації про операційний профіль програми. Іншими словами, в цьому випадку оцінювання надійності з урахуванням моделі використання є більш придатним, ніж на основі кількості залишкових помилок. Якщо ж, однак,

такої інформації немає, єдиним прийнятним припущенням є припущення про однорідне використання.

7. *В ролі аргументу інтенсивності виявлення помилок використовується час;*

Більшість моделей використовують час в ролі аргументу для визначення змін інтенсивності виявлення помилок. Таке використання передбачає, що трудомісткість тестування пропорційна календарному чи процесорному часу. Також час в загальному достатньо легко піддається кількісному вимірюванню і більшість протоколів тестування ведуться із зазначенням часових рамок. Іншою перевагою використання часу в ролі аргументу функції є те, що час згладжує різниці в трудомісткості тестування. Якщо ж, однак, процес тестування не пропорційний часу, моделі все одно залишаються валідними для будь-якого іншого адекватного аргументу. Наприклад, можна використовувати кількість протестованих рядків коду, кількість протестованих функцій, кількість виконаних тестових наборів, чи нормувати затрати на процес тестування, наприклад, в людино-годинах.

8. *Процес тестування є репрезентативною вибіркою експлуатаційного використання програми.*

Це припущення є необхідним у випадку, коли оцінка надійності, побудована на основі процесу тестування, проектується на етап експлуатації ПЗ. Найбільш важливим це припущення є в моделях на основі області вхідних даних. Моделі на основі часу між відмовами та кількості помилок також повинні базувати на цьому припущенні, якщо метою є визначення експлуатаційної надійності. Тестові набори зазвичай вибираються так, щоб забезпечити усі функціональні вимоги до системи. Однак конкретний користувач програмної системи може використовувати функції програми в іншій пропорції, ніж це використовувалось на етапі тестування. В цьому випадку тестування не відображатиме експлуатаційне використання програми. Якщо інформація про шаблонні схеми використання програмної системи наявна, процес тестування повинен бути відповідно модифікований.

9. *Вигляд функції зростання надійності ПЗ (експоненційний чи S-подібний) залежить від складності програми.*

Це припущення говорить про те, що поведінка кривої інтенсивності відмов ПЗ залежить тільки від його складності, під якою розуміємо деяку комплексну величину, яка залежить як від кількості рядків коду, так і від більш складних метрик складності (цикломатична складність, якість програмної документації [137] тощо), і фактично відносить цю модель до узагальнених моделей надійності (див. напр. [26]). Таке припущення є справедливим, якщо справджується припущення про досконале відлагодження, незалежність відмов різних помилок та залежності інтенсивності відмов ПЗ тільки від помилок, які залишаються в коді програми (припущення №№ 1, 3, 6).

Слід зазначити, що аргументи, наведені в обґрунтуванні припущень моделі, не є універсальними для будь-якого процесу розробки ПЗ, оскільки процес розробки ПЗ залежить від середовища розробки, технологій програмування тощо. Положення і твердження, справедливі для одного середовища, можуть не справджуватись для іншого і т.д. Тому навіть коректні і обґрунтовані припущення, які справджуються, наприклад, під час тестування однієї функції чи підсистеми, можуть не бути такими при подальшому тестуванні цієї функції. Остаточне рішення про прийнятність припущень, що лежать в основі певної моделі, та застосовність моделі в даній конкретній ситуації повинно прийматись користувачем моделі.

На основі цих припущень в [66] було запропоновано МНПС, в якій параметр потоку відмов має наступний вигляд:

$$\lambda(t) = \alpha\beta^{s+1}t^{s}\exp(-\beta t), \qquad (2.1)$$

де α – коефіцієнт, який характеризує загальну кількість відмов, які були виявлені в програмі від початку спостереження, β – коефіцієнт, що характеризує швидкість зміни функції параметру потоку відмов (завжди $\beta > 0$), s – динамічний показник складності ПЗ.

Для функції параметру потоку відмов вигляду (2.1) функція кумулятивної кількості помилок ПЗ має вигляд:

$$\mu(t) = \int_0^t \lambda(\tau)d\tau = \alpha\left[-\beta^s t^s e^{-\beta t} + s\Gamma_{\beta t}(s)\right] \quad (2.2)$$

де, $\Gamma_z(p) = \int_0^z t^{p-1}e^{-t}dt$, $(Re\ p > 0)$ – неповна гамма-функція.

Такий вираз кумулятивної функції $\mu(t)$ отримаємо, врахувавши вигляд функції параметру потоку відмов (2.1) та провівши наступні заміни $\beta\tau = v,\ d\tau = \frac{dv}{\beta}, \tau = \frac{v}{\beta}, v = \beta t$:

$$\mu(t) = \int_0^t \lambda(\tau)d\tau = \int_0^t \alpha\beta^{s+1}\tau^s e^{-\beta\tau}d\tau = \alpha\beta^{s+1}\int_0^t \tau^s e^{-\beta\tau}d\tau =$$

$$= \alpha(-v^s e^{-v}\Big|_0^{\beta t} + s\int_0^{\beta t} v^{s-1}e^{-v}dv) = -\alpha\beta^s t^s e^{-\beta t} + s\alpha\Gamma_{\beta t}(s).$$

Зауважимо, що при $s = 1$, оскільки $\Gamma(1) = \int_0^{\beta t} e^{-\tau}d\tau = -e^{-\tau}|_0^{\beta t} = 1 - e^{-\beta t}$, отримаємо:

$\mu(t) = \alpha\Gamma_{\beta t}(1) - \alpha\beta t e^{-\beta t} = \alpha\left(1 - e^{-\beta t}\right) - \alpha\beta t e^{-\beta t} = \alpha\left(1 - e^{-\beta t} - \beta t e^{-\beta t}\right) = \alpha(1 - e^{-\beta t}(1 + \beta t))$.

При $s = 1$ вигляд функції параметру потоку відмов (2.1) та кумулятивної функції (2.2) є ідентичними вигляду відповідних функцій S-подібної моделі.

Загальна кількість помилок в ПЗ визначається кумулятивною функцією при $t \to \infty$, таким чином:

$$\mu(\infty) = \alpha s\Gamma(s), \quad (2.3)$$

де $\Gamma(s)$ – гама-функція.

Отже, аналітичний вигляд описаної рівняннями (2.1) та (2.2) моделі дає змогу узагальнити вираз для загальної кількості помилок в (2.3), яка залежить від величини та складності ПЗ і визначається параметрами моделі. Крім того, частковий випадок при $s = 1$ (S-подібна модель) з урахуванням того, що $\Gamma(1) =$

1, дає значення $\mu(\infty) = \alpha$, що відповідає S-подібній моделі. Вирази (2.1) та (2.2) є моделлю з динамічним показником складності програмного засобу.

Під складністю ПЗ розуміємо комплексний показник, який пов'язаний з метриками складності коду програмного продукту [133].

Модель надійності з показником складності [66] можна віднести до "узагальнених пуассонових моделей", разом з узагальненою пуассоновою моделлю [26], узагальненою моделлю неоднорідного пуассонового процесу Goel [138] та моделлю Тимошенко–Дідковської [65]. Параметр потоку відмов [85] в моделі [66] описується виразом (2.1).

Для дослідження поведінки функції параметру потоку відмов (2.1) та встановлення її екстремумів знайдемо похідну цієї функції по часу:

$$\frac{d\lambda(t)}{dt} = \alpha\beta^{s+1}\left(\frac{dt^s}{dt}e^{-\beta t} + \frac{de^{-\beta t}}{dt}t^s\right) = \lambda(t)(st^{-1} - \beta). \quad (2.4)$$

Прирівнявши рівняння (2.4) до нуля отримаємо:

$$\alpha\beta^{s+1}t^s e^{-\beta t}(st^{-1} - \beta) = 0,$$

звідки

$$t^s(st^{-1} - \beta) = 0. \quad (2.5)$$

Розв'язавши рівняння (2.5) отримаємо значення часу, при якому функція $\lambda(t)$ є максимальною:

$$t_{max} = \frac{s}{\beta}. \quad (2.6)$$

Як видно з (2.6) положення максимуму параметру потоку відмов програмного продукту, на відміну від усіх відомих моделей надійності ПЗ, залежить як від якості тестування (параметр β [28]), так і від складності програмного продукту, що тестується. Зсув максимуму функції інтенсивності відмов по часовій шкалі в залежності від значення показника складності ілюструє рис. 2.1 (криві 2–5). Усі криві на рис. 2.1 були побудовані з використанням однакових значень параметрів α та β (a та b для моделі Goel–Okumoto), а значення параметра s для кривих 2–5 становило 0,5; 1; 2 та 2,5

відповідно. Зауважимо, що крива, яка відповідала параметру потоку відмов у випадку S-подібної моделі [63] повністю співпадала з кривою 3.

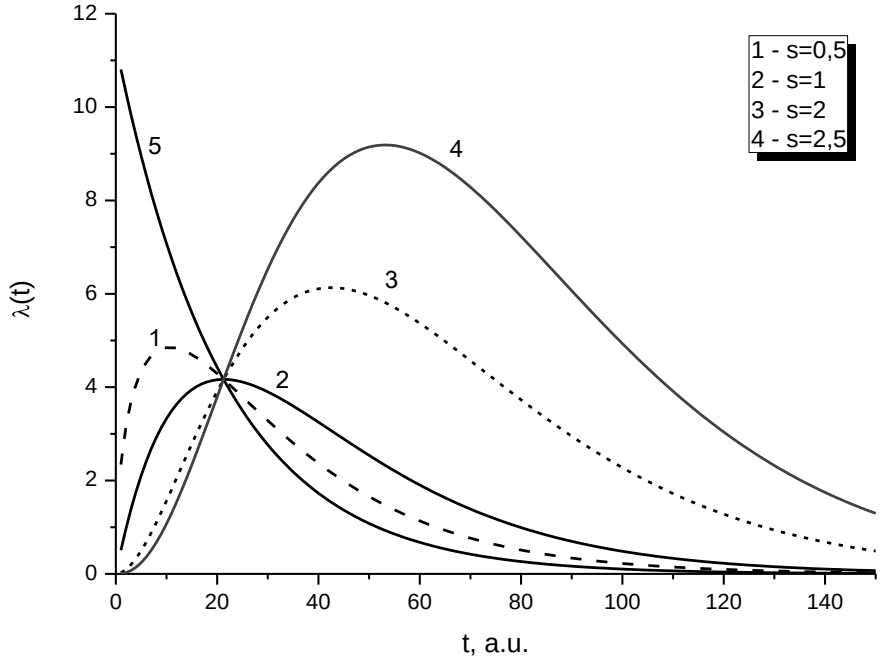


Рис. 2.1. Залежність поведінки функції параметру потоку відмов з часом для узагальненої моделі з показником складності (криві 1–4) та моделі Goel–Okumoto (крива 5).

Проаналізуємо максимальне значення параметру потоку відмов λ_{max} та його залежність від параметрів моделі.

$$\lambda_{max}(\alpha,\beta,s)=\lambda(t_{max})=\alpha\beta^{s+1}\left(\frac{s}{\beta}\right)^{s}e^{-\beta\frac{s}{\beta}}=\alpha\beta s^{s}e^{-s}. \tag{2.7}$$

Обчисливши часткові похідні функції (2.7) та прирівнявши їх до нуля знайдемо екстремальні точки:

$$\frac{\partial\lambda_{max}(\alpha,\beta,s)}{\partial\alpha}=\beta s^{s}e^{-s}, \tag{2.8}$$

$$\frac{\partial\lambda_{max}(\alpha,\beta,s)}{\partial\beta}=\alpha s^{s}e^{-s}, \tag{2.9}$$

$$\frac{\partial\lambda_{max}(\alpha,\beta,s)}{\partial s}=\alpha\beta s^{s}(\ln s+1)e^{-s}-\alpha\beta s^{s}e^{-s}=\alpha\beta s^{s}e^{-s}\ln s, \tag{2.10}$$

Як видно з рівнянь (2.8) та (2.9) за параметрами α і β функція (2.7) є монотонно зростаючою без екстремумів (за винятком нульових значень цих

параметрів, що не має фізичного змісту). Разом з тим, з рівняння (2.10) видно, що поведінка максимального значення параметру потоку відмов в залежності від показника складності s є складнішою і має екстремум в точці $s = 1$. Неважко показати, що екстремальна точка є точкою мінімуму функції.

Зауважимо, що при $s = 0$ функція параметру потоку відмов (2.11) співпадає з такою для моделі Goel–Okumoto (2.12):

$$\lambda(t)|_{s=0} = \alpha\beta e^{-\beta t} \tag{2.11}$$

$$\lambda(t) = abe^{-bt}, \tag{2.12}$$

а при $s = 1$ – S-подібної моделі (вирази (2.13) та (2.14) відповідно):

$$\lambda(t)|_{s=1} = \alpha\beta^2 te^{-\beta t}, \tag{2.13}$$

$$\lambda(t) = \alpha\beta^2 te^{-\beta t}. \tag{2.14}$$

Для моделі Goel–Okumoto параметр a має зміст очікуваної загальної кількості помилок, яка буде виявлена при $t \to \infty$, а параметр b – коефіцієнт виявлення відмов на одну помилку (показує швидкість виявлення відмов за одиницю часу на одну помилку, присутню в програмі) [26].

В МНПС параметр α характеризує загальну кількість помилок в програмному продукті, але на відміну від моделі Goel–Okumoto не дорівнює їй (очікувана загальна кількість помилок, яка буде виявлена при $t \to \infty$ становить $\alpha s\Gamma(s)$, тобто містить модифікатор, який залежить від складності ПЗ); параметр β так само має розмірність обернену до розмірності часу і характеризує швидкість виявлення відмов, віднесену до однієї помилки програми; параметр s є показником складності програмного продукту [66] і визначається деякою комплексною метрикою ПЗ. Крім того, з рис. 2.1 видно, що на тривалість процесу виявлення помилок впливає не тільки параметр β, але й параметр s, що відповідає практиці проведення тестування та виявлення помилок для програмних продуктів різної величини та складності.

Максимальне значення параметру потоку відмов в моделі Goel–Okumoto становить (за формулою (2.12)) ab. Порівнявши цей вираз з виразом (2.7) можна припустити, що максимальне значення параметру потоку відмов (так само, як і

загальна кількість помилок) у випадку МНПС містить базове значення (значення при $s = 0$, що співпадає з моделлю Goel–Okumoto) помножене на деякий модифікатор (в даному випадку $s^s e^{-s}$), який залежить від складності ПЗ. Припустивши, що значення такого модифікатора не може бути більшим за одиницю отримаємо граничні значення параметра s для МНПС – значення параметра s не може перевищувати числа Ейлера e. Таке припущення можна обґрунтувати взявши до уваги, що у випадку базового значення усі помилки виявляються з максимальною інтенсивністю при $t = 0$ і за мінімальний загальний час (див. рис. 2.1, крива 1), а збільшення тривалості виявлення тієї самої кількості помилок не може привести до більшого миттєвого значення параметру потоку відмов в довільний момент часу. Відтак, з рівняння (2.7) отримаємо максимальне значення параметру потоку відмов для трьох основних значень показника складності (0 – мінімальне значення; 1 – значення, що відповідає мінімуму параметру потоку відмов; e – максимальне значення):

$$\lambda_{max}(\alpha, \beta, s)|_{s=0} = \alpha\beta,$$

$$\lambda_{max}(\alpha, \beta, s)|_{s=1} = \frac{\alpha\beta}{e},$$

$$\lambda_{max}(\alpha, \beta, s)|_{s=e} = \alpha\beta,$$

Залежність нормованого максимального значення параметру потоку відмов від показника складності програмного продукту наведено на рис. 2.2.

Діапазони значень показника складності програмного продукту визначимо з умови рівності площі під кривою максимального значення параметру потоку відмов (рис. 2.2) в кожному діапазоні. Шляхом ітеративного підбору верхньої межі інтегралу, записаного на основі виразу (2.7), можна визначити наступні діапазони показника складності програмного продукту (рис. 2.2):

1. $s \in [0; 0{,}66)$ ПЗ можна вважати нескладним;
2. $s \in [0{,}66; 1{,}6)$ ПЗ можна вважати середньої складності;
3. $s \in [1{,}6; 2{,}28)$ ПЗ можна вважати складним;
4. $s \in [2{,}28; e]$ – дуже складним.

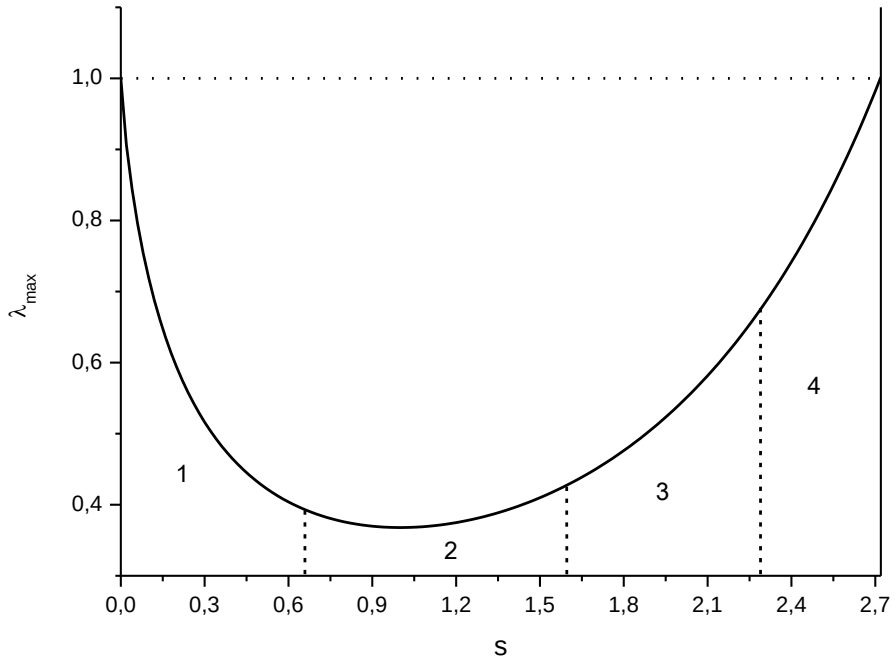


Рис. 2.2. Залежність нормованого максимального значення параметру потоку відмов від показника складності s.

Поведінка параметру потоку відмов може бути пояснена наступним чином. Зі збільшенням ступеня складності програмного продукту максимальний параметр потоку відмов зменшується за абсолютним значенням (рис. 2.2) та зсувається в часі (рис. 2.1) оскільки тестувальникам потрібно все більше часу на тестування усіх функцій продукту. Для проектів середньої складності значення параметру потоку відмов майже не залежить від складності і проходить через точку мінімуму, при цьому далі зсуваючись в бік більших значень часової шкали. Зі збільшенням показника складності максимальне значення параметру потоку відмов продовжує зростати і при $s = e$ за абсолютним значенням співпадає з таким для випадку $s = 0$. На думку автора [133] це може бути пояснено зростанням кількості тестувальників, більшою модульністю, яка повною мірою використовує переваги об'єктно-орієнтованого підходу тощо. При цьому загальна тривалість процесу виявлення помилок монотонно зростає (див. рис. 2.1), а максимум параметру потоку відмов настає все пізніше зі зростанням складності продукту, що відповідає якісній картині процесу тестування промислового ПЗ.

Для опису емпіричних результатів про відмови програмних продуктів необхідно отримати точкові оцінки параметрів моделі. Для цього у [66, 136] було застосовано ММП.

Якщо на проміжках $(t_i, t_{i+1}]$, $i = \overline{0,n}$ виявлено m_i помилок, тоді $\sum_{i=1}^{n} m_i = k$. Припустивши, що кількість відмов відповідає розподілу Пуассона, побудуємо функцію $L(\alpha, \beta, s)$ максимальної правдоподібності:

$$L(\alpha, \beta, s) = \prod_{i=1}^{n} \frac{[\mu(t_i) - \mu(t_{i-1})]^{m_i}}{m_i!} \exp\big(\mu(t_{i-1}) - \mu(t_i)\big)$$

Враховуючи вигляд $\mu(t)$ з рівняння (2.2) отримуємо вираз:

$$L(\alpha, \beta, s) = \prod_{i=1}^{n} \frac{1}{m_i!} \left[\alpha s \Gamma_{\beta t_i}(s) - \alpha \beta^s t_i^s e^{-\beta t_i} - \alpha s \Gamma_{\beta t_{i-1}}(s) + \alpha \beta^s t_{i-1}^s e^{-\beta t_{i-1}}\right]^{m_i} \times$$

$$\times e^{\mu(t_{i-1}) - \mu(t_i)} =$$

$$= \prod_{i=1}^{n} \frac{\alpha^{m_i}}{m_i!} \left[s\left(\Gamma_{\beta t_i}(s) - \Gamma_{\beta t_{i-1}}(s)\right) + \beta^s \left(t_{i-1}^s e^{-\beta t_{i-1}} - t_i^s e^{-\beta t_i}\right)\right]^{m_i} \times e^{\mu(t_{i-1}) - \mu(t_i)}.$$

Функція максимальної правдоподібності L задовольняє таким умовам:

1. диференційована при довільних значеннях вибірки;
2. досягає максимуму в інтервалі можливих значень.

Значення оцінок величин α, β, s дає розв'язок системи рівнянь:

$$\begin{cases} \dfrac{\partial L(\alpha, \beta, s)}{\partial \alpha} = 0, \\ \dfrac{\partial L(\alpha, \beta, s)}{\partial \beta} = 0, \\ \dfrac{\partial L(\alpha, \beta, s)}{\partial s} = 0. \end{cases} \tag{2.15}$$

Без зменшення загальності замість функції L у системі (2.15) розглядаємо функцію $\ln L$:

$$\ln L(\alpha, \beta, s) = \sum_i \ln(m_i!)^{-1} + \ln \alpha k +$$

$$+ \sum_i m_i \ln \left[s\left(\Gamma_{\beta t_i}(s) - \Gamma_{\beta t_{i-1}}(s)\right) + \beta^s \left(t_{i-1}^s e^{-\beta t_{i-1}} - t_i^s e^{-\beta t_i}\right)\right] +$$

$$+\sum_i \big(\mu(t_{i-1}) - \mu(t_i)\big) = -\sum_i \ln(m_i!) + k\ln\alpha\ +$$

$$+\sum_i m_i \ln\left[s\left(\Gamma_{\beta t_i}(s) - \Gamma_{\beta t_{i-1}}(s)\right) + \beta^s\left(t_{i-1}^s e^{-\beta t_{i-1}} - t_i^s e^{-\beta t_i}\right)\right] +$$

$$+\alpha\beta^s t_n^s e^{-\beta t_n} - \alpha s\Gamma_{\beta t_n}(s),$$

Здійснимо потрібні перетворення:

$$\frac{\partial \ln L}{\partial \alpha} = \frac{k}{\alpha} + \beta^s t_n^s e^{-\beta t_n} - s\Gamma_{\beta t_n}(s),$$

$$\frac{k}{\alpha} + \beta^s t_n^s e^{-\beta t_n} - s\Gamma_{\beta t_n}(s) = 0,$$

та отримаємо перше рівняння для системи (2.15):

$$\alpha = \frac{k}{s\Gamma_{\beta t_n}(s) - \beta^s t_n^s e^{-\beta t_n}}.$$

Здійснимо аналогічні перетворення для другого рівняння системи (2.15):

$$\frac{\partial \ln L}{\partial \beta} =$$

$$= \sum_i m_i \frac{s\left(\Gamma_{\beta t_i}(s) - \Gamma_{\beta t_{i-1}}(s)\right)'_\beta + s\beta^{s-1}\left(t_{i-1}^s e^{-\beta t_{i-1}} - t_i^s e^{-\beta t_i}\right) + \beta^s\left(-t_{i-1}^{s+1} e^{-\beta t_{i-1}} + t_i^{s+1} e^{-\beta t_i}\right)}{s\left(\Gamma_{\beta t_i}(s) - \Gamma_{\beta t_{i-1}}(s)\right) + \beta^s\left(t_{i-1}^s e^{-\beta t_{i-1}} - t_i^s e^{-\beta t_i}\right)}$$

+

$$+\alpha s\beta^{s-1} t_n^s e^{-\beta t_n} - \alpha\beta^s t_n^{s+1} e^{-\beta t_n} - \alpha s\left(\Gamma_{\beta t_n}(s)\right)'_\beta =$$

$$= -\alpha t_n^{s+1} e^{-\beta t_n} + \sum_i \frac{m_i\left(t_i^{s+1} e^{-\beta t_i} - t_{i-1}^{s+1} e^{-\beta t_{i-1}}\right)}{s\left(\Gamma_{\beta t_i}(s) - \Gamma_{\beta t_{i-1}}(s)\right) + \beta^s\left(t_{i-1}^s e^{-\beta t_{i-1}} - t_i^s e^{-\beta t_i}\right)}.$$

Оскільки $\left(\Gamma_{\beta t_n}(s)\right)'_\beta = \left(\int_0^{\beta t} \tau^{s-1} e^{-\tau} d\tau\right)'_\beta = t(\beta t)^{s-1} e^{-\beta t} = \beta^{s-1} t^s e^{-\beta t}$, та

$$s\left(\beta^{s-1} t_i^s e^{-\beta t_i} - \beta^{s-1} t_{i-1}^s e^{-\beta t_{i-1}}\right) + s\beta^{s-1} t_{i-1}^s e^{-\beta t_{i-1}} - s\beta^{s-1} t_i^s e^{-\beta t_i} +$$

$$+\beta^s t_i^{s+1} e^{-\beta t_i} - \beta^s t_{i-1}^{s+1} e^{-\beta t_{i-1}} = \beta^s\left(t_i^{s+1} e^{-\beta t_i} - t_{i-1}^{s+1} e^{-\beta t_{i-1}}\right);$$

$$\alpha s\beta^{s-1} t_n^s e^{-\beta t_n} - \alpha s\beta^s t_n^{s+1} e^{-\beta t_n} - \alpha s\left(\Gamma_{\beta t_n}(s)\right)'_\beta =$$

$$= \alpha s\beta^{s-1} t_n^s e^{-\beta t_n} - \alpha s\beta^s t_n^{s+1} e^{-\beta t_n} - \alpha s\beta^{s-1} t_n^s e^{-\beta t_n} = -\alpha s\beta^s t_n^{s+1} e^{-\beta t_n}.$$

Для того, щоб мати вигляд третього рівняння системи (2.15) достатньо зробити такі перетворення:

$$\frac{\partial \ln L}{\partial s} = \sum_i m_i \frac{\Gamma_{\beta t_i}(s) - \Gamma_{\beta t_{i-1}}(s) + s\left(\Gamma_{\beta t_i}(s) - \Gamma_{\beta t_{i-1}}(s)\right)'_s}{s\left(\Gamma_{\beta t_i}(s) - \Gamma_{\beta t_{i-1}}(s)\right) + \beta^s\left(t_{i-1}^s e^{-\beta t_{i-1}} - t_i^s e^{-\beta t_i}\right)} +$$

$$+\beta^s \ln\beta\left(t_{i-1}^s e^{-\beta t_{i-1}} - t_i^s e^{-\beta t_i}\right) + \beta^s\left(t_{i-1}^s \ln t_{i-1}\, e^{-\beta t_{i-1}} - t_i^s \ln t_i\, e^{-\beta t_i}\right) +$$

$$+\alpha\beta^s \ln\beta t_n^s\, e^{-\beta t_n} + \alpha\beta^s t_n^s \ln\beta t_n\, e^{-\beta t_n} - \alpha\Gamma_{\beta t_n}(s) - \alpha s\left(\Gamma_{\beta t_n}(s)\right)'_s =$$

$$= \sum_i m_i \frac{\Phi_{\beta t_i t_{i-1}}(s) + s\mathrm{F}_{\beta t_i t_{i-1}}(s) + \beta^s t_{i-1}^s e^{-\beta t_{i-1}} \ln(\beta t_{i-1}) - \beta^s t_i^s e^{-\beta t_i} \ln(\beta t_i)}{s\Phi_{\beta t_i t_{i-1}}(s) + \beta^s(t_{i-1}^s e^{-\beta t_{i-1}} - t_i^s e^{-\beta t_i})} +$$

$$+\alpha\beta^s t_n^s e^{-\beta t_n} \ln(\beta t_n) - \alpha\Gamma_{\beta t_n}(s) - \alpha s \int_0^{\beta t} \tau^{s-1} \ln\tau\, e^{-\tau} d\tau,$$

де

$$\left(\Gamma_{\beta t_n}(s)\right)' = \left(\int_0^{\beta t} \tau^{s-1} e^{-\tau} d\tau\right)' = \int_0^{\beta t} \tau^{s-1} \ln\tau\, e^{-\tau} d\tau,$$

$$\Phi_{\beta t_i t_{i-1}}(s) = \int_{\beta t_{i-1}}^{\beta t_i} \tau^{s-1} e^{-\tau} d\tau, \qquad \mathrm{F}_{\beta t_i t_{i-1}}(s) = \int_{\beta t_{i-1}}^{\beta t_i} \tau^{s-1} \ln\tau\, e^{-\tau} d\tau,$$

$$\beta^s \ln\beta t_{i-1}^s\, e^{-\beta t_{i-1}} - \beta^s \ln\beta t_i^s\, e^{-\beta t_i} + \beta^s t_{i-1}^s \ln t_{i-1}\, e^{-\beta t_{i-1}} - \beta^s t_i^s \ln t_i\, e^{-\beta t_i} =$$

$$= \beta^s t_{i-1}^s e^{-\beta t_{i-1}} \ln(\beta t_{i-1}) - \beta^s t_i^s e^{-\beta t_i} \ln(\beta t_i),$$

$$\alpha\beta^s \ln\beta t_n^s\, e^{-\beta t_n} + \alpha\beta^s t_n^s \ln t_n\, e^{-\beta t_n} = \alpha\beta^s t_n^s e^{-\beta t_n} \ln(\beta t_n).$$

Отже, система (2.15) прийме вигляд:

$$\begin{cases} \alpha = \dfrac{k}{s\Gamma_{\beta t_n}(s) - \beta^s t_n^s e^{-\beta t_n}}; \\ \displaystyle\sum_{i=1}^{n} m_i \frac{t_i^{s+1} e^{-\beta t_i} - t_{i-1}^{s+1} e^{-\beta t_{i-1}}}{s\Phi_{\beta t_i t_{i-1}}(s) + \beta^s(t_{i-1}^s e^{-\beta t_{i-1}} - t_i^s e^{-\beta t_i})} + \dfrac{k t_n^{s+1} e^{-\beta t_n}}{s\Gamma_{\beta t_n}(s) - \beta^s t_n^s e^{-\beta t_n}} = 0; \\ \displaystyle\sum_{i=1}^{n} m_i \frac{\Phi_{\beta t_i t_{i-1}}(s) + s\mathrm{F}_{\beta t_i t_{i-1}}(s) + \beta^s\left(t_{i-1}^s e^{-\beta t_{i-1}} \ln(\beta t_{i-1}) - t_i^s e^{-\beta t_i} \ln(\beta t_i)\right)}{s\Phi_{\beta t_i t_{i-1}}(s) + \beta^s(t_{i-1}^s e^{-\beta t_{i-1}} - t_i^s e^{-\beta t_i})} + \\ + \dfrac{k}{s\Gamma_{\beta t_n}(s) - \beta^s t_n^s e^{-\beta t_n}}\left(\beta^s t_n^s e^{-\beta t_n} \ln(\beta t_n) - \Gamma_{\beta t_n}(s) - s\displaystyle\int_0^{\beta t} \tau^{s-1} \ln\tau\, e^{-\tau} d\tau\right) = 0. \end{cases} \tag{2.16}$$

Система трансцендентних рівнянь (2.16) може бути розв'язана відповідним чисельним методом (наприклад, модифікованим методом Ньютона), а отримані наближені значення α, β, s можуть бути використані для підтримки процесу прийняття рішень на етапі тестування ПЗ, аналізу складності та надійності програмного продукту тощо.

2.2. Оцінювання показників надійності ПЗ на основі МНПС

2.2.1. Використання моделей на основі НПП на різних етапах його ЖЦ

Розглянемо можливість застосування МНПС на різних етапах життєвого циклу (ЖЦ) ПЗ. Деякі зроблені вище загальні зауваження, зокрема стосовно важливості та правильної інтерпретації припущень, що лежать в основі моделі, залишаються в силі і при розгляді цього питання. Зокрема, слід зазначити, що точна формалізація припущень необхідна для моделювання навіть якщо процес розробки суттєво відрізняється від моделі. Адже будь-яка модель – це опис складного реального процесу з метою його розуміння і управління, нехтуючи несуттєвими для поставленої задачі властивостями системи. Для практичних потреб модель надійності ПЗ, таким чином, повинна бути достатньо простою і не може описувати в деталях кожну особливість процесу поведінки помилок в системі. Реалізація накладених обмежень в математичній моделі допомагає у виборі моделі, найбільш адекватної заданому етапу ЖЦ ПЗ, або у виборі етапів ЖЦ, на яких ця модель є найбільш адекватною. Далі розглянуто використання МНПС на етапах проектування, модульного тестування, інтеграційного тестування та на етапі експлуатації і визначено етапи ЖЦ, на яких модель (2.1) є найбільш адекватною, з метою розроблення методу використання цієї моделі протягом ЖЦ ПЗ [130].

1. Етап проектування.

На етапі проектування помилки можуть бути виявлені візуально чи за допомогою інших формальних чи неформальних процедур. Найбільш поширені моделі надійності ПЗ непридатні для використання на цьому етапі, оскільки для виявлення помилок потрібні тестові набори (які ще не існують) у випадку

моделей на основі внесення помилок чи на основі області вихідних даних, а у випадку моделей на основі часу між відмовами чи кількості помилок відсутня історія виявлення помилок [26]. Таким чином найбільш поширені моделі надійності ПЗ можуть бути застосовані не раніше етапу тестування ПЗ.

2. Фаза модульного тестування.

Основною особливістю модульного тестування є те, що тестові набори, що генеруються для вхідних даних модуля, не є репрезентативними щодо експлуатаційного використання системи в цілому. Крім того, інтервал часу між виявленими помилками модуля може бути не випадковим, оскільки використана стратегія тестування може не включати випадкового тестування. В реальних проектах тестові набори зазвичай мають детермінований характер.

За таких умов для оцінювання надійності найбільш придатними видаються моделі на основі внесення помилок [26], оскільки можна з високим ступенем достовірності вважати, що вбудовані і уведені помилки мають однакову ймовірність проявитись при тестуванні. Суттєвою перешкодою для використання таких моделей, однак, може бути випадок, коли на цьому етапі програміст є водночас і тестувальником, що особливо характерно для невеликих програмних проектів. Використання моделей на основі області вхідних даних утруднюється тим фактом, що тестовий профіль може не відповідати експлуатаційному профілю використання програми. Завдяки цьому, такі моделі можуть бути використані, однак їх використання може не дати хороших практичних результатів.

Моделі, залежні від часу, особливо моделі на основі часу між відмовами, зазвичай не можна використовувати на цьому етапі, оскільки припущення про незалежність часових інтервалів між відмовами значно порушується.

Однак у випадку оцінювання і прогнозування надійності модуля, а не системи в цілому і за умови ретельного відбору випадкових і однорідних тестових наборів, МНПС може бути використана і на цьому етапі ЖЦ, включаючи використання критерію достатності процесу тестування (див. четвертий розділ цієї дисертації).

3. Фаза інтеграційного тестування.

На етапі інтеграційного тестування усі модулі інтегруються в цілісну систему, а тестові набори слугують для перевірки коректності роботи інтегрованої системи. Тестові набори для цієї задачі можуть генеруватись випадково з множини вхідних даних, або ж створюватись детерміновано, слідуючи певній стратегії тестування, при цьому останнє виглядає більш ефективним і більш широко використовується в реальній практиці розробки ПЗ. Виявлені помилки виправляються, однак є висока ймовірність, що під час виправлення виявлених помилок можуть вноситись нові.

За таких умов тестування теоретично можна застосовувати моделі на основі внесення помилок, оскільки на цьому етапі можна дозволити собі вносити помилки в систему. Так само можуть застосовуватись моделі на основі області вхідних даних із застосування чіткого розподілу тестових наборів [26]. Складність застосування таких моделей полягає в дуже великій кількості логічних варіантів виконання усієї програмної системи.

Якщо використовується детерміністичне тестування (наприклад аналіз граничних значень, тестування шляхів виконання тощо), використання моделей на основі часу між відмовами може бути неприйнятним, оскільки порушується умова незалежності часових інтервалів між відмовами. Моделі на основі кількості відмов можуть застосовуватись, якщо набори тестів є взаємно незалежними, навіть якщо тести множини вибираються детерміновано. Це пов'язано з тим, що основним припущенням такого класу моделей є те, що інтенсивність відмов спадає в результаті виконання набору тестів, а не після кожної помилки.

Якщо здійснюється випадкове тестування згідно з передбачуваним розподілом вхідних даних, тоді можна використовувати більшість існуючих моделей надійності ПЗ. Так, моделі класу „на основі області вхідних даних” повинні використовувати розподіл тестових даних, які статистично еквівалентні розподілу експлуатаційних даних. Так само можна використовувати моделі на основі висівання помилок, оскільки (як вже зазначалось) на цьому етапі ЖЦ

можна вносити контрольні помилки, а припущення про рівну ймовірність виявлення помилок не повинно значно порушуватись завдяки випадковій природі процесу генерування тестів. Моделі на основі часу між відмовами та кількості відмов найбільш придатні у випадку випадкового тестування.

Таким чином можна вважати, що МНПС, завдяки її перевагам над основними відомими моделями цього класу, є найбільш придатною для використання на етапі інтеграційного тестування, при цьому слід звернути особливу увагу на використання взаємно незалежних наборів тестових даних та однорідного охоплення процесом тестування усіх модулів програмної системи. Перевагою МНПС над моделями на основі внесення помилок є відсутність необхідності внесення помилок в код програми (тим більше ці помилки повинні мати рівну ймовірність виявлення з неконтрольованими помилками); перевагою над моделями на основі часу між відмовами є її толерантність (за дотримання описаних вище умов) до детермінованого процесу тестування, який найчастіше використовується при розробці промислових програмних продуктів; а перевагою порівняно з моделями на основі області вхідних даних є менш строга вимога до передбачення експлуатаційного виконання програми, яка може бути замінена вимогою до тестування усіх модулів програми замість усіх логічних шляхів виконання програми.

4. Фаза приймального тестування.

Протягом фази приймального тестування створюються набори вхідних даних на основі експлуатаційного використання продукту для перевірки придатності продукту до уведення в експлуатацію та оцінювання його надійності. У цій фазі внесення помилок є неприйнятним з практичної точки зору, а виявлені помилки не завжди одразу виправляються. Таким чином, моделі на основі внесення помилок та часу між відмовами є непридатними для оцінювання надійності у цій фазі життєвого циклу [26]. Інші міркування подібні до наведених у частині, присвяченій інтеграційному тестуванню, тому моделі на основі кількості відмов та області вхідних даних в загальному є прийнятним у цій фазі.

Таким чином МНПС може використовуватись і на цьому етапі життєвого циклу ПЗ, крім того, на відміну від існуючих моделей, у цій фазі суттєву підтримку в процесі прийняття рішення про уведення програмного продукту в експлуатацію надає використання критерію достатності процесу тестування разом з оцінкою кількості залишкових помилок в програмі та імовірності їх появи чи середнього часу напрацювання на відмову (див. нижче).

5. Етап експлуатації.

Коли якість ПЗ визнається достатньою, програмний продукт вводиться в експлуатацію. Під час експлуатації вхідні дані користувачів можуть не бути випадковими оскільки, наприклад, користувач може використовувати тільки деякі одні й ті ж функції продукту під час використання програми. Вхідні дані також можуть бути взаємопов’язаними (наприклад в системах реального часу). Крім того, помилки зазвичай не виправляються одразу після виявлення. За таких умов видається, що моделі на основі кількості відмов є єдиними придатними для відстеження інтенсивності відмов чи для визначення оптимального часу для встановлення нової версії продукту.

Таким чином, розглянуті особливості використання моделей надійності ПЗ на різних етапах ЖЦ дають змогу зробити висновок, що МНПС має ряд переваг порівняно з існуючими моделями і може бути застосована практично на усіх етапах (крім аналізу вимог та проектування) ЖЦ ПЗ навіть у випадках коли порушуються деякі припущення, зроблені при побудові моделі.

Оскільки МПНС відноситься до класу моделей на основі кількості відмов і розглядає кількість відмов ПЗ на часових інтервалах як неоднорідний пуассонів процес, процеси виправлення помилок та відновлення після збоїв у розгляді цієї моделі нехтуються, а тривалість перебування в стані відновлення вважається нехтовно малою. Тому такі кількісні показники надійності як функція простою, коефіцієнт готовності, коефіцієнт простою, середня тривалість відновлення тощо неможливо обчислити виходячи з параметрів моделі. Для коректного визначення таких показників надійності можна, наприклад, розглядати поведінку ПЗ як марковський процес [11, 37].

2.2.2. Метод оцінювання показників надійності ПЗ на основі МНПС

Розглянемо метод оцінювання і прогнозування показників надійності ПЗ, а також процедуру прийняття рішень при розробці програмного продукту на основі МНПС. Схема цього методу наведена на рис. 2.3 і складається з 7 кроків [130, 136].

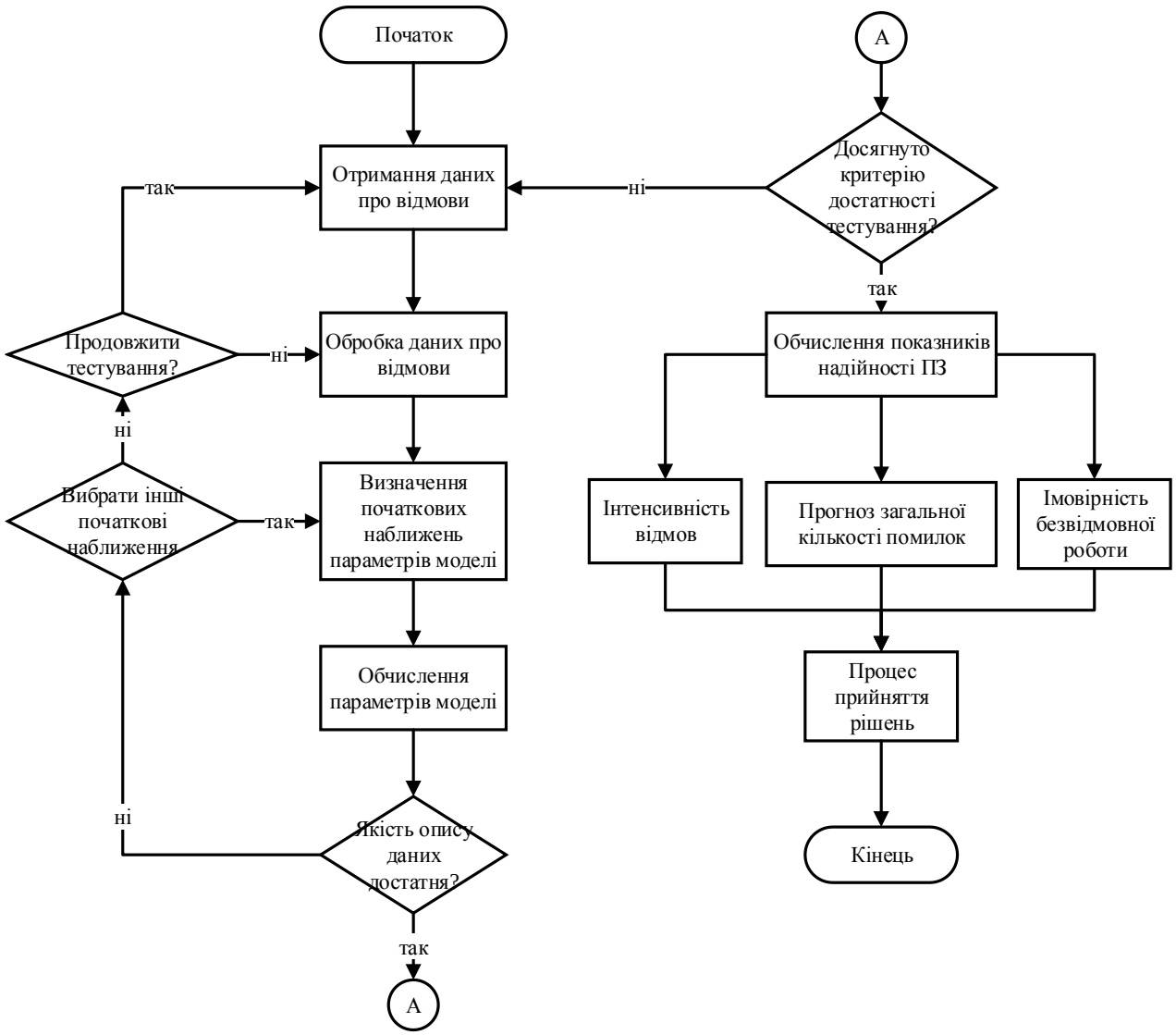


Рис. 2.3. Схема методу оцінювання та прогнозування показників надійності ПЗ на основі МНПС.

Крок 1. Підготовка та отримання даних про помилки ПЗ.

Вихідні дані для моделі отримуються в процесі тестування програмного забезпечення. Для моделі з показником складності використовується кількість

відмов на певних часових інтервалах. Однак перед початком застосування моделі слід забезпечити дотримання вимог та спрощень моделі і ретельно проаналізувати вхідні дані.

Першим етапом перед аналізом даних є представлення їх у вигляді кількості відмов на вибраних часових інтервалах. Для цього діапазон вхідної статистичної вибірки ділять на k інтервалів однакової довжини ($k \leq 5 \lg n$, де n – загальна кількість відмов, отриманих в процесі тестування).

Якщо процес тестування відбувався нерівномірно в часі, чи в процесі була задіяна різна кількість виконавців, вхідні дані слід привести до однорідного, рівномірного масштабу, наприклад до кількості людино-годин, витрачених на процес тестування. В ролі часової шкали також може бути використана кількість тестових наборів чи одиниці процесорного часу. Для аналізу вхідних даних щодо придатності зазначеної моделі надійності доцільно зобразити ці дані на графіку у вигляді кумулятивної кількості відмов як функції вибраного часового масштабу. Основним завданням на цьому етапі є встановлення відповідних змінних, які будуть використовуватись в моделі, зокрема аргументу функції кумулятивної кількості та інтенсивності виявлення відмов. Наприклад, залежно від вхідних даних та інформації про процес розробки, в ролі часової шкали може виявитись більш придатною кількість пройдених тестових наборів ніж, скажімо, календарний час. Інколи доцільним може бути поєднання декількох мір часу чи нормалізація даних, наприклад для урахування змін розміру системи під час тестування.

Крок 2. Визначення початкових наближень параметрів моделі.

Оскільки параметри моделі отримують як розв'язки системи рівнянь (2.16) ітераційним методом, треба встановити початкові наближення для параметрів α, β, s моделі (2.1), (2.2). Зрозуміло, що у кожному конкретному випадку, для кожних вхідних даних початкові наближення будуть різними, однак можна дати деякі загальні рекомендації стосовно вибору початкових наближень.

- В ролі початкового наближення параметру α можна вибрати значення кількості відмов, виявлених від початку процесу тестування;
- Як початкове наближення параметру β можна вибрати обернене значення тривалості процесу тестування у вибраній часовій шкалі;
- При виборі початкового наближення параметру s можна скористатись рекомендаціями щодо співвідношення цього параметру зі складністю ПЗ [133], і в ролі початкового наближення брати значення із середини відповідного інтервалу.

Необхідно зазначити, що початкові наближення для параметрів α, β, s моделі (2.1), (2.2) можливо встановити тільки після проведення тестування хоча б на одному часовому інтервалі.

Крок 3. Обчислення параметрів моделі.

Для визначення параметрів моделі в загальному можна використати різні методи в залежності від природи наявних даних [26]. Тут використовується ММП. Система рівнянь для визначення точкових оцінок моделі (2.16), для розв'язку цієї системи використовується модифікований метод Ньютона з початковими наближеннями, отриманими на кроці 2. Після розв'язку системи рівнянь, обчислені точкові оцінки підставляють в модель і отримують числовий опис моделі на основі наявних даних про поведінку відмов.

Крок 4. Перевірка якості опису експериментальних даних моделлю.

Перед подальшим опрацюванням даних слід визначити, чи підлягає експериментальний розподіл відмов використовуваній моделі та наскільки точно експериментальні дані описуються моделлю з показником складності. Для цього пропонується проведення ряду досліджень, зокрема:

- провести тест Колмогорова–Смірнова, який доцільно застосовувати в тих випадках, коли треба перевірити чи підлягає спостережена випадкова величина певному закону розподілу, відомому з точністю до параметрів. Це один з основних і найбільш широко

використовуваних непараметричних тестів, оскільки він є достатньо чутливим до різниць в досліджуваних вибірках [3];

- другим критерієм є значення коефіцієнту кореляції між експериментальною та теоретичною вибіркою, отриманою з використанням моделі в точках, які відповідають експерименту. Коефіцієнт кореляції використовується як показник характеру взаємного стохастичного впливу зміни двох випадкових величин, тому що проводиться вимірювання змінних з інтервальною шкалою. Оскільки в цих дослідження коефіцієнт кореляції дуже близький до одиниці, що свідчить про функціональний зв'язок (лінійну залежність) експериментальних та теоретичних даних, для кількісної оцінки якості опису вхідних даних моделлю використовуємо значення квадрату цього коефіцієнту;
- останнім критерієм [128], що використовується для оцінки якості опису поведінки помилок в програмній системі моделлю з показником складності є середнє квадратичне відхилення, яке, в свою чергу дозволяє обчислити інтервал довіри та дисперсію.

У випадку успішного проходження згаданих тестів, можна вважати, що експериментальні дані з достатньою точністю описуються МНПС (2.1), (2.2) і можна продовжувати проводити оцінювання та прогнозування показників надійності ПЗ. У іншому випадку слід спробувати змінити початкові значення (крок 2) і отримати нові точкові оцінки моделі (крок 3), або ж характер процесу виявлення відмов (можливо внаслідок специфіки розробки чи процесу тестування) не відповідає цій моделі, і слід вибрати інший варіант змінних моделі (часового аргументу) або ж продовжити процес тестування з урахуванням припущень і обмежень моделі (див. вище). Не існує однозначної відповіді про те, скільки потрібно експериментальних даних тестування ПЗ чи як вибирати часову шкалу чи початкові наближення, крім уже зазначених рекомендацій.

Крок 5. Перевірка критерію достатності процесу тестування.

З метою більш обґрунтованого прийняття рішення про достатність процесу тестування і випуск фінальної версії продукту крім показників надійності, доцільно застосувати критерій достатності процесу тестування [128, 130] (засоби підтримки прийняття рішень при виробництві програмних продуктів, і критерій достатності процесу тестування зокрема, будуть детально описані в четвертому розділі цієї дисертаційної роботи), який базується на використанні в розробленій моделі дійсного неперервного показника s, пов'язаного з величиною і характеристиками ПЗ.

Для перевірки цього критерію слід мати набір значень параметру s в різні моменти часу (не менше трьох точок). Для цього часову шкалу процесу тестування $(0; T]$ ділять на проміжки $(0; t_1], (0; t_2], \dots, (0; T]$ (де $t_1 < t_2 < \dots < T$), кожен з яких включає попередній, і для кожного проміжку незалежно отримують точкові оцінки параметрів моделі. Іншим випадком може бути визначення точкових оцінок параметрів моделі після кожної ітерації процесу тестування ПЗ, якщо такий передбачається проектом розробки системи.

Після цього графічно будується залежність $s(t)$, де в ролі абсцис будуть значення $t_1, t_2, \dots, T$, а відповідними ординатами – точкові оцінки параметра s, отримані з розв'язку системи рівнянь ММП з експериментальними даними з відповідного часового інтервалу. Визначена залежність чисельно диференціюється. Критерієм достатності процесу тестування буде зменшення похідної до нуля. При досягненні цього критерію процес появи відмов відповідає НПП, а модель максимально коректно відповідає покладеним в основу припущенням, і, відповідно, адекватно описує експериментальні дані.

Крок 6. Обчислення кількісних характеристик надійності ПЗ.

На цьому етапі можна обчислити різні кількісні характеристики для оцінювання надійності ПЗ, такі як прогнозована загальна кількість помилок в системі, кількість залишкових помилок в системі, середній час до виявлення наступної помилки, інтенсивність відмов, ймовірність безвідмовної роботи тощо

[3, 44]. Також можна обчислити інтервал довіри для кожної з цих характеристик для оцінки ступеня невизначеності обчислених значень показників надійності.

Крок 7. Процес прийняття рішень.

Загальною метою побудови усіх моделей надійності ПЗ є використання їх результатів для прийняття рішень стосовно програмної системи, наприклад стосовно випуску фінальної версії чи продовження процесу розробки, перерозподілу ресурсів проекту тощо. На цьому етапі приймаються такі рішення на основі інформації, отриманої на попередній кроках описаного методу.

При цьому процедура прийняття рішень обов'язково повинна враховувати критерій достатності процесу тестування як відправну точку для прийняття рішень. Використання цього критерію робить процес прийняття рішень більш обґрунтованим порівняно з існуючими моделями надійності та процедурами прийняття рішень, описаними, наприклад, в [26]. Кількісно мірою процесу прийняття рішень може бути кількість залишкових помилок в системі, середня тривалість до наступної відмови, ймовірність безвідмовної роботи тощо та відповідні інтервали довіри цих величин. Крім того процес виявлення помилок різних типів (критичних, блокуючих, тривіальних тощо) може бути описаний на основі тієї ж МНПС зі своїми числовими значеннями параметрів з використанням того самого критерію достатності процесу тестування по кожному типу помилок. В такому разі процес прийняття рішень може базуватись, наприклад, на залишковій кількості критичних помилок, імовірності безвідмовної роботи стосовно блокуючих помилок тощо.

2.3. Верифікація моделі та методу оцінювання показників надійності ПЗ за результатами статистичних випробувань

2.3.1. Верифікація МНПС на основі експериментальних даних тестування програмних засобів

Верифікація – це спосіб підтвердження за допомогою доказів певних теоретичних положень шляхом їх співставлення з емпіричними даними. Відповідно, для підтвердження коректності розглянутих моделей надійності

програмного забезпечення та методу аналізу надійності ПЗ можна використати емпіричні дані про відмови програмних продуктів, отримані в процесі їх тестування. Для максимально коректної верифікації моделі надійності ПЗ з показником складності та порівняння її з відомими моделями слід використати достатню статистичну вибірку емпіричних даних, отриманих в результаті відтворюваного, контрольованого експерименту. В якості джерела таких даних в цій роботі розглядаються результати, опубліковані в [90], у вигляді серії контрольованих програмних експериментів, результати яких наведено в додатку А (табл. А.1 та А.2 для першого та другого експерименту відповідно).

Експеримент 1. В роботі [90] наведено опис програмних експериментів з метою перевірки належності поведінки зростання надійності ПЗ до неоднорідного пуассонового процесу. Як емпіричні спостереження, так і тестування статистичних гіпотез показали, що поведінка надійності ПЗ не відповідає неоднорідному пуассоновому процесу в загальному, і не описується моделлю Goel–Okumoto зокрема [90].

Для можливості коректного порівняння результатів застосування МНПС з відомими моделями надійності ПЗ було використано дані першого експерименту, описаного в роботі [90].

Програмний засобом, що використовувався для досліджень була програма Space [90], яка складалась з 9564 рядків коду мовою С, з яких 6218 виконуваних. За функціональним призначенням ця програма є інтерпретатором мови визначення масивів ADL. Тестовим пулом (вхідними даними) в дослідженні [90] було 13498 тестових випадків, кожен з яких був файлом ADL. В програму Space було внесено 38 помилок, з яких дві не виявлялись жодним тестовим випадком з тестового пулу вхідних файлів. Після внесення помилок в код програми, вона піддавалась автоматизованому тестуванню засобом SRATE (Software Reliability Analysis, Testing, and Evaluation) [90]. Після кожної відмови, одна і тільки одна помилка, яка спричиняла цю відмову, видалялась з коду програми. Процес тестування зупинявся після виконання 1200-го тесту. Такий процес в [90] називається випробуванням. Кожне випробування включало 1200 тестів

незалежно від того, скільки помилок було виявлено в його ході. В [90] було проведено 40 випробувань (див. табл. А.1), для кожного з яких генерувався різний випадковий тестовий профіль.

Дослідження були проведені так, щоб імітувати використання моделі надійності на етапі тестування проектів з розробки ПЗ:

- етап тестування з 1200 ітерацій розбивали на фази, в даному випадку по 50 ітерацій;
- після кожних 50 ітерацій тестування, методом Ньютона з використанням пакету Mathcad 14.0 розв'язували систему рівнянь (2.16) на інтервалі $(0; t_i]$, і отримували точкові оцінки параметрів $\hat{\alpha}$, $\hat{\beta}$ та $\hat{s}$. Отримані наближені значення підставляли у вираз (2.2) і будували залежність $\mu(t)$ на інтервалі $(t_{i-1}; t_i]$;
- після побудови залежності $\mu(t)$ на усьому проміжку експериментальних даних розраховували коефіцієнт детермінації та середньоквадратичну похибку апроксимації для експериментальних значень і значень, отриманих з МНПС.

Такі дослідження було проведено для моделі (2.2), моделі Goel–Okumoto [62] та S-подібної моделі [63].

Коефіцієнт детермінації (R^2) та середня квадратична похибка між експериментальною та обчисленою кумулятивною кількістю відмов ($\Delta^2 = \sum_i \left(\mu_{exp}(t_i) - \mu_{calc}(t_i)\right)^2 / n$, тут $\mu_{exp}(t_i)$ – експериментальне значення кумулятивної кількості відмов в момент часу t_i , $\mu_{calc}(t_i)$ – розраховане з параметрів відповідної моделі значення кумулятивної кількості відмов в момент часу t_i) наведено в табл. 2.1.

Таблиця 2.1.

Параметри опису експериментальних даних дослідженими моделями.

Модель	$\boldsymbol{R^2}$	$\boldsymbol{\Delta^2}$
Модель Goel–Okumoto	0,993	0,12
S-подібна модель	0,995	0,18
МНПС	0,999	0,05

Дані табл. 2.1 показують, що МПНС суттєво краще описує реальні експериментальні результати: коефіцієнт детермінації для експериментальних даних і МНПС становить 0,999, в той час як для моделі Goel–Okumoto – 0,993, а для S-подібної моделі – 0,995; середнє значення квадрату відхилень між обчисленими та експериментальними значеннями становить 0,05 для моделі (2.2), що майже на порядок краще ніж двох інших моделей – 0,12 для моделі Goel–Okumoto та 0,18 для S-подібної моделі. Крім того, отримані результати показують, що в цілому S-подібна модель краще описує експериментальні дані, ніж модель Goel–Okumoto.

Розрахована за рівнянням (2.3) загальна кількість помилок в ПЗ з використанням отриманих для усього етапу тестування параметрів моделі (2.2) становить 35,1, тоді як в цей програмний продукт було впроваджено 38 помилок. Таким чином МНПС придатна до використання на підприємстві для оцінювання загальної кількості помилок та прогнозування частоти їх появи. Отримана оцінка показника складності ПЗ s для усього етапу тестування становить 0,085, що дозволяє віднести програму до невеликих, що і є насправді – програма складалась з 9564 рядків коду мовою С, з яких 6218 виконуваних [90].

Відмінності в описі поведінки надійності досліджуваного ПЗ цими трьома показано на рис. 2.4, де наведено залежність параметру потоку відмов від часу (крива 1 – МНПС, крива 2 – модель Goel–Okumoto, крива 3 – S-подібна модель).

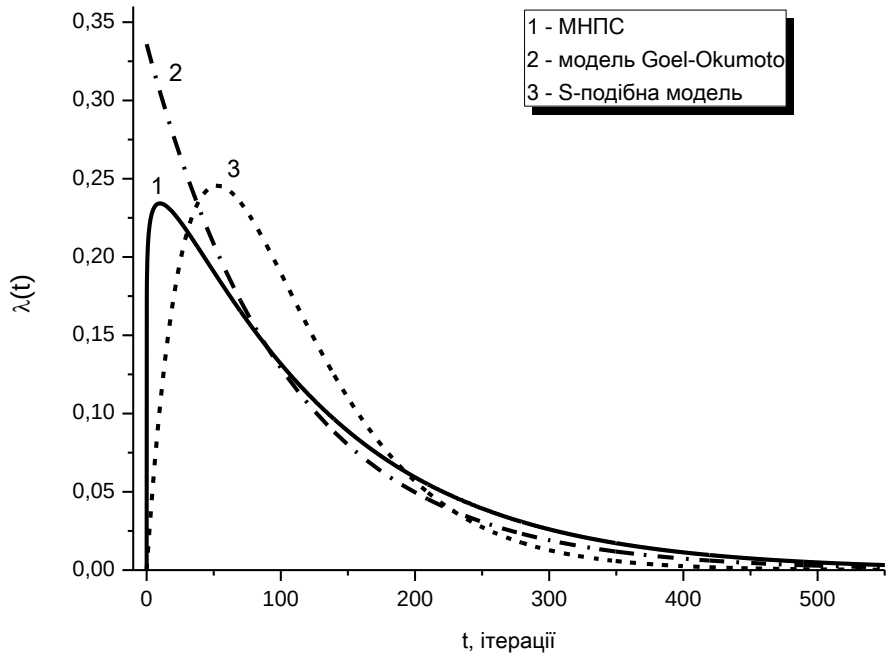


Рис. 2.4. Залежність параметру потоку відмов від кількості ітерацій для різних моделей.

Як видно з рис. 2.4 як модель Goel–Okumoto, так і S-подібна модель показують завищені, порівняно з експериментом, значення параметру потоку відмов при малих значеннях часу, і навпаки – занижені значення при великих t, вказуючи на більш раннє завершення процесу виявлення помилок, ніж спостерігається в реальних експериментальних даних. Крім того, модель Goel–Okumoto неадекватно описує початковий етап тестування (рис. 2.4). Своєю чергою S-подібна модель найгірше описує середину процесу тестування (в даному випадку приблизно від 50 до 175 ітерації).

Важливою характеристикою надійності є частота відмов [3], яка за означенням є густиною розподілу імовірностей випадкової величини ξ – часу до відмови, і визначається як $f(t) = \lambda(t)e^{-\mu(t)}$.

Графік залежності частоти відмов від часу, обчислений з використанням трьох розглянутих моделей надійності наведено на рис. 2.5.

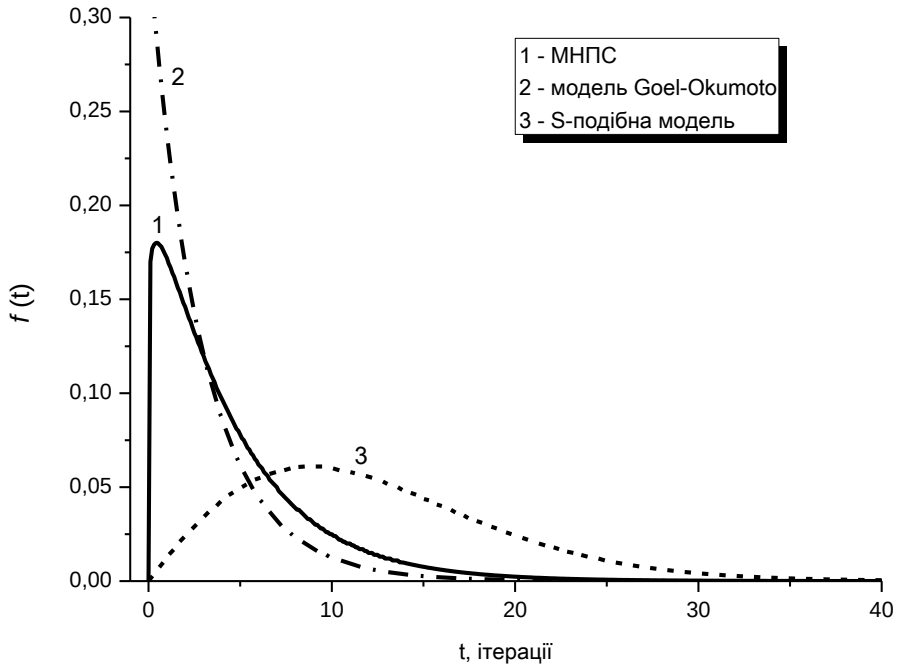


Рис. 2.5. Залежність частоти відмов від кількості ітерацій для різних моделей.

Як і у випадку параметру потоку відмов, залежність частоти відмов (рис. 2.5) дає підстави зробити аналогічні висновки. Модель Goel–Okumoto (крива 2, рис. 2.5) показує, що частота відмов є аномально високою відносно експериментальних даних в початкові моменти часу і аномально швидко затухає з часом, що не відповідає практичним результатам тестування ПЗ. Натомість S-подібна модель (крива 3, рис. 2.5) показує занижену максимальну частоту відмов при більш повільному її зниканні з часом.

Як бачимо з рис. 2.4 та рис. 2.5 МНПС усуває недоліки обох попередніх моделей шляхом уведення динамічного показника складності ПЗ, і, завдяки цьому, більш адекватно описує залежності показників надійності реальних програмних продуктів.

Експеримент 2. Наступний крок досліджень використовував емпіричні дані другого експерименту, описаного в роботі [90], стосовно яких було зроблено висновок, щодо невідповідності їх розподілу пуассоновому, а відповідно моделі, в основі яких лежить такий розподіл, не достатньо коректно описують поведінку цих експериментальних даних.

Як об'єкт дослідження використовувався програмний засіб під назвою SESD (Software Environment for Software Data collection) [90]. Функціональність цього засобу полягала в граматичному аналізі, що використовується в середовищах програмування для збирання програмних даних. Цей програмний засіб містив 3559 рядків коду мовою C++, серед яких 3179 – виконуваного коду. Вхідними даними для засобу SESD є будь-яка програма мовою C. Для такого входу засіб створює п'ять вихідних значень: кількість рядків коду; загальна кількість використання операторів; загальна кількість використання операндів; кількість унікальних використань операторів; кількість унікальних використань операндів. Засіб SESD був реалізований одним програмістом, а потім був предметом незалежного тестування. Під час процесу тестування було виявлено і задокументовано 28 помилок.

Для отримання експериментальних даних виявлені 28 помилки були повторно уведені в програму SESD. Після цього програма пройшла процес автоматизованого випадкового тестування з використанням платформи SRATE, розробленої авторами роботи [90]. Програма SESD без уведених 28 помилок використовувалась як еталонна для оцінювання коректності отриманих результатів. Будь-яке відхилення між результатами виконання тестового випадку (test case) досліджуваною програмою та результатами виконання тестового випадку еталонною програмою розцінювалось як відмова програми. Одна і тільки одна помилка, що спричиняла відмову вилучалась з програми після виникнення відмови. Множина тестів містила 5477 тестових випадків, в якій один тестовий випадок відповідав одній програмі мовою C. Ці програми мовою C були завантажені з мережі Інтернет. Під час тестування було виявлено, що 2 з 28 помилок не виявлялись на даних тестах. Оскільки в реальних умовах індустрії програмного забезпечення процес тестування може бути припинено до виявлення усіх помилок, в цьому експерименті кожен раунд тестування припинявся після виявлення 25 помилок, незалежно від того, скільки часу це займало. Було проведено 40 раундів тестування, результати яких наведено в табл. А.2. Для кожного раунду створювався відповідний профіль тестування. Цей

профіль являв собою певну випадкову послідовність (з однорідним розподілом) вхідних файлів з множини тестів, які подавались на вхід програми SESD [90].

В ролі часової шкали використовувалась кількість ітерацій (тестових випадків), як універсальна міра, що не залежить від зовнішніх факторів та умов тестування. Для використання МНПС експериментальні дані, які представляли у вигляді номерів ітерацій, на яких виникали відмови, були розбиті на інтервали по 10 ітерацій, для яких рахувалась кількість відмов на відповідному інтервалі, після чого будувалась кумулятивна функція відмов. Для досліджень, пов'язаних з визначенням критерію достатності процесу тестування (див. розділ 4), вхідні дані додатково опрацьовувались таким чином:

- раунд тестування розбивали на фази, в даному випадку по 50 ітерацій;
- після кожних 50 ітерацій тестування на інтервалі $(0; t_i]$ отримували точкові оцінки параметрів моделі $\hat{\alpha}$, $\hat{\beta}$ та $\hat{s}$.

Такі дії мали на меті імітувати використання моделі надійності на етапі тестування проектів з розробки програмного забезпечення, де проводиться певна кількість тестувань, після чого кожного разу приймається рішення про продовження чи достатність етапу тестування.

Дослідження проводились як для усереднених за усіма 40 раундами значень кількості помилок, так і для значень кількості помилок, отриманих протягом 10-го, 20-го, 30-го і 40-го раундів (див. табл. А.2) з метою встановлення статистичної однорідності отриманих результатів та точкових оцінок параметрів моделі.

Такі дослідження було проведено для МНПС, моделі Goel–Okumoto та S-подібної моделі.

Першим кроком досліджень був опис кумулятивної функції відмов відповідними функціями моделей надійності ПЗ та отримання точкових оцінок параметрів моделей. Точкові оцінки параметрів моделей отримували ММП. Точкові оцінки параметрів розглянутих моделей надійності та прогнозована

кількість помилок при $t = \infty$ для кожної моделі наведені в табл. 2.2. Як видно з цієї таблиці усі три моделі дають однаковий прогноз загальної кількості помилок в програмі і не дають змоги виявити помилки, які не проявляються при тестуванні. Таким чином ці результати не дають можливості порівняти якість опису експериментальних даних розглянутими моделями надійності ПЗ.

Таблиця 2.2.

Точкові оцінки параметрів моделей.

Модель	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	$\boldsymbol{s}$	**Прогноз помилок**
Модель Goel–Okumoto	25,08	0,016	–	25,08
S-подібна модель	25,08	0,028	–	25,08
МНПС	26,75	0,013	0,137	25,08

Усереднене за 40 раундами тестування значення кумулятивної функції відмов та результати опису цієї функції різними моделями надійності ПЗ зображено на рис. 2.6. На цьому рисунку точками зображено експериментальні дані, крива 1 відповідає кумулятивній функції МНПС (2.2), крива 2 – моделі Goel–Okumoto, крива 3 – S-подібній моделі. Як показано на рис. 2.6 на пізніх етапах тестування усі моделі практично однаково описують експериментальні дані, що узгоджується з результатами роботи [90]. Однак на початкових етапах тестування модель Goel–Okumoto, так само як і S-подібна модель, дещо гірше описують дані тестування реального ПЗ, на відміну від МНПС (крива 1, рис. 2.6).

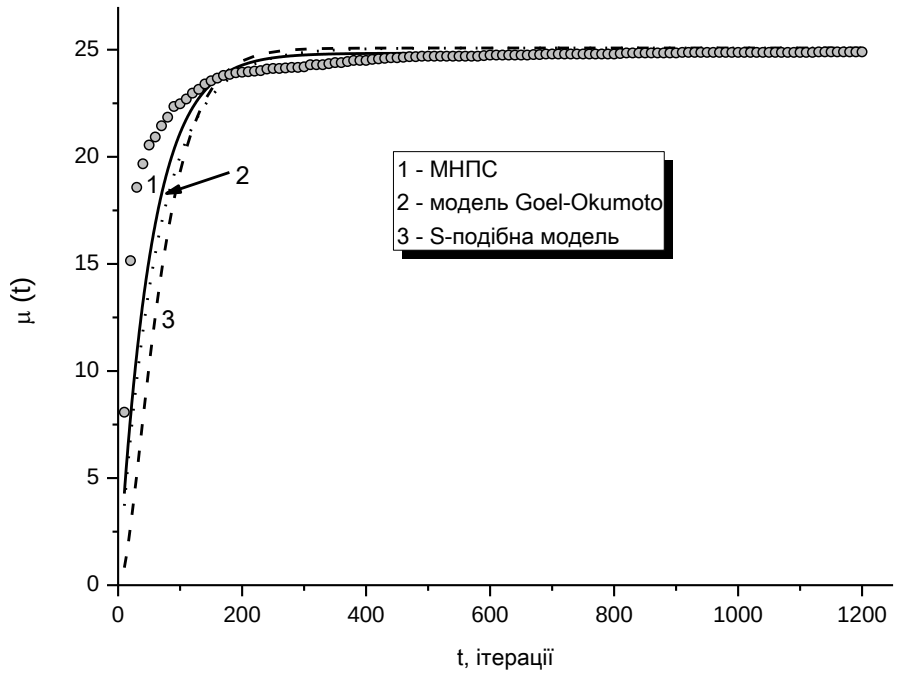


Рис. 2.6. Функція кумулятивної кількості відмов для різних моделей надійності.

Для статистичного оцінювання якості опису експериментальних даних моделями надійності ПЗ було розраховано коефіцієнт детермінації (R^2) та середня квадратична похибка апроксимації для кожної моделі (Δ^2). Значення цих величин наведені в табл. 2.3.

Таблиця 2.3.

Якість опису експериментальних даних дослідженими моделями.

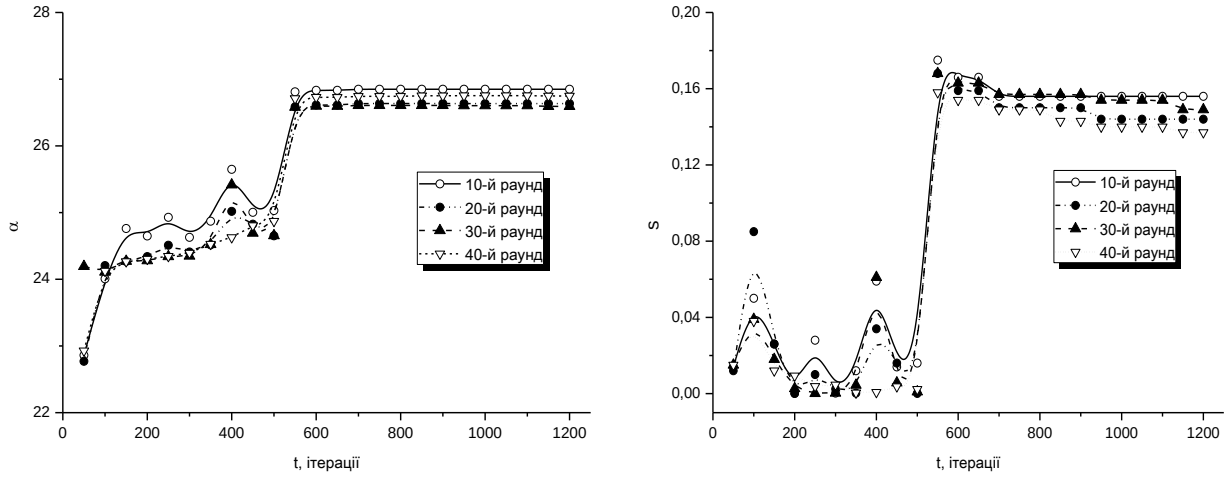
Модель	R^2	Δ^2
Модель Goel–Okumoto	0,974	0,137
S-подібна модель	0,995	0,105
МНПС	0,989	0,080

Як видно з табл. 2.3 найгіршими статистичними показниками якості опису експериментальних даних володіє модель Goel–Okumoto – коефіцієнт детермінації є найменшим (0,974), а середня квадратична похибка найбільша (0,137). МНПС за середньою квадратичною похибкою суттєво (більш ніж на 20%) переважає S-подібну модель (0,080 проти 0,105), в той час як за значенням

коефіцієнта детермінації дещо поступається S-подібній моделі (0,989 проти 0,995). Такі результати пояснюються тим, що коефіцієнт детермінації, на відміну від середньоквадратичного відхилення описує узгодженість моделі з реальними даними з ймовірністю близькою до одиниці, в той час як середня квадратична похибка характеризує апроксимацію моделлю статистичних даних в рівномірній метриці. Таким чином за статистичними характеристиками якості опису експериментальних даних різними моделями можна зробити висновок, що модель з динамічним показником складності продукту більш адекватно описує реальні експериментальні дані порівняно з традиційними S-подібною моделлю та моделлю Goel–Okumoto.

Для опису поведінки МНПС при використанні на різних наборах тестових профілів було проведено дослідження параметрів моделі для різних раундів тестування та їх статистичний опис. На рис. 2.7 зображено залежності параметрів α (а) та s (б) моделі з показником складності від тривалості процесу тестування для різних тестових профілів.

Як видно з цього рисунку, усі залежності для різних раундів тестування є близькими між собою і проявляють однакові якісні залежності з різким стрибком в околі $t = 500$, що підтверджує припущення про ефективність використання цих параметрів для побудови критерію достатності процесу тестування [128] та визначення точки переходу вибірки вхідних даних до пуассонового розподілу [90].



а). *б*).

Рис. 2.7. Залежність параметрів α (*а*) та s (*б*) моделі з показником складності від тривалості процесу тестування для різних тестових профілів.

Для кількісної характеристики однорідності розподілу точкових оцінок параметрів моделі з показником складності проекту розраховано статистичні характеристики такого розподілу (табл. 2.4). Основною характеристикою в цьому випадку є структурна характеристика вибірки – медіана. Вона практично визначає структуру вибіркових даних і визначається через ці дані. Цей показник має статус основного або головного при асиметричному розподілі даних, причому, у випадку асиметрії розподілу медіана бере на себе роль середнього значення, крім того, медіана вважається найбільш стійкою характеристикою вибірки, а тому може бути основою для критерію оптимального розподілу даних в інтервалах. Як бачимо, в нашому випадку значення медіани вибірки кожного параметра практично співпадає з середнім значенням точкової оцінки кожного параметра, що підтверджує статистичну однорідність даних. Крім того розмах між мінімальним та максимальним значенням точкової оцінки кожного параметра не перевищує 15 % (а у випадку параметру α – 1 %) з дуже низькими значеннями дисперсії та стандартного відхилення, що також свідчить про однорідність вибірки.

Таблиця 2.4.

Статистичні характеристики розподілу точкових оцінок параметрів моделі для різних вхідних наборів тестових даних.

Характеристика	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	$\boldsymbol{s}$	$\boldsymbol{\alpha s\Gamma(s)}$
Найбільше значення	26,849	0,015	0,156	25,08
Найменше значення	26,592	0,013	0,137	24,82
Розмах	0,257	0,002	0,019	0,26
Медіана	26,691	0,014	0,147	24,95
Середнє арифметичне значення	26,706	0,014	0,147	24,95
Дисперсія	0,014	$6{,}7\times10^{-7}$	$6{,}4\times10^{-5}$	0,013
Стандартне відхилення	0,12	$8{,}2\times10^{-4}$	$8{,}0\times10^{-3}$	0,11

Оскільки, середнє вибірки є незміщеною ефективною і правдивою оцінкою для математичного сподівання випадкової величини та враховуючи близькість значень медіани і середнього арифметичного, можна зробити висновок, що дані є статистично однорідні. Це дозволяє використовувати МНПС для прогнозування помилок різного типу.

2.3.2. Верифікація методу аналізу надійності ПЗ на основі даних тестування промислового програмного засобу

Для ілюстрації використання описаного в підрозділі 2.2.2 методу та з метою порівняння моделей надійності ПЗ використано експериментальні дані, які наведені в роботі [26]. Програмним продуктом є система управління реального часу, розроблена Bell Laboratories. Дані тестування є результатом виявлених помилок під час тестування системи протягом 25 годин процесорного часу. В роботі [26] для опису цих даних застосована модель негомогенного пуассонового процесу Goel–Okumoto [62] завдяки її простоті та широкому використанню при дослідженні надійності ПЗ. Нижче наведено застосування методу оцінювання та прогнозування надійності для цих двох моделей на експериментальних даних з [26].

Крок 1: Початкові дані представлені у вигляді часу між відмовами. Оскільки, як модель з динамічним показником складності, так і модель Goel–Okumoto [62] передбачають роботу з даними у вигляді кількості відмов на часових інтервалах, а також з метою усунення можливих залежностей між цими величинами, дані були представлені у вигляді кількості відмов за годину процесорного часу [62]. Ці початкові експериментальні дані наведені в табл. 2.5.

Таблиця 2.5.

Вхідні дані тестування програмного продукту.

Час, год.	**Кількість відмов за годину**	**Кумулятивна кількість відмов**
1	2	3
1	27	27
2	16	43
3	11	54
4	10	64
5	11	75
6	7	82
7	2	84
8	5	89
9	3	92
10	1	93
11	4	97
12	7	104
13	2	106
14	5	111
15	5	116
16	6	122
17	0	122
18	5	127

1	2	3
19	1	128
20	1	129
21	2	131
22	1	132
23	2	134
24	1	135
25	1	136

Зауважимо, що згідно з критерієм, застосованим при описі методу кількість інтервалів доцільно обрати не більше 10, оскільки максимальна виявлена кількість помилок становить 136, однак для максимально коректного порівняння отриманих результатів з результатами, отриманими в [26], надалі наведено результати використання методу аналізу надійності для 25 інтервалів, як наведено в [26].

З табл. 2.5 видно, що процес тестування ще не можна вважати завершеним, оскільки навіть на 25-й годині тестування було виявлено помилку, і функція кумулятивної кількості помилок ще не виходить на насичення, що може впливати на результати оцінювання та прогнозування надійності ПЗ згаданими моделями. Графічна залежність кумулятивної функції помилок від часу показана у вигляді точок на рис. 2.8.

Крок 2: Згідно з описаними рекомендаціями встановимо в ролі початкових наближень такі значення: $\alpha_0 = 150$, $\beta_0 = 0{,}04$, $s_0 = 0{,}35$ оскільки ми не маємо докладних відомостей про програмний продукт. Попередні оцінки даних тестування дають змогу припустити, що цей продукт можна віднести до категорії невеликих через специфіку процесу виявлення помилок, а саме – більшість помилок виявляється вже на перших етапах тестування, тобто параметр потоку відмов практично строго спадає з часом, в той час як для великих і дуже великих проектів характерною рисою є початкова невелика інтенсивність відмов, яка поступово досягає свого максимуму, а потім строго спадає.

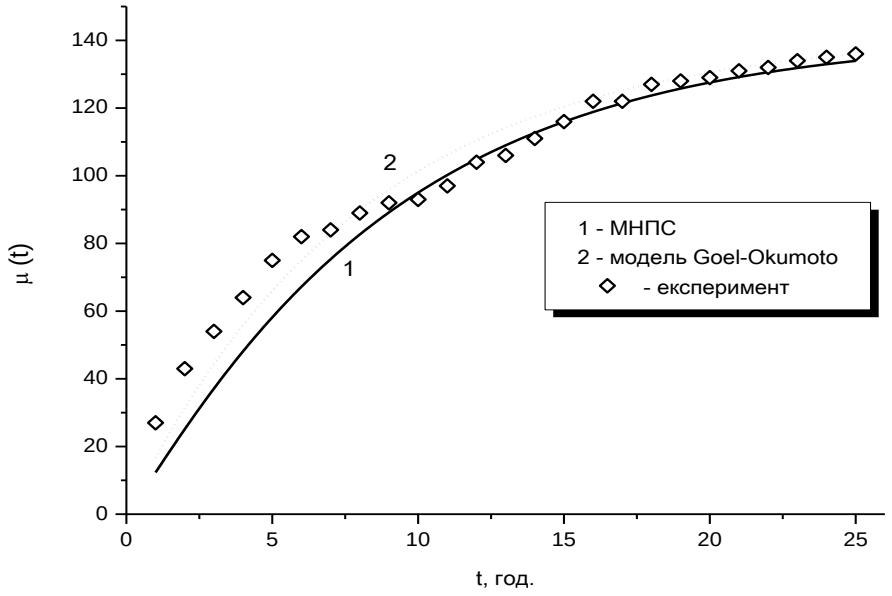


Рис. 2.8. Кумулятивна кількість відмов в залежності від часу тестування для МНПС (крива 1) та моделі Goel–Okumoto (крива 2). Точки – експериментальні значення (табл. 2.5).

Крок 3: З використанням ММП знаходимо розв'язок системи рівнянь для отримання точкових оцінок параметрів МНПС. Отримані після розв'язування системи точкові значення параметрів становлять $\hat{\alpha} = 150{,}0$, $\hat{\beta} = 0{,}126$, $\hat{s} = 0{,}120$. Зауважимо, що значення індексу складності продукту (параметр s) підтверджує припущення про невелику складність ПЗ. Прогнозована загальна кількість помилок в програмній системі становить $\hat{\alpha}\hat{s}\Gamma(\hat{s}) = 142$, що опосередковано підтверджує припущення про недостатність процесу тестування.

У роботі [90] отримані такі значення параметрів моделі Goel–Okumoto для опису цих експериментальних даних: $\hat{a} = 142{,}32$, $\hat{b} = 0{,}1246$. В цій моделі прогнозована загальна кількість помилок дорівнює параметру $\hat{a}$ і становить 142,32, що узгоджується з результатами моделі з динамічним показником складності проекту. Близькість значень параметрів $\hat{\beta}$ моделі [66] та $\hat{b}$ моделі Goel–Okumoto пояснюється їх спільним фізичним змістом: кількість відмов виявлених на помилку за годину [26].

Графіки функцій кумулятивної кількості помилок для моделі (2.1), (2.2), моделі Goel–Okumoto з отриманими значеннями параметрів та експериментальні точки наведені на рис. 2.8 (крива 1, крива 2 та точки відповідно).

Крок 4: Тест Колмогорова–Смірнова дає характеристику результату статистичного порівняння між експериментальним даними та моделлю, отриманою на кроці 3 і використовується для перевірки адекватності моделі. Обидві моделі успішно проходять такий тест, що підтверджує, що вони адекватно описують експериментальні дані з табл. 2.5 статті [26]. З рис. 2.8 також можна помітити, адекватність обох моделей для опису кумулятивної кількості помилок в системі. Значення коефіцієнту детермінації (R^2) та середньої квадратичної похибки (Δ^2) наведено в табл. 2.6. Як видно з табл. 2.6 середня квадратична похибка є майже вдвічі меншою у випадку використання моделі Goel–Okumoto, однак коефіцієнт детермінації є помітно ближчим до одиниці у випадку використання МНПС, що дає можливість зробити висновок про більш адекватний імовірнісний опис експериментальних даних моделлю з показником складності, зауваживши також коректність моделі Goel–Okumoto та можливість її використання для опису цього процесу.

Крок 5: Для перевірки достатності процесу тестування точкові оцінки параметрів моделі були обчислені після 5, 10, 15, 20, 21, 22, 23 та 24 годин тестування. Графік, побудований в результаті чисельного диференціювання визначеної залежності параметру $\hat{s}$ від часу припинення процесу тестування наведено на рис. 2.9.

Таблиця 2.6.

Параметри опису експериментальних даних дослідженими моделями.

Модель	$\boldsymbol{R^2}$	$\boldsymbol{\Delta^2}$
Модель Goel–Okumoto	0,982	35,5
МНПС	0,985	70,9

Як видно з рис. 2.9 залежність похідної параметра $\hat{s}$ по часу далека від досягнення критерію достатності, згідно з яким ця залежність повинна прямувати до нуля [128]. Таким чином цей факт можна вважати прямим підтвердженням припущення про те, що процес тестування досліджуваного програмного продукту ще не завершений, і рішення, прийняті на цьому етапі можуть містити істотну помилку. Можна зауважити, що інші моделі не дають такого критерію, який є важливим засобом підтримки в процесі прийняття рішень стосовно етапу тестування програмних продуктів.

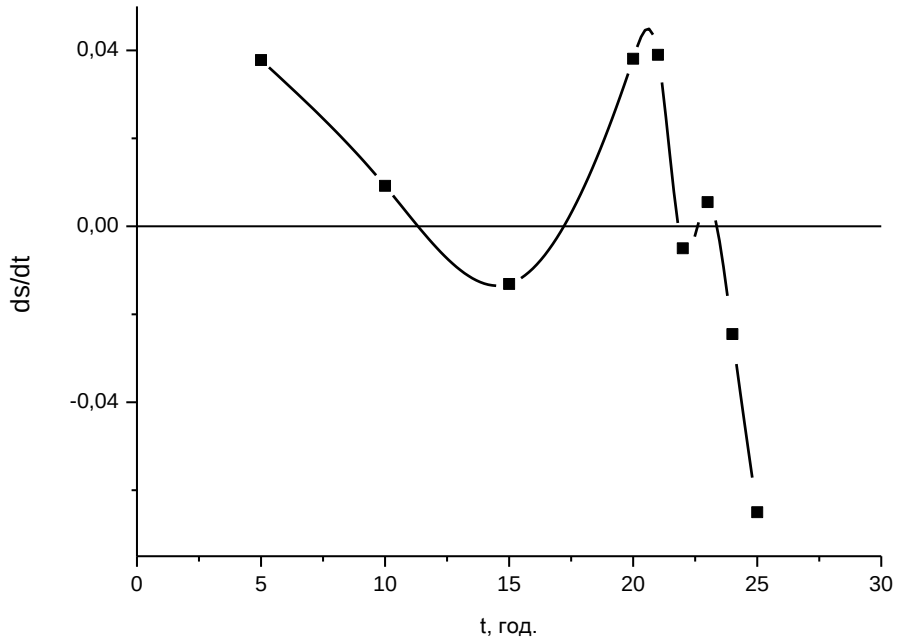


Рис. 2.9. Залежність критерію достатності процесу тестування від часу припинення тестування.

Крок 6: На цьому кроці, можна розрахувати показники надійності даного програмного засобу користуючись виразами, наведеними в [3, 129].

Крок 7: Розроблений метод може бути використаний для опису поведінки процесу появи відмов, а також для визначення додаткових ресурсів для тестування та моменту готовності програмної системи до вводу в експлуатацію. Інформація цього плану доступна на різних часових проміжках і не обов'язково очікувати завершення процесу тестування. Для прикладу в [26] розглядається

випадок, якщо б процес тестування був припинений після 16 годин. В такому випадку було б виявлено тільки 122 помилки (див. табл. 2.5), а точкові оцінки параметрів моделі Goel–Okumoto мали б такі значення $\hat{a} = 138{,}37$, $\hat{b} = 0{,}133$ [26]. Отже, можна зробити висновок, що прогнозована залишкова кількість помилок становить 16,37.

Використання МНПС в цій же ситуації дає наступні результати. Точкові оцінки параметрів моделі становлять $\hat{\alpha} = 158{,}8$, $\hat{\beta} = 0{,}114$, $\hat{s} = 0{,}108$; прогнозована загальна кількість помилок в програмній системі – $\hat{\alpha}\hat{s}\Gamma(\hat{s}) = 150{,}5$. Таким чином прогнозована залишкова кількість помилок становить 28,5.

Як бачимо, завдяки завищеній прогнозованій кількості залишкових помилок МНПС може ввести в оману стосовно необхідних ресурсів для тестування програмного продукту, однак критерій достатності процесу тестування є чітким індикатором того, що процес тестування ще далекий від свого завершення і, можливо, процес появи відмов у цій фазі носить не пуассонів характер.

На основі цих даних можна робити певні висновки стосовно програмного продукту, з урахуванням вимог до його надійності. Так, наприклад, якщо критерієм уведення програми в експлуатацію є наявність менше певної кількості прогнозованих залишкових помилок чи середня тривалість безвідмовної роботи більший певного значення, на основі даних моделі можна отримати кількісне обґрунтування прийнятого рішення. На практиці визначення часу випуску продукту в експлуатацію, перерозподілу ресурсів, проведення додаткового тестування і т.д. базується на значно складніших міркуваннях, ніж кількість залишкових помилок. Результати використання моделей надійності дають вхідну інформацію для процесу прийняття рішень стосовно програмних проектів [130], а МНПС дає нову кількісну характеристику для підтримки процесу прийняття рішень при виробництві програмних продуктів – критерій достатності процесу тестування [128].

2.4. Вплив моделей надійності ПЗ на результати аналізу показників надійності ПАС

На сьогоднішній день розробка програмно-апаратних систем (ПАС), що входять до складу комплексів озброєння, космічних, авіаційних, енергетичних та інших комплексів критичного призначення є важливим напрямком наукових досліджень. Однією із задач цього напрямку є забезпечення вимог, що до надійності (безвідмовності та готовності) і функціональної безпеки. При цьому можливі ризики двох типів, з одного боку пов’язаних з правильністю оцінювання показників надійності та безпечності експлуатації комплексів критичного призначення, а з іншого боку з правильністю вибору засобів забезпечення їх надійності та безпеки.

Оскільки результати оцінювання показників надійності ПЗ впливають на правильність оцінювання показників надійності ПАС, було проведено дослідження впливу моделі надійності ПЗ, на основі якої отримували значення показників надійності ПЗ, на результати оцінювання показників надійності ПАС, зокрема функції та коефіцієнту готовності [134, 135].

Надійнісне проектування відмовостійких ПАС [11, 139] має свою специфіку, оскільки властивість відмовостійкості необхідно реалізувати як в ПЗ, так і в АЗ. Для забезпечення відмовостійкості ПЗ використовується дві і більше його версій від різних розробників [140]. А тому для відмовостійких ПАС з кількома версіями ПЗ обов'язковим є використання структурного резервування. При здачі в експлуатацію ПЗ в його коді залишаються помилки, які призводять до втрати працездатності ПАС. Виправлення таких помилок проводиться шляхом оновлення версій ПЗ. В процесі експлуатації відмовостійких ПАС присутні збої АЗ, що призводять до збоїв в роботі ПЗ ("зависання" роботи ПАС). Для відновлення працездатності ПЗ використовується процедура автоматичного перезавантаження [4]. Працездатність відмовостійкої ПАС в апаратній частині забезпечується технічним обслуговуванням та ремонтом її апаратних засобів [4].

2.4.1. Структура та модель надійності програмно-апаратного засобу з версійно-структурним резервуванням

Розроблення засобів оцінки показників надійності ПАС полягає у побудові моделі ПАС, яка б враховувала його надійнісну та функціональну поведінку [11]. Функціональну поведінку можна подати блок-схемою алгоритму поведінки, який відображає процес виконання цільової функції ПАС.

Модель ПАС, яка дозволить визначити її показники надійності, можна отримати кількома способами, застосовуючи відомі підходи:

- за допомогою логіко-ймовірнісного моделювання [141, 142];
- за допомогою імітаційного моделювання [143];
- у вигляді мережі Петрі [144];
- у вигляді системи диференційних рівнянь, використовуючи як модель дискретно-неперервну стохастичну систему марковського типу [11].

Для побудови надійнісної моделі ПАС було використано технологію моделювання складних інформаційних систем на основі марковських процесів [11], в якій ПАС представляється як дискретно-неперервна стохастична система, оскільки саме вона дозволяє виконати поставлені завдання визначення та дослідження показників надійності ПАС.

Виходячи з того, що відмовостійка ПАС повинна працювати безперервно без втрати працездатності при відмовах процесорів та збоях ПЗ, та без зниження ефективності виконання заданої функції, використано загальне заміщувальне гаряче резервування [4]. Структурна схема надійності відмовостійкої ПАС показана на рис. 2.10 [4, 145].

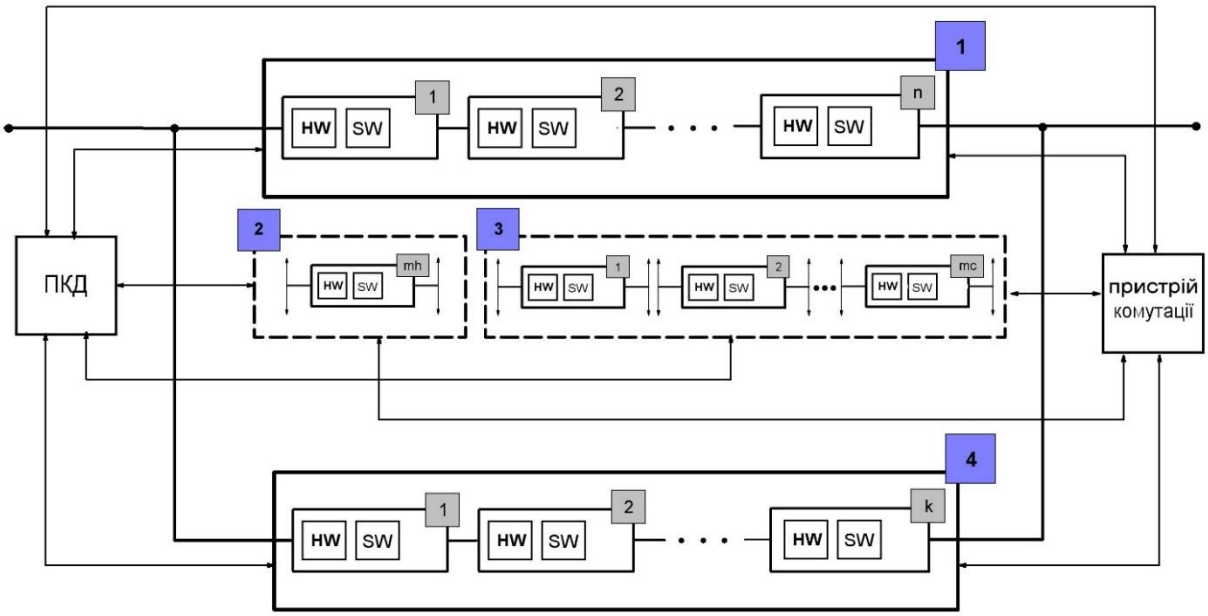


Рис. 2.10. Структурна схема надійності ПАС (1 – основна ПАС, 2 – процесори, що знаходяться в гарячому резерві, 3 – процесори, що знаходяться в холодному резерві, 4 – резервна ПАСЗ, ПКД – процесор контролю та діагностування) [4].

До складу відмовостійкої ПАС входять: основна ПАС, яка складається з n процесорів; резервна ПАС, яка складається з k процесорів; для обох ПАС передбачено спільне ковзне резервування процесорів, при цьому один (перший в черзі на використання) резервний процесор перебуває в гарячому резерві, а решта – в холодному; процесор контролю та діагностування визначає стан ПАС в апаратній та програмній частинах та подає команди для управління резервним ресурсом; пристрій комутації виконує функції підключення (або відключення) резервної ПАС, а також відключення несправних процесорів та підключення процесорів ковзного резерву [4, 145–147].

Розглядається ПАС, для якої при оцінці показників надійності необхідно враховувати: відмови ПЗ; заміну ПЗ оновленими версіями (оновлення версії ПЗ розпочинається з моменту введення системи в експлуатацію); затрати часу на автоматичне перезавантаження ПЗ; збої ПЗ викликані збоями АЗ; відмови АЗ. На рис. 2.11 представлено модель поведінки такої ПАС. В ПАС присутні наступні стани: S1, S4 та S7 – стани, в яких ПАС є працездатною; S2, S5 – стани простою

в яких АЗ є справним, проводиться заміна версії ПЗ оновленою версією; S3, S6, S8 – стани простою, відбувся збій ПЗ, та запущено процедуру автоматичного перезавантаження ПЗ; S9, S10 – стани простою з причини відмови ПЗ, запускається процедура оновлення ПЗ; S11 – стан простою, в якому ПАС несправна з причини відмови АЗ, запускається процедура відновлення АЗ.

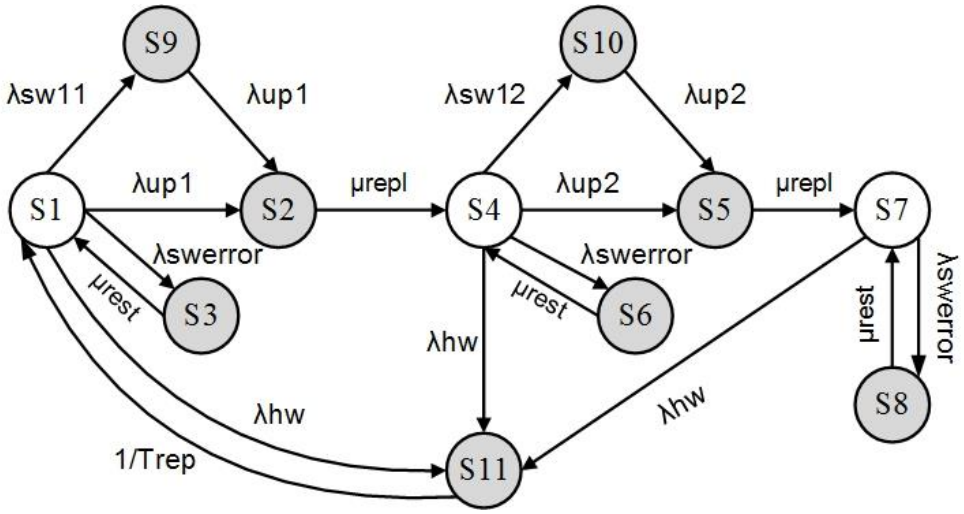


Рис. 2.11. Граф станів та переходів, яким описується надійнісна поведінка ПАС з двократним оновленням ПЗ та автоматичним перезавантаженням ПЗ після збоїв в його роботі [145].

В стані S1 можуть відбутися такі події [4, 145]: відмова ПЗ з інтенсивністю λ_{sw11}, збій ПЗ з інтенсивністю $\lambda_{swerror}$, закінчення експлуатації першої версії ПЗ (оскільки готова до експлуатації друга (оновлена) версія ПЗ) з інтенсивністю $\lambda_{up1} = 1/T_{up1}$, де T_{up1} – середнє значення тривалості розробки оновленої версії ПЗ (тривалість експлуатації першої версії ПЗ), а також відмова АЗ з інтенсивністю λ_{hw}. В стані S4 можуть відбутися такі події [4, 145]: відмова ПЗ (після першого оновлення) з інтенсивністю λ_{sw12}, збій ПЗ викликаний збоями АЗ з інтенсивністю $\lambda_{swerror}$, закінчення експлуатації версії ПЗ після першого оновлення (оскільки готова до експлуатації версія ПЗ після другого її оновлення) з інтенсивністю $\lambda_{up2} = 1/T_{up2}$ відповідно, де T_{up2} – середнє значення тривалості другого оновлення версії ПЗ (тривалість експлуатації версії ПЗ після

першого його оновлення), а також відмова АЗ з інтенсивністю λ_{hw}. В стані S7 може відбутися відмова оновленого другий раз ПЗ з інтенсивністю λ_{sw13}, збій ПЗ викликаний збоями АЗ з інтенсивністю $\lambda_{swerror}$ та відмова АЗ з інтенсивністю λ_{hw} [4, 145].

З працездатних станів S1, S4, S7 при збоях ПЗ, яке викликане збоями АЗ, ПАС попадає в стани S3, S6, S8, відповідно. Закінчення перезавантаження ПЗ відбувається з інтенсивністю $\mu_{rest} = 1/T_{rest}$, де T_{rest} – середнє значення тривалості перезавантаження ПЗ [4, 145]. Коли ПАС перебуває в станах S2 та S5, то відбувається процедура вилучення з ПАС попередньої версії ПЗ і завантаження оновленої версії ПЗ з інтенсивністю $\mu_{repl} = 1/T_{repl}$, де T_{repl} – середнє значення тривалості заміни версії ПЗ в ПАС [4, 145]. Зі станів S9, S10 ПАС попадає в стани S2, S5 з інтенсивностями $\lambda_{up1} = 1/T_{up1}$, де T_{up1} – середнє значення тривалості першого оновлення ПЗ та $\lambda_{up2} = 1/T_{up2}$, де T_{up2} – середнє значення тривалості розробки третьої версії ПЗ відповідно [4, 145].

Після введення в експлуатацію ПАС проводиться перше оновлення версії ПЗ за час T_{up1}, в якій мають бути виправлені помилки при здачі ПЗ в експлуатацію. За час T_{up2} проводиться друге оновлення версії ПЗ. Оскільки після першого оновлення версії ПЗ зменшується кількість помилок, то для їх виявлення підчас другого оновлення необхідні більші затрати часу, відповідно $T_{up2} > T_{up1}$ [4, 145].

Метод розробки надійнісної моделі досліджуваної структури ПАС у вигляді графа станів та переходів, описаний у монографії [11], який передбачає формалізоване представлення об'єкта дослідження у вигляді моделі ПАС для автоматизованої побудови графу-станів та переходів. Для розробки моделі ПАС у [4, 145–147] було виконано наступні завдання: сформовано вербальний опис об’єкту дослідження; визначено базові події; визначено компоненти вектору стану, якими можна описати стан системи в довільний момент часу; сформовано множину параметрів, якими можна описати досліджувану структуру; сформовано дерево правил модифікації компонентів вектору стану.

Параметрами розробленої у [4, 145–147] надійнісної моделі ПАС є: n – кількість модулів, що перебувають в складі основної ПАС; k – кількість модулів, що перебувають в складі резервної ПАС; m_h – кількість модулів гарячого резерву з завантаженим ПЗ; m_c – кількість модулів холодного резерву з не завантаженим ПЗ; λ_{hw} – інтенсивність відмов модуля, який перебуває в складі основної (резервної) ПАС та в гарячому резерві; λ_{sw11}, λ_{sw12} – інтенсивність відмов ПЗ початкової версії та після першого оновлення; $\lambda_{swerror}$ – інтенсивність збоїв ПЗ; T_{up1}, T_{up2} – середнє значення тривалості першого та другого оновлень версії ПЗ; T_{rest} – тривалість перезавантаження ПЗ на модулі, що перебуває в непрацездатному стані з ознакою збою/відмови ПЗ; T_{switch} – тривалість оновлення ПЗ; T_{rep} – тривалість ремонту АЗ в ПАС.

Розроблена модель дає можливість згідно технології [11] автоматизовано побудувати графи станів та переходів. Граф станів та переходів, в якому враховані наступні параметри відмовостійкої ПАС: $n = 2$; $k = 2$; $m_h = 0$; $m_c = 0$; λ_{hw}; λ_{sw11}; λ_{sw12}; $\lambda_{swerror}$; T_{up1}; T_{up2}; T_{rest}; T_{switch}; T_{rep}, представлено на рис. 2.12 [135, 145].

На основі отриманого графа станів та переходів (рис. 2.12), можна скомпонувати формули для надійнісного проектування [11]. Зокрема запропонований граф дозволяє обчислити: функції готовності при зміні параметрів досліджуваної системи; середнє значення тривалості до відмови; провести оцінку безпечності тощо. На основі побудованого графа станів та переходів сформуємо систему диференційних рівнянь Колмогорова–Чепмена (2.17), розв’язком якої є імовірності перебування ПАС в станах, зображених на графі (рис. 2.12), відповідна комбінація яких дає змогу оцінити значення показників надійності ПАС.

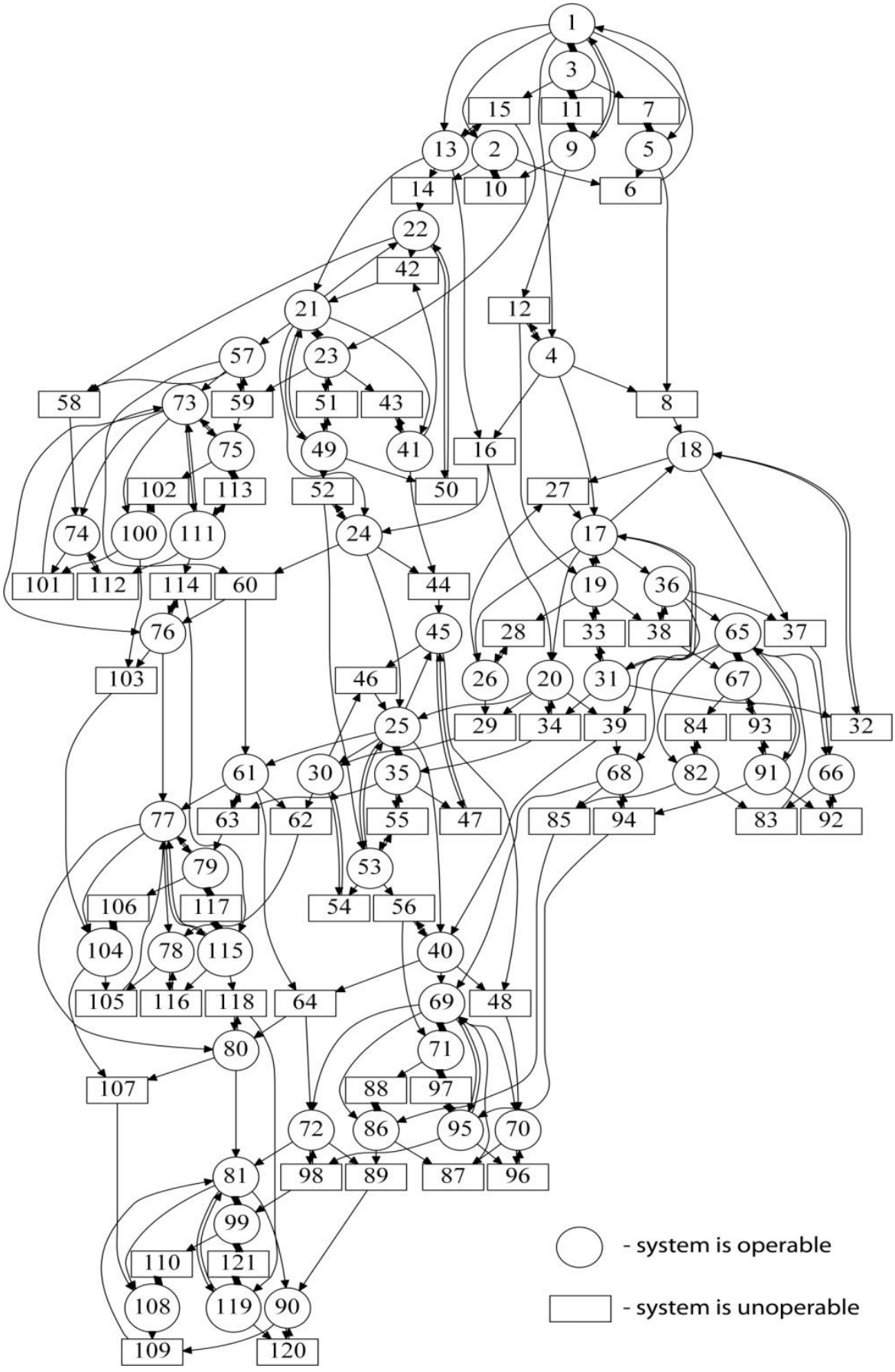


Рис. 2.12. Граф станів та переходів досліджуваної ПАС [135, 145].

$$\begin{cases} \dfrac{dP_1(t)}{dt} = -2\lambda_{hw}\big(P_2(t)+P_5(t)\big) - 2\lambda_{sw11}\big(P_4(t)+P_{13}(t)\big) - 2\lambda_{swerror}P_3(t) - \\ \quad -2\lambda_{swerror}P_3(t) - 2\lambda_{swerror}P_9(t) + \dfrac{1}{T_{rep}}P_6(t) + \dfrac{1}{T_{rep}}\big(P_3(t)+P_3(t)\big); \\ \dfrac{dP_2(t)}{dt} = 2\lambda_{hw}P_1(t) - \dfrac{1}{T_{rest}}P_{10}(t) - 2\lambda_{hw}P_6(t) - 2\lambda_{swerror}P_{10}(t) - 2\lambda_{sw11}P_{14}(t); \\ \qquad\qquad \vdots \\ \dfrac{dP_{120}(t)}{dt} = 2\lambda_{swerror}\big(P_{99}(t)+P_{119}(t)\big) - \dfrac{1}{T_{rest}}\big(P_{99}(t)+P_{119}(t)\big); \\ \dfrac{dP_{121}(t)}{dt} = -\dfrac{1}{T_{rest}}P_{90}(t) + 2\lambda_{swerror}P_{90}(t) - 2\lambda_{hw}P_{119}(t). \end{cases} \quad (2.17)$$

Система лінійних диференційних рівнянь (2.17) розв'язується чисельним методом Рунге–Кутта. Розв'язання даної системи рівнянь дає можливість провести оцінку показників надійності відмовостійкої ПАС, в тому числі визначити функцію та коефіцієнт готовності. Коефіцієнт готовності ПАС розраховується як сума ймовірностей перебування у станах, де відсутні критичні відмови. Таким чином, для розрахунку коефіцієнта готовності досліджуваної ПАС необхідно вибрати стани, в яких система є працездатною. Виходячи з цієї умови функція готовності розглянутої ПАС з версійно-структурним резервуванням записується як:

$$\begin{aligned} К_Г(t) = &\sum_{i=1}^{5} P_i(t) + P_9(t) + P_{13}(t) + \sum_{i=17}^{26} P_i(t) + \sum_{i=30}^{31} P_i(t) + \sum_{i=33}^{36} P_i(t) + \sum_{i=40}^{41} P_i(t) + \\ &+P_{45}(t) + P_{49}(t) + P_{53}(t) + P_{57}(t) + P_{61}(t) + \sum_{i=65}^{82} P_i(t) + P_{86}(t) + \sum_{i=90}^{91} P_i(t) + \\ &+P_{95}(t) + P_{97}(t) + \sum_{i=99}^{100} P_i(t) + P_{104}(t) + P_{108}(t) + P_{111}(t) + P_{115}(t) + P_{119}(t). \end{aligned} \quad (2.18)$$

Коефіцієнт готовності ПАС, за визначенням [3] обчислюється як $К_Г = \lim\limits_{t\to\infty} К_Г(t)$.

2.4.2. Результати розрахунку функції готовності ПАС з використанням різних моделей надійності ПЗ

В [4] було проведено порівняння показників надійності відмовостійкого програмно-апаратного засобу радіотехнічної системи, визначених з урахуванням показників надійності ПЗ та без нього. Встановлено [4], що врахування

показників надійності ПЗ дає різницю між середніми значеннями тривалості безвідмовної роботи ПАС 11,2 % за умови, що оновлення версії ПЗ буде здійснено протягом 1 години. Враховуючи, що оновлення версії ПЗ може зайняти значно більше часу (до 1 місяця), в [4] також було виконано розрахунок для такого випадку середнього значення тривалості безвідмовної роботи відмовостійкої ПАС. Для такого випадку було встановлено, що нехтування тривалістю оновлень версії ПЗ, призводить до завищення середнього значення тривалості безвідмовної роботи на 74,6%.

В [148, 149] було досліджено вплив оновлення ПЗ на показники надійності відмовостійкої багатопроцесорної системи. Зокрема було показано, що при збільшенні середнього значення тривалості першого оновлення версії ПЗ тривалість експлуатації, коли забезпечується ймовірність безвідмовної роботи ПАС на рівні 0,99, зменшується. Отримані результати дали змогу встановити граничні значення тривалостей розробки оновлених версій ПЗ для забезпечення заданого рівня надійності ПАС. Досліджено вплив тривалості розробки оновлених версій ПЗ на показники надійності відмовостійкого програмно-апаратного засобу і встановлено, що при збільшенні середнього значення тривалості розробки оновлених версій ПЗ середнє значення безвідмовної роботи відмовостійкої ПАС зменшується [148, 149].

В [145] досліджували значення функції готовності в залежності від значення тривалості оновлення першої та другої версії ПЗ. Було показано, що на початкових етапах експлуатації ПАС при оновленні ПЗ та виправленні всіх його помилок функція готовності має провал, в залежності від тривалості оновлення ПЗ; встановлено, що оновлення першої версії ПЗ сильно впливає на початковий етап роботи ПАЗ.

В даній дисертаційній роботі досліджено вплив моделі надійності ПЗ на оцінку функції та коефіцієнта готовності ПАС з версійно-структурним резервуванням [135]. В якості моделей надійності ПЗ, які дають вхідне значення інтенсивності відмов ПЗ для моделі надійності ПАС, обрано МНПС (підрозділ

2.1) з одного боку, та моделі Goel–Okumoto [62] і S-подібна модель [63] як найбільш поширені моделі надійності ПЗ на основі НПП, з іншого.

В якості вхідних даних для отримання інтенсивності відмов ПЗ згідно різних моделей надійності було взято результати тестування двох ПЗ з роботи [90] (див. Додаток А). Як було показано в підрозділі 2.3.1, зазначені три моделі надійності майже однаково описують результати статистичних випробувань ПЗ в сенсі оцінювання загальної кількості помилок в обох програмних засобах. Однак поведінка функції інтенсивності відмов (див. рис. 2.4) є суттєво різною, що, безумовно, впливає на результати оцінювання показників надійності ПАС, для яких інтенсивність відмов ПЗ є одним із вхідних параметрів.

В табл. 2.7 наведено значення інтенсивностей відмов двох програмних засобів, отриманих на основі різних моделей надійності ПЗ. Як видно з табл. 2.7, незважаючи на майже однаковий прогноз кумулятивної кількості відмов (див. напр. табл. 2.2), значення інтенсивності відмов може відрізнятись на три–шість порядків (напр. λ_{sw11} дорівнює $1{,}221\times10^{-3}$ год$^{-1}$ у випадку використання для опрацювання результатів тестування моделі надійності ПЗ з показником складності [66] та $3{,}516\times10^{-6}$ год$^{-1}$ у випадку використання моделі Goel–Okumoto [62] чи $1{,}744\times10^{-9}$ год$^{-1}$ для S-подібної моделі [63]).

Таблиця 2.7

Інтенсивності відмов ПЗ, отримані на основі різних моделей надійності

λ, год$^{-1}$	**Модель Goel–Okumoto**	**S-подібна модель**	**МНПС**
λ_{sw11}	$3{,}516\times10^{-6}$	$1{,}744\times10^{-9}$	$1{,}221\times10^{-3}$
λ_{sw12}	$2{,}143\times10^{-9}$	$3{,}64\times10^{-14}$	$7{,}286\times10^{-8}$

Для порівняння значень функції готовності в залежності від значення інтенсивності відмов ПЗ, яке визначалося за допомогою різних моделей надійності ПЗ, чисельним методом розв'язували систему рівнянь (2.17) та знаходили імовірності перебування в різних станах ПАС (рис. 2.12), після цього

комбінуючи імовірності перебування у станах, де відсутні критичні відмови, згідно рівняння (2.18), отримували значення функції та коефіцієнта готовності ПАС.

На рис. 2.13 представлено залежності функції готовності при різних значеннях інтенсивності відмов ПЗ, які представлено в табл. 2.7. Розрахунки проводилися при наступних параметрах ПАС: λ_{hw} = 1×10^{-4} год $^{-1}$; $\lambda_{swerror}$ = 1×10^{-2} год$^{-1}$; T_{rest} = 6 хв.; T_{switch}, T_{rep} = 200 год; T_{up1} = 100 год; T_{up2} = 200 год. Розрахунок стаціонарного коефіцієнту готовності дає значення $К_Г = 0{,}977$ у випадку використання МНПС, та $К_Г = 0{,}981$ для двох інших моделей надійності ПЗ [135].

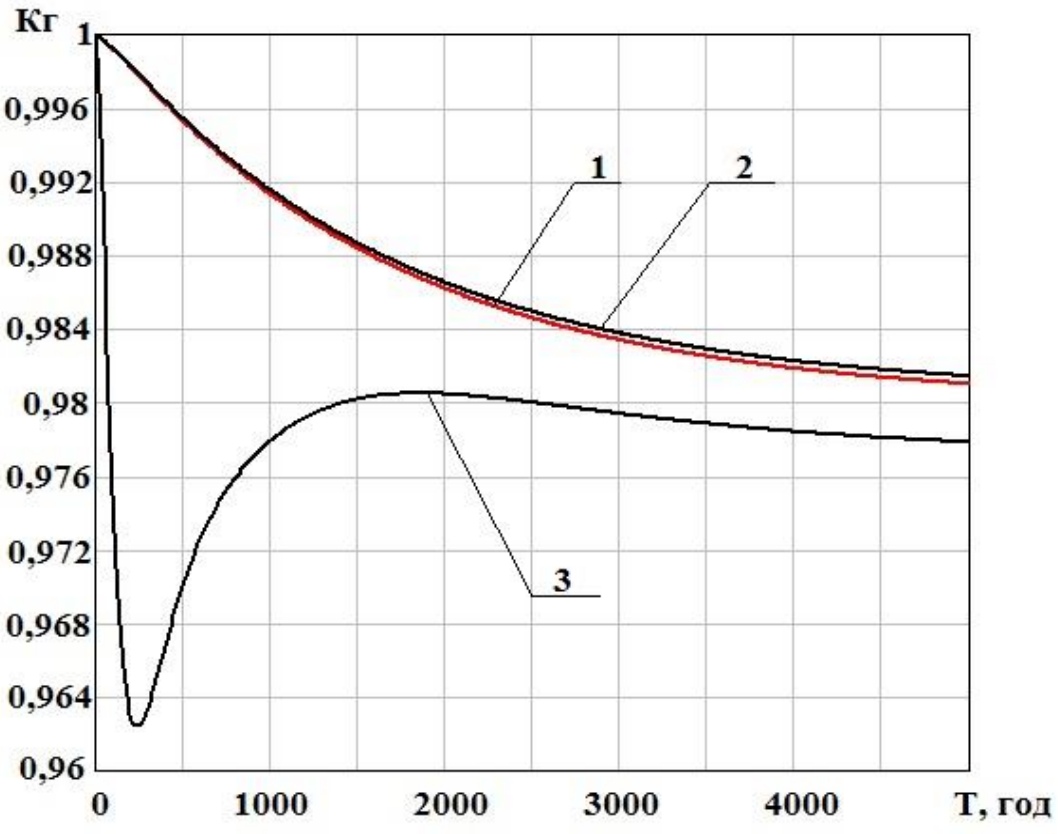


Рис. 2.13. Залежності функції готовності від часу при значеннях інтенсивності відмов ПЗ, отриманих на основі різних моделей (1 – модель Goel–Okumoto, 2 – S-подібна модель, 3 – МНПС).

Як видно з рис. 2.13 та табл. 2.7, значення інтенсивності відмов ПЗ, отримані на основі МНПС, є нижчими за значення інтенсивності відмов, отримані на основі найбільш поширених моделей надійності ПЗ на основі НПП. Таким чином, використання традиційних моделей надійності ПЗ призводить до завищених оцінок показників надійності ПАС, що не дає можливості достовірно оцінити ризики функціонування такої системи, та може потенційно спричинити значні втрати внаслідок відмов ПАС. Крім того, як видно з рис. 2.13, поведінка функції готовності ПАС, розрахованої на основі вхідних даних, отриманих з МНПС (крива 3, рис. 2.13), є якісно іншою, ніж поведінка функції готовності, отриманої з використанням значення інтенсивності відмов ПЗ на основі моделей Goel–Okumoto та S-подібної (криві 1, 2 на рис. 2.13). Дійсно, крива 3 на рис. 2.13 демонструє немонотонну залежність з екстремальною точкою, причому в даному випадку ця точка є точкою мінімуму та розташована в області невеликих значень часу, що може суттєвим чином відобразитись на поведінці даного ПАС в сенсі надійності, та повинно бути враховано під час експлуатації та регламентного технічного обслуговування таких систем.

2.5. Висновки до розділу 2

В цьому розділі отримали подальший розвиток математичні моделі надійності ПЗ на основі НПП, які використовують для опрацювання статистичних результатів тестування програмних засобів з метою отримання значень показників їх надійності. Для підвищення ступеня адекватності реальним об'єктам побудовано модель надійності з показником складності програмних засобів. Показано, що форми кривих кумулятивної кількості відмов та інтенсивності відмов ПЗ у випадку МНПС більш точно відповідають практичним результатам тестування ПЗ.

Наведено результати досліджень поведінки функції інтенсивності відмов МНПС в залежності від параметрів моделі, які дозволили встановити діапазони значень цього показника та пояснити поведінку параметру потоку відмов в залежності від складності програмних засобів. Показано, що МНПС більш

адекватно описує процес тестування ПЗ, причому залежність характеру відмов від складності програмного засобу має нелінійний характер і володіє мінімумом при значенні показника складності $s = 1$. Показано взаємозв'язок виразу для параметру потоку відмов МНПС з найбільш поширеними моделями надійності ПЗ, які є частковими випадками даної моделі. Показано, що вирази для різних показників, отриманих на основі МНПС (загальна кількість помилок, тривалість тестування тощо) відрізняються від базових виразів модифікаторами, які включають показник складності і мають нелінійну залежність стосовно цього показника.

Встановлено, що залежності точкових оцінок параметрів моделі для різних тестових профілів є близькими між собою і проявляють однакові якісні залежності, що підтверджує припущення про ефективність використання цих параметрів для критерію достатності процесу тестування та визначення точки переходу вибірки вхідних даних до пуассонового розподілу. Статистичні характеристики вибірки точкових оцінок параметрів моделі (розмах, мода, медіана) для різних тестових профілів показують високу однорідність даних, що підтверджує адекватність та ефективність використання МНПС для різних наборів вхідних даних.

Наведено метод оцінювання показників надійності програмного забезпечення на основі МНПС. Цей метод може бути використаний для оцінювання показників надійності ПЗ з урахуванням його складності, але без урахування його внутрішньої структури, а також для підтримки прийняття рішень з управління проектом з розробки ПЗ. МНПС та метод оцінювання надійності ПЗ на її основі пройшли верифікацію шляхом впровадження у виробництво [136], що дало можливість зменшити загальні економічні затрати на розробку ПЗ на 5%.

Досліджено вплив вибору моделі надійності ПЗ на результати оцінювання показників надійності ПАС з версійно-структурним резервуванням. Показано, що модель надійності ПЗ з показником складності, за рахунок підвищеного ступеня адекватності порівняно з найбільш поширеними S-подібною моделлю та

моделлю Goel–Okumoto, дає на 3–6 порядків нижчі значення інтенсивності відмов ПЗ, при незначній різниці в прогнозі загальної кількості відмов. Такі відмінності в динаміці процесу відмов спричиняють суттєві відхилення в оцінці показників надійності ПАС.

Показано, що використання традиційних моделей надійності ПЗ призводить до завищених оцінок показників надійності ПАС, і зокрема коефіцієнта готовності, що не дає можливості достовірно оцінити ризики функціонування такої системи. Поведінка функції готовності ПАС, розрахованої на основі вхідних даних, отриманих з моделі надійності ПЗ з показником складності демонструє немонотонну залежність з екстремальною точкою, причому в даному випадку ця точка є точкою мінімуму та розташована в області невеликих значень часу, що може суттєвим чином відобразитись на поведінці даного ПАС в сенсі надійності, та повинно бути враховано під час експлуатації та регламентного технічного обслуговування таких систем.

РОЗДІЛ 3. ОЦІНЮВАННЯ ПОКАЗНИКІВ НАДІЙНОСТІ ПРОГРАМНИХ СИСТЕМ З УРАХУВАННЯМ ЇХ СКЛАДНОСТІ НА ОСНОВІ ДАНИХ ПРО НАДІЙНІСТЬ ЇХ СКЛАДОВИХ

В цьому розділі представлено моделі і методи аналізу складних програмних систем на основі математичного апарату марковських процесів вищого порядку, які дають змогу урахувати взаємозалежність виконання модулів програмної системи, і таким чином підвищити точність та достовірність оцінювання показників надійності програмної системи на основі даних про надійність її елементів (програмних модулів). Описано моделі надійності програмних систем (ПС) на основі ланцюгів Маркова вищого порядку (ЛМВП) з дискретним та неперервним часом, а також для випадку неідеальної передачі потоку управління між модулями. Здійснено верифікацію цих моделей надійності ПС на основі емпіричних даних для чотирьох програмних засобів і показано перевагу використання ЛМВП над моделями, що використовують математичний апарат класичних марковських процесів. Розроблено і описано узагальнений метод аналізу надійності ПЗ, який включає використання різних моделей надійності (як на основі НПП, так і на основі ЛМВП) в залежності від складності ПЗ на різних етапах ЖЦ.

Основні результати і висновки, наведені в цьому розділі, викладено в працях [57, 150–167].

3.1. Моделі надійності програмних систем на основі марковського процесу вищого порядку

Розроблені та описані в другому розділі цієї дисертаційної роботи моделі, методи і засоби, хоча і враховують складність ПЗ, відносяться до моделей типу "чорної скриньки". Вони володіють усіма недоліками, притаманними такому класу моделей, які описані в першому розділі. Модель надійності ПЗ з показником складності, як і практично усі моделі на основі НПП, призначена для оцінювання надійності програмних засобів, складність яких дозволяє нехтувати взаємозв'язками їх складових. По аналогії із методами і засобами оцінювання

надійності АЗ, такі моделі можна віднести до засобів оцінювання показників надійності елементів за даними статистичних випробувань. У випадку ж більш складного програмного забезпечення, яке називають програмними системами, слід використовувати для аналізу їх надійності компонентні моделі, описані в підрозділі 1.5.

Моделі оцінювання надійності ПС на основі архітектурного підходу у більшості випадків використовують теорію класичних марковських процесів, припускаючи незалежність виконання модулів (модулів) програмної системи, що є спрощеним описом реального процесу виконання ПЗ. В більш адекватних моделях необхідно враховувати те, що імовірність переходу в наступний стан (наприклад передачі потоку управління до іншого модуля програми) може залежати не тільки від поточного стану, а й від передісторії потрапляння в цей стан (тобто який модуль програми виконувався до того, як почав виконуватись поточний модуль). У роботах [168, 169] для усунення спрощення запропоновано використовувати ланцюги Маркова вищого порядку (ЛМВП), які дають змогу більш достовірно оцінити надійність ПС, оскільки враховується взаємозалежність виконання їх модулів.

3.1.1. Модель надійності ПС на основі ланцюгів Маркова вищого порядку з дискретним часом

У роботах [152, 154, 161, 170] удосконалено модель оцінювання надійності програмного забезпечення [15], в якій архітектуру ПС подано марковським процесом вищих порядків з дискретним часом. Розглядається поглинальний марковський процес, що передбачає наявність одного або більше поглинальних станів, тобто станів, з яких не можна перейти в інші стани. Під станом розуміють виконання відповідного модуля програмної системи. Таким чином елементами цієї моделі надійності є: $\{C_i\}$ – напрямлений граф, який відображає структуру програмної системи, вершини графа відповідають модулям системи, а ребра відображають передачу потоку управління між модулями ($i = \overline{1, N}$, де N – кількість модулів програмної системи); $\boldsymbol{P} = \{p_{ij\ldots kl}\}$ матриця ймовірностей

переходів в модуль l в залежності від виконання попередніх K модулів; $\lambda_i(t)$ – інтенсивності відмов кожного програмного модуля.

В моделі надійності ПС на основі ЛМВП з дискретним часом відображені наступні її особливості:

- програмна система складається зі скінченної кількості модулів, які з точки зору структурної схеми надійності з'єднані послідовно;
- передача управління між модулями ПС відбувається через однакові дискретні проміжки часу;
- кожен модуль ПС може бути протестований незалежно і характеризується інтенсивністю відмов $\lambda_i(t)$;
- операційний профіль ПС відповідає протестованим варіантам її використання і не змінюється з часом, а усі особливості поведінки ПС відображені у матриці імовірностей переходів $\boldsymbol{P}$;
- імовірності переходів p_{ij} між станами i та j залежать від переліку попередніх станів системи, який закінчується поточним станом i (кількість попередніх станів, що впливають на імовірність p_{ij} дорівнює порядку марковського процесу K);
- надійність i -ого модуля залежить тільки від сумарного часу виконання цього модуля та інтенсивності його відмов, але не залежить від конкретних моментів часу, в які було передано управління цьому модулю;
- модель є ієрархічною, тобто спочатку обчислюються параметри архітектури ПС на основі моделі її функціонування з використанням теорії марковських процесів, а далі враховується поведінка відмов кожного модуля.

Згідно такої моделі [154] ймовірність безвідмовної роботи ПС в цілому обчислюється за формулою:

$$R(t) = \prod_{i=1}^{N} R_i(t). \tag{3.1}$$

Для урахування можливої взаємної залежності передачі потоку управління між модулями системи, в [152, 154, 161] запропоновано ймовірність безвідмовної роботи кожного модуля R_i $\left(i = \overline{1, N}\right)$ обчислювати з використанням формалізму марковських процесів вищого порядку:

$$R_l(t) = \exp\left(- \int_0^{\sum_{ij\ldots k} V_{ij\ldots kl} t_{ij\ldots kl}} \lambda_l(\tau) d\tau \right). \tag{3.2}$$

Тут $V_{ij\ldots kl}$ – очікувана кількість відвідувань модуля l в залежності від виконання попередніх K модулів, $t_{ij\ldots kl}$ – тривалість виконання модуля l у залежності від виконання попередніх K модулів. Для знаходження $V_{ij\ldots kl}$ потрібно розв’язати таку систему лінійних алгебраїчних рівнянь:

$$V_{j\ldots kl} = q_{j\ldots kl} + \sum_{i=1}^{N-1} V_{ij\ldots k} p_{j\ldots kl}, \tag{3.3}$$

де $p_{ij\ldots kl}$ – ймовірність переходу в модуль l в залежності від виконання попередніх K модулів, $q_{ij\ldots kl}$ – початковий ймовірнісний вектор.

Отримавши числові значення усіх параметрів моделі, можна обчислити як ймовірність безвідмовної роботи кожного модуля так і всієї системи за формулами (3.2) та (3.1) відповідно. Таким чином використання математичного апарату ЛМВП, у свою чергу, призводить до необхідності розв’язку наступних задач для практичного використання моделі (3.2):

- визначення інтенсивності відмов $\lambda_i(t)$ кожного модуля ПЗ;
- визначення матриці ймовірностей переходів між модулями в залежності від виконання попередніх K модулів;
- визначення оптимального порядку K Марковського ланцюга.

На основі моделі оцінювання надійності ПС з використанням ЛМВП, яка дає змогу врахувати взаємозалежність виконання модулів та адекватно

змоделювати процес аналізу надійності ПС, у [161, 170] розроблений метод оцінювання надійності ПС (рис. 3.1), що складається з таких основних двох етапів:

1. *Визначення вхідних параметрів моделі оцінювання надійності ПС*

1.1.Визначення інтенсивності відмов кожного модуля.

Для обчислення інтенсивності відмов кожного модуля можна використати відомі моделі надійності ПЗ. Переважна частина таких моделей – це моделі на основі НПП. Показано, що багато існуючих моделей надійності ПЗ можна сформулювати в межах НПП [61, 62]. Численні емпіричні спостереження підтверджують валідність моделей цього типу [171].

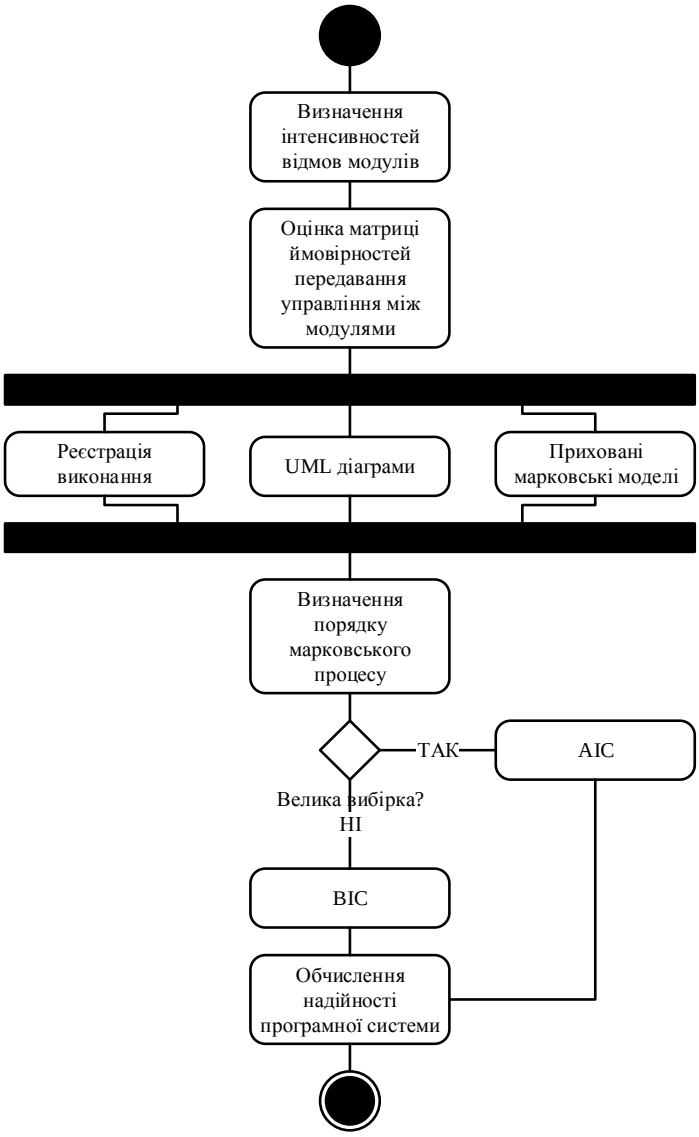


Рис 3.1. Схема методу аналізу надійності ПС на основі ЛМВП з дискретним часом.

Для обчислення $\lambda_i(t)$ кожного модуля ПС можна використати МНПС, описану в підрозділі 2.1. Для використання цієї моделі необхідно попередньо здійснити тестування програмного засобу для отримання статистики відмов та їх розподілу в часі. Для отримання більш достовірних емпіричних даних необхідно проводити тестування з максимальним покриттям коду, що дозволить виявити більшу кількість помилок. Для забезпечення цієї умови доцільно використовувати метод автоматизованого формування сценаріїв тестування (АФСТ) на основі моделі функціонування ПС з урахуванням множини значень змінних коду [172, 173].

1.2. Обчислення матриці ймовірностей переходів між модулями ПС та тривалості виконання модулів.

Визначення ймовірностей переходів між модулями на сьогодні залишається важким в практичній реалізації завданням. Одним із способів вирішення даної проблеми є підхід, що базується результатах моніторингу виконання ПС [156], адже при виконанні ПС управління передається між модулями, і маючи результати передачі управляння можна визначити показники переходів між модулями. Для визначення ймовірностей переходів в модуль C_j з C_i необхідно визначити відношення кількості переходів з C_i в модуль C_j до загальної кількості переходів з модуля C_i. Для перевірки запропонованого підходу авторами розроблено реєстратор (logger) на основі мови програмування Java, який записує протокол виконання програм, написаних мовою Java [156]. Розроблений реєстратор дозволяє досліджувати програмні засоби без внесення змін в програмний код. Такий підхід дає можливість визначати та уточнювати матрицю ймовірностей переходів між модулями ПС навіть на етапі його впровадження та експлуатації, з урахуванням реальних сценаріїв використання ПС різними класами користувачів.

Окрім того для прогнозування надійності програмних систем на ранніх етапах життєвого циклу, що дозволить уникнути значних витрат для виправлення відмов на більш пізніх етапах, визначення матриці ймовірностей

переходів можна здійснювати на основі аналізу UML діаграм [153, 155, 158]. З діаграми класів можна сформувати матрицю ймовірностей переходів між модулями, а саме її структуру. Проте необхідно зауважити, що з цієї діаграми неможливо отримати значення ймовірності переходів та тривалості виконання модулів ПС. З використанням діаграми прецедентів на основі ймовірностей (або відносних частот) виконання усіх випадків використання можна оцінити ймовірності виконання модулів ПС та переходів між ними. Окрім того використовуючи UML діаграму послідовностей, можна отримати кількість переходів між модулями, з яких легко відповідно визначити матрицю ймовірності переходів між ними, а також тривалість перебування в модулі. Провівши аналіз розроблених UML діаграм послідовностей під час проектування ПС необхідно порахувати сумарну кількість переходів з модуля в модуль і сформувати матрицю переходів між модулями відповідно до підходу описаного в [156]. Але необхідно пам’ятати, що на етапі проектування можна тільки наближено говорити про модулі ПС, а тому згадані оцінки імовірностей отримують апріорно на основі експертних оцінок.

Іншим варіантом отримання матриці ймовірностей переходів між модулями є використання прихованих Марковських ланцюгів. Завдяки своїм можливостям приховані марковські моделі дедалі частіше використовуються при моделюванні надійності ПС. Зокрема використання таких моделей добре зарекомендувало себе при діагностиці несправностей, моделювання мережевого трафіку, розпізнавання мови, аналізі генетичної послідовності та ін. [174–177]. Використання прихованих марковських моделей потребує розв’язання трьох таких фундаментальних задач: 1) знаходження ймовірності появи спостережуваної послідовності; 2) визначення сукупності станів, переходи між якими найбільш імовірно приводять до генерування спостережуваної послідовності; 3) визначення значення параметрів матриці ймовірностей передавання управління між модулями і матриці ймовірностей відповідності прихованих і спостережуваних станів для прихованих марковських моделей [178]. Для вирішення цих задач можна використати алгоритм Baum–Welch [179].

Отже, для отримання матриці ймовірностей переходів між модулями потрібно перейти до марковської моделі, де прихованими станами є виконання того чи іншого модуля системи, а спостережуваними станами – зависання програми, виявлення помилок програми, аварійне завершення системи на певному етапі її виконання. Для запуску програми на основі алгоритму Baum–Welch необхідно проініціалізувати матрицю переходів, матрицю спостережень і початковий вектор станів прихованої марковської моделі, а також мати певну послідовність спостережень за системою. На виході алгоритму отримується початковий вектор ймовірностей та матриця ймовірностей переходів між модулями.

1.3.Визначення порядку марковського процесу.

Для визначення порядку марковського процесу в задачах оцінювання надійності ПС запропоновано використовувати інформаційні критерії, застосування яких описано у підрозділі 3.2. Для малих обсягів вибірки емпіричних даних стосовно функціонування надійності програмного продукту доцільно використовувати байєсівський критерій BIC, який дає змогу накладати сильніші штрафи вже для обсягу $n \geq 8$. Це дає можливість уникнути необґрунтованого підвищення оптимального порядку моделі через недостатню кількість емпіричних даних і збільшує ймовірність вибору "точної" моделі (якщо вона існує). Якщо обсяг вибірки є великим, то для вибору критерію потрібні експериментальні дослідження, оскільки за допомогою критерію Akaike, з одного боку, не можна передбачити наявність "точної" моделі, а лише оцінити ступінь наближення моделей тестової множини до "істинної" моделі. З іншого боку, із використанням критерію BIC при збільшенні обсягу вибірки можна збільшити ймовірність вибору "точної" моделі (якщо вона існує).

2. Оцінювання надійності ПС з використанням моделі надійності у вигляді ланцюгів Маркова вищого порядку.

Отримавши необхідні вхідні параметри для розробленої моделі оцінювання надійності ПС, спочатку з використанням формули (3.3) визначається очікувана кількість виконання модуля i в залежності від виконання

попередніх N модулів. Для цього використовується чисельний метод Зейделя для розв'язування системи лінійних алгебраїчних рівнянь. Після цього знаходиться надійність кожного модуля за формулою (3.2), чисельним інтегруванням методом Гауса-Лежандра, після чого отримується значення надійності комп'ютерної програми у цілому з використанням формули (3.1).

3.1.2. Модель надійності ПЗ на основі ланцюгів Маркова вищого порядку з неперервним часом

Використання моделі надійності на основі ланцюгів Маркова з дискретним часом має ряд обмежень та спрощень, що були описані в попередньому підрозділі. Так, ця модель враховує тільки сумарну тривалість виконання i-ого модуля та інтенсивність його відмов, не враховуючи розподіл часу виконання модуля та конкретні моменти часу, в які відбувалась передача потоку управління цьому модулю. Це може бути важливим у випадку залежності інтенсивності відмов модуля від часу. Іншим спрощенням є те, що в реальних програмних системах передача управління між модулями відбувається в довільні моменти часу. Крім того на результати, отримані за допомогою моделі з дискретним часом, сильно впливатиме значення кванту часу, з яким здійснюється дискретизація поведінки системи. Тому, для підвищення ступеня адекватності моделі надійності ПС, що враховує його архітектуру, слід використовувати ланцюги Маркова з неперервним часом.

Модель надійності ПС на основі ЛМВП з неперервним часом складається з наступних складових: $\{C_i\}$ – напрямлений граф, який відображає структуру ПС, вершини графа відповідають модулям системи (програмним модулям), а ребра відображають передачу потоку управління між модулями ($i = \overline{1,N}$, де N – кількість модулів програмної системи); $\boldsymbol{A} = \{a_{ij}(t)\}$ – матриця інтенсивності переходів між станами системи, які відповідають вершинам графа $\{C_i\}$; $\boldsymbol{P} = \{p_i(t)\}$ вектор імовірностей перебування системи в стані C_i в момент часу t; $\lambda_i(t)$ – інтенсивність відмов i-ого програмного модуля. Зауважимо, що матриця інтенсивності переходів між станами системи відображає кількість переходів зі

стану i в стан j за одиницю часу, і в загальному випадку її елементи можуть бути залежними від часу $a_{ij} = f(t)$, а у випадку використання ЛМВП, залежать від шляху, яким система потрапила в стан i.

В моделі надійності ПС на основі ЛМВП з неперервним часом відображені наступні її особливості:

- ПС складається зі скінченної кількості модулів, які з точки зору структурної схеми надійності з'єднані послідовно;
- передача управління між модулями ПС відбувається в довільні моменти часу;
- кожен модуль ПС може бути протестований незалежно і характеризується інтенсивністю відмов $\lambda_i(t)$;
- операційний профіль програми відповідає протестованим варіантам використання ПС і не змінюється з часом, а усі особливості поведінки ПС відображені у матриці інтенсивності переходів $\boldsymbol{A}$;
- інтенсивність переходів a_{ij} між станами i та j залежить від історії переходів між попередніми станами системи, яка закінчується поточним станом i (кількість попередніх станів, що впливають на інтенсивність a_{ij} дорівнює порядку марковського процесу K);
- надійність i-ого модуля залежить від конкретних моментів часу, в які було передано управління цьому модулю (у випадку, якщо $\lambda_i = f(t)$);
- в довільний момент часу t відмова ПС може бути спричинена виключно відмовою того модуля, який виконується в даний момент часу[1];

[1] Зауважимо, що дане твердження є справедливим для послідовного з'єднання елементів в структурній схемі надійності. У випадку ж наявності паралелізму (наприклад в паралельних системах, що передбачають виконання декількох потоків на декількох ядрах/процесорах одночасно) таких модулів в момент часу t буде декілька, і ймовірність відмови системи буде добутком імовірностей відмов цих модулів (за умови, що їх відмови є незалежними випадковими подіями).

- модель є ієрархічною, тобто спочатку обчислюються параметри архітектури ПС на основі моделі її функціонування з використанням теорії марковських процесів, а далі враховується поведінка відмов кожного модуля.

В такому випадку інтенсивність відмов ПС, яка складається з N модулів може бути записана як [180]:

$$\lambda(t) = \sum_{i=1}^{N} p_i(t)\lambda_i(t), \tag{3.4}$$

де $\lambda_i(t)$ – інтенсивність відмов i-ої компонети, $p_i(t)$ – імовірність виконання i-ого модуля в момент часу t (що відповідає перебуванню системи в стані C_i).

Як і у випадку моделі з дискретним часом [154, 175] інтенсивність відмов модулів ПС може бути отримана на основі результатів модульного тестування з використанням відомих моделей надійності типу "чорної скриньки", наприклад МНПС на основі НПП [181].

Якщо потік управління між модулями системи моделюється як процес Маркова з неперервним часом (стан C_i процесу відповідає виконанню i-го програмного модуля), то імовірність перебування системи в i-му стані $p_i(t)$ отримують з розв'язку системи рівнянь Колмогорова–Чепмена [182]:

$$\frac{dp_i(t)}{dt} = -\sum_{j \in S} a_{ij}(t)p_i(t) + \sum_{j \in S} a_{ji}(t)p_j(t)\,, i \in S, \tag{3.5}$$

де $a_{ij}(t)$ – інтенсивність переходу зі стану i в стан j, а S позначає множину усіх станів системи.

Як уже зазначалось вплив процесу вищого порядку виражається в залежності інтенсивності переходу між станами від історії потрапляння в поточний стан, і чим вищий порядок процесу, тим триваліший ланцюжок переходів впливає на цю інтенсивність. В цьому випадку система рівнянь (3.5) повинна бути відповідним чином модифікована. Інший підхід для урахування процесу вищого порядку був запропонований в [159, 164], де автори наводять метод побудови еквівалентного процесу першого порядку, використання якого

зводить задачу до обчислення імовірностей $p_i(t)$ на основі класичної системи рівнянь Колмогорова–Чепмена (3.5).

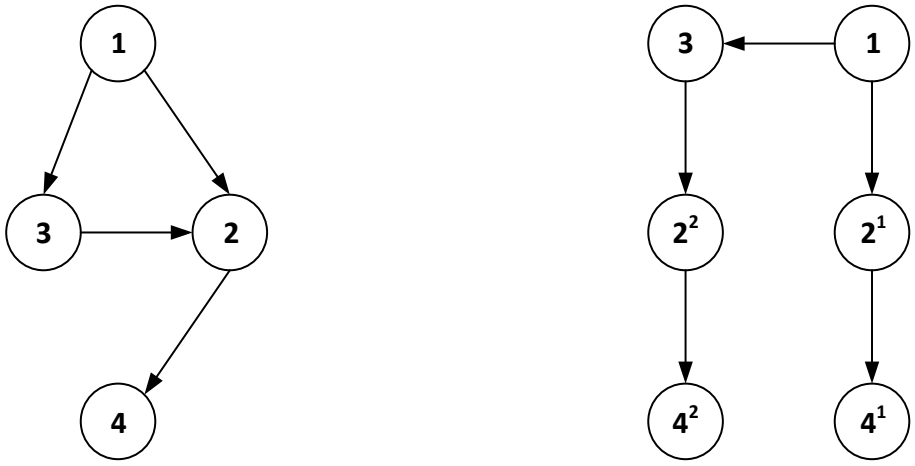
З теорії марковських процесів відомо, що процес вищого порядку може бути представлений у вигляді процесу першого порядку шляхом відповідного перевизначення простору станів [183]. Для програмної реалізації моделей, що використовують ЛМВП, слід мати формалізований алгоритм такого подання. Так, наприклад, у роботах [184, 185] пропонують матрицю ймовірностей Марковського процесу другого порядку $\boldsymbol{P}^2$, що містить S станів, представляти у вигляді:

$$P^2 = \begin{pmatrix} p_{111} & \cdots & p_{11N} \\ \vdots & \ddots & \vdots \\ p_{NN1} & \cdots & p_{NNN} \end{pmatrix},$$

де p_{ijk} – ймовірність переходу зі стану C_i в стан C_j, а потім в стан C_k. Але така матриця є сильно розрідженою, у зв'язку з тим, що не існує шляхів другого порядку між відповідними станами, тому таке представлення не є ефективним з точки програмної реалізації, оскільки вимагає великих затрат пам'яті для зберігання нульових елементів даної матриці або ж використання спеціалізованих методів і структур даних для роботи з розрідженими матрицями.

Скориставшись аналогією з відомим методом фаз Ерланга [186] у [159, 164] представлено марковський процес вищого порядку у вигляді еквівалентного процесу першого порядку з додатковими фіктивними станами. При цьому кожен стан початкового графа розщеплюється на таку кількість фіктивних станів, скільки є різних шляхів до цього стану. Таким чином вирішення задачі, по суті, зводиться до використання апарату теорії графів.

Для ілюстрації підходу, запропонованого в [159] на рис. 3.2(а) зображено граф переходів для деякої системи із чотирьох модулів. Якщо розглядати Марковський процес другого порядку, то в стан C_4 можна потрапити двома шляхами $C_1 \to C_2 \to C_4$ та $C_3 \to C_2 \to C_4$. Відповідно, стан C_2 можна розщепити на два фіктивних стани C_2^1 та C_2^2 і система матиме вигляд, представлений на рис. 3.2(б).



а) *б)*

Рис. 3.2. Приклад еквівалентного перетворення Марковського процесу другого порядку в процес першого порядку (*а* – граф станів вихідної системи, *б* – граф станів системи з фіктивними станами).

Нехай M^K – матриця, елементи якої m_{ij}^K означають кількість можливих шляхів порядку K зі стану C_i до C_j. Розглянемо матрицю M^1, елементами якої є кількість переходів першого порядку між відповідними станами, зображеними на рис. 3.2(а) ($m_{12}^1 = 1$ – елемент матриці M^1, який означає, що зі стану C_1 в C_2 є один можливий шлях першого порядку):

$$M^1 = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Неважко показати, що матриця M^2, яка міститиме кількість ланцюжків зі стану C_i в C_j до другого порядку включно, матиме вигляд:

$$M^2 = \begin{pmatrix} 0 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

де $m_{12}^2 = 2$ означає, що зі стану C_1 в стан C_2 існує два шляхи – один довжиною 2 (Марковський ланцюг другого порядку) $C_1 \to C_3 \to C_2$ та один ланцюжок довжиною 1 – $C_1 \to C_2$.

Матриця M^K має наступну властивість: сума елементів у j-му стовпчику матриці дорівнює кількості шляхів, що входять в стан C_j (кількості фіктивних станів, на яку потрібно розщепити стан C_j), а сума елементів у i-му рядку матриці рівна кількості шляхів, що виходять зі стану C_i.

Отже, для обчислення кількості ланцюжків K-порядку зі стану C_i в стан C_j можна скористатись формулою, що базується на методі Флойда:

$$m_{ij}^K = \sum_{j=1}^{N} \left(m_{il}^{K-1} + e_{il}\right) m_{lj}^1, \tag{3.6}$$

тут N – кількість всіх станів системи, e_{ij} – елементи одиничної матриці.

Таким чином отримується розширена матриця імовірностей переходів між станами, на основі якої формується система диференціальних рівнянь Колмогорова–Чепмена. У випадку Марковського процесу змінного порядку у виразі (3.6) для кожного стану використовують різне значення порядку процесу без зміни суті розрахунку.

На основі (3.6) отримують еквівалентний граф станів системи $\{C'_i\}$, який є еквівалентним представленням вихідного графу $\{C_i\}$ у випадку процесу K-го порядку. Це дозволяє побудувати систему рівнянь Колмогорова–Чепмена (3.5) для еквівалентного графу та отримати з її розв'язку імовірності перебування системи в станах C_i, що відповідає імовірностям виконання i-го програмного модуля в момент часу t. Використавши отримані імовірності в рівнянні (3.4) разом з отриманими в процесі модульного тестування значеннями інтенсивності відмов кожного модуля отримують часову залежність інтенсивності відмов програмної системи в цілому. Інші показники надійності можуть бути розраховані за відомими співвідношеннями між показниками надійності техніки [182].

У [49, 69] описано модель надійності ПС з урахуванням модульної структури. Ця модель представляє передачу управління між модулями ПС як марковський процес, при цьому розглядаються два типи відмов системи. Перший тип відмов пов'язаний з відмовами програмних модулів, вважаючи, що ці

відмови реалізують пуассонів процес. Другий тип відмов породжується зв'язками між модулями – вважається, що при інтеграції модулів в систему вносяться нові помилки, а відмови такого типу пов'язані з інтерфейсами модулів. Припускаючи, що відмови модулів та інтерфейсів є незалежними випадковими подіями, Літлвуд показав [69], що процес відмов ПС в цілому є асимптотично пуассоновим.

Нехай N – кількість модулів ПС; λ_i – пуассонова інтенсивність відмов i-го модуля; q_{ij} – імовірність виникнення відмови при переході системи зі стану i в стан j; a_{ij} – матриця інтенсивності переходів між станами системи; $\mathbf{\Pi} = \{\pi_i\}$ – рівноважний вектор матриці інтенсивності переходів між станами системи.

У [69] показано, що при $\lambda_i, q_{ij} \to 0$, процес відмов ПС є асимптотично пуассоновим процесом з інтенсивністю $\sum_{i=1}^{N} \pi_i\left(\lambda_i + \sum_{j\neq i} a_{ij} q_{ij}\right)$.

Взявши за основу модель Літлвуда, побудуємо модель надійності ПС вищого порядку, в якій відображені наступні її особливості:

- ПС складається зі скінченної кількості модулів $(1,2,\dots,N)$, які можна представити напрямленим графом $\{C_i\}$, і які з точки зору структурної схеми надійності з'єднані послідовно;
- передача управління між модулями ПС відбувається в будь-які моменти часу (час наступного переходу теж є випадковою величиною) та описується однорідним ланцюгом Маркова з неперервним часом $\{X(t)\}$;
- кожен модуль ПС може бути протестований незалежно і характеризується інтенсивністю відмов λ_i, при цьому процес виникнення відмов для кожного модуля представлений однорідним пуассоновим (для кількості подій "відмова") або експоненційним (для інтервалів часу між відмовами) законами розподілу;
- поведінка ПС в часі (потік управління) описується марковською матрицею інтенсивності переходів $\boldsymbol{A}$ ($a_{ij}\delta t = P\{X(t+\delta t) = j | X(t) = i\}, i \neq j, a_{ii} = -\sum_{j\neq i} a_{ij}$);

- операційний профіль програми відповідає протестованим варіантам використання системи і не змінюється з часом, а усі особливості поведінки ПС відображені у матриці інтенсивності переходів $\boldsymbol{A}$;
- інтенсивність переходів a_{ij} між станами i та j залежить від історії переходів між попередніми станами системи, яка закінчується поточним станом i (кількість попередніх станів, що впливають на інтенсивність a_{ij} визначає порядок марковського процесу K);
- вважається, що інтеграція модулів в програмну систему є неідеальною і на додачу до відмов програмних модулів виникають відмови, пов'язані з переходами між модулями; q_{ij} позначає імовірність таких відмов при передачі управління від модуля i до модуля j;
- в довільний момент часу t відмова ПС може бути спричинена відмовою програмного модуля, який виконується в даний момент часу, та відмовою при передачі управління між модулями. Імовірність відмови ПС є умовною імовірністю відмови такого модуля за умови безпомилкової передачі управління цьому модулю;
- модель є ієрархічною, тобто спочатку обчислюються параметри архітектури ПС на основі моделі її функціонування з використанням теорії марковських процесів, а далі враховується поведінка відмов кожного модуля.

З урахуванням наведених припущень інтенсивність відмов програмної системи можна записати як:

$$\lambda(t) = \sum_{i=1}^{N} p_i(t)\lambda_i + \sum_{i \neq j} a_{ij}(t)q_{ij}, \tag{3.7}$$

тут елементи матриці інтенсивності переходів залежать від історії переходів до поточного стану довжиною K (де K – оптимальний порядок ланцюга Маркова) і в загальному випадку можуть бути функціями часу.

Зауважимо, що хоча інтенсивності відмов модулів ПС не залежать від часу (за умови розгляду однієї її версії), інтенсивність відмов ПС є функцією часу виконання (3.7), що є наслідком складної структури системи до якої входять модулі з різною інтенсивністю відмов.

Як і у випадку моделі (3.4) ефективна робота з матрицею інтенсивності переходів вищого порядку може бути досягнута шляхом побудови еквівалентного марковського процесу першого порядку [159, 164] та розв'язком рівняння Колмогорова–Чепмена для такого процесу з подальшою агрегацією фіктивних станів (зауважимо, що інтенсивність відмов програмного модуля, на відміну від інтенсивності переходів між модулями, не залежить від шляху передачі управління до цього модуля).

Важливим практичним питанням використання цієї моделі є визначення матриці імовірності виникнення відмов при передачі управління між модулями системи. Не виключаючи чисто емпіричного підходу до вирішення цієї задачі (шляхом, наприклад, реєстрації і аналізу викликів методів (функцій) програмних модулів та відмов, що при цьому виникають), можна запропонувати оцінку такої імовірності на основі кількості інформації, що передається при виклику функції та структурної складності ПС:

$$q_{ij} = f(I(i,j), M), \tag{3.8}$$

де $I(i,j)$ – кількість інформації, що передається між модулями і залежить від кількості та типу задіяних програмних змінних, M – цикломатична складність програми (метрика MacCabe), $M = b_1(G,t) = rank\, H_1(G,t)$. Обидва значення можуть бути отримані під час розробки ПС шляхом статичного аналізу програмного коду.

Модель надійності ПС у вигляді (3.7) є однією з найбільш загальних моделей, яка зводиться до моделі (3.4) у випадку $q_{ij} = 0$, чи до моделі типу (2.1) у випадку розгляду системи як чорної скриньки, тобто з одним модулем.

3.2. Засоби визначення оптимального порядку марковського процесу в задачах аналізу надійності ПС

Практичне використання моделей на основі ЛМВП передбачає визначення порядку марковського процесу, який залежатиме від ПС, що досліджується, тобто від об'єкту моделювання. Така задача належить до задач структурної ідентифікації, або, іншими словами, задачі створення моделей за експериментальними даними, що зводиться до формування за даними вибірки деякої множини F моделей різної структури та вибору серед них найефективнішої.

У загальному, критерії, що використовуються для вибору адекватної моделі, можна розділити на три основні групи [187]:

- статистичні критерії, що базуються на обчисленні величини помилки результату, який отримуємо із застосуванням моделі, та застосуванні апарату перевірки статистичних гіпотез;
- зовнішні критерії, які застосовуються в методі групового урахування аргументів (МГУА) та ґрунтуються на розбитті вибірки на дві й більше частин. На першій частині – «внутрішній» – здійснюється оцінювання параметрів, на другій – «зовнішній» – визначається прогнозна здатність моделей. Застосовуються чотири групи таких критеріїв: а саме критерії точності, згоди, балансу і динамічні критерії;
- критерії з явним штрафом за складність (зокрема, критерії AIC, BIC та Mallows), до помилки моделі "приєднують" адитивні або мультиплікативні штрафні члени, які залежать від кількості точок у вибірці і від складності (кількості параметрів) моделі.

Для визначення порядку марковського процесу в задачах оцінки надійності ПС пропонується використовувати інформаційні критерії, тобто критерії зі штрафом за складність, адже вони не є тестами перевірки гіпотези та не використовують рівень значущості [188]. У загальному інформаційні критерії

базуються на методі максимальної правдоподібності та відрізняються різною функцією штрафів за складність моделі (кількість її параметрів). Якщо можна оцінити дисперсію помилок моделі, то використовується критерій Mallows [189], в інакшому випадку критерії сімейства Akaike (AIC) та Bayes (BIC). Ці критерії є критеріями вибору моделі – інструментами порівняння відповідності реальному процесу декількох моделей на основі однакових спостережуваних даних та вибору найефективнішої із даної множини. Ці критерії широко використовувались в ряді досліджень, присвячених задачам прогнозування погоди, вибору адекватної екологічної моделі [190, 191]. Слід зауважити, що якщо модель, що найбільше відповідає реальному процесу, існує, але не належить множині, з якої здійснюється вибір, то дані критерії не в змозі її визначити. Разом з тим, застосування цих критеріїв в задачах моделювання надійності ПС залишається малодослідженим.

Варто зауважити, що у теорії інформаційних критеріїв розглядувані моделі, за означенням, є лише апроксимацією реального процесу. Крім цього, "краща модель" для аналізу даних залежить від обсягу вибірки, адже деякі властивості обєкта дослідження, які мають бути відображені в моделі, часто можуть бути виявлені тільки за рахунок збільшення обсягу спостережень.

Проаналізуємо два класи інформаційних критеріїв [151, 154, 160], які базуються: 1) на мінімізації відстані Кульбака–Лейблера (міра відхилення моделі, що розглядається, від реального об’єкту) та 2) на факті Байєса, та розглянемо їх застосування для випадку моделювання поведінки ПЗ марковським ланцюгом вищого порядку.

Критерії, побудовані на основі мінімізації відстані Кульбака–Лейблера. Інформаційний критерій Akaike (AIC) використовується для вибору адекватної моделі з набору моделей та базується на розширеному методі максимальної правдоподібності. Його вперше опублікував Akaike у 1974 році [192]. Модель обирається так, щоб у ній мінімізувалась відстань Кульбака–Лейблера, яка є мірою відхилення двох ймовірнісних розподілів, один з яких, як правило, є "істинним", а інший – "тестовим".

У загальному випадку такий критерій обчислюється як:

$$AIC = 2k - 2\ln(L), \tag{3.9}$$

де k – кількість параметрів в статистичній моделі, L – значення функції максимальної правдоподібності моделі.

Розглянемо ПС, що складається з N модулів (функціональні одиниці, які можна тестувати незалежно одна від одної) [150]. Вважаємо, що процес поведінки (передачі потоку управління між модулями) ПЗ моделюється марковським процесом, який містить N станів.

Через $n_{ij\dots kl}$ позначимо кількість переходів з модуля i в j залежно від перебування в попередніх модулях ($i \to j \to \cdots \to k \to l$) у спостережуваній послідовності, а $p_{ij\dots kl}$ – ймовірності переходів між модулями у даній послідовності, які обчислюються за формулою:

$$p_{ij\dots kl} = \frac{n_{ij\dots kl}}{n_{ij\dots k}}, \tag{3.10}$$

де $n_{ij\dots k} = \sum_{i=1}^{N} n_{ij\dots kl}$.

Тоді функція правдоподібності в такому випадку для критерію AIC дорівнює:

$$L = \prod_{i,j,\dots,k,l} p_{ij\dots kl}^{n_{ij\dots kl}}. \tag{3.11}$$

де $i, j, \dots, k, l = \overline{1, N}$.

Якщо підставити значення функції правдоподібності даної моделі ПЗ (3.11) у вираз (3.9), а замість параметра k – кількість параметрів моделі K-порядку, що містить N модулів ($k = N^K(N-1)$), то отримаємо вираз

$$AIC(K) = 2N^K(N-1) - 2\ln\left(\prod_{i,j,\dots,k,l} p_{ij\dots kl}^{n_{ij\dots kl}}\right), \tag{3.12}$$

який перепишемо у вигляді:

$$AIC(K) = 2N^K(N-1) - 2\sum_{i,j,\dots,k.l} n_{ij\dots kl}\ln\left(p_{ij\dots kl}\right). \tag{3.13}$$

Покладемо $K = 2; 3$ і перепишемо співвідношення (3.13) відповідно

$$AIC(2) = 2N^2(N-1) - 2\sum_{i=1}^{N}\sum_{j=1}^{N} n_{ij}\ln(p_{ij}),$$

$$AIC(3) = 2N^3(N-1) - 2\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} n_{ijk}\ln(p_{ijk}),$$

і т.д.

Значення порядку K, що мінімізує критерій AIC (3.13) відповідає оптимальному порядку марковського ланцюга вищого порядку для даної моделі надійності ПС.

Як бачимо, у формулі (3.13) даний критерій не залежить від обсягу вибірки. Тому критерій AIC застосовується для випадку великого обсягу вибірки (спостережуваної послідовності), тобто коли програмне забезпечення тестується багато разів і записується послідовність виконання модулів ПС.

Для того, щоб застосувати даний критерій для невеликої кількості тестувань ПС та відповідно невеликого обсягу вибірки, варто використовувати *критерій AIC другого порядку* AICc [193], що в загальному випадку визначається у вигляді:

$$AICc = -2\ln(L) + \frac{2kn}{n-k-1} = AIC + \frac{2k(k+1)}{n-k-1}, \tag{3.14}$$

де n – обсяг спостережуваної послідовності. Очевидно, що зі збільшенням обсягу вибірки критерій AICc наближається до критерію AIC (адже оскільки $n \gg k$, то $n-k-1 \to n$).

Для випадку пошуку оптимального порядку K марковського ланцюга моделі надійності ПС, що містить N станів із використанням критерію AIC другого порядку отримаємо:

$$AICc(K) = AIC(K) + \frac{2N^K(N-1)(N^K(N-1)+1)}{n-k-1}. \tag{3.15}$$

Варто зазначити, що для обчислення відстані Кульбака-Лейблера для K-тої марковської моделі, можна скористатись формулою [187]:

$$\Delta K = AIC(K) - AIC_{min}. \tag{3.16}$$

Також існують інші критерії, що базуються на мінімізації відстані Кульбака–Лейблера, але всі вони є розширенням критерію AIC (QAIC, TIC та ін.) [194].

Байєсівські критерії. Альтернативою критерію AIC є байєсівський інформаційний критерій (BIC) [195], який запропонував Schwarz, що враховує кількість елементів спостережуваної послідовності і виражається у вигляді:

$$BIC(K) = -2\ln(L) + \ln(n)\,k. \tag{3.17}$$

Критерій BIC може бути розглянутий як апроксимація відношення апостеріорної ймовірності до апріорної. Легко помітити, що наведений критерій є аналогом критерію AIC зі строгішою функцією штрафів (вже починаючи з обсягу вибірки $n \geq 8$, $\ln(n)\,k$ перевищує $2k$).

Особливістю даного критерію є його застосування для випадку невеликого обсягу спостережуваної вибірки.

У випадку використання критерію BIC для визначення оптимального порядку марковського ланцюга, що використовується при моделюванні процесу відмов ПС, отримується формула аналогічно як і для критерію AIC:

$$BIC(K) = 2N^K(N-1)\ln(n) - 2\sum_{i,j,\ldots,k,l} n_{ij\ldots kl}\ln p_{ij\ldots kl}. \tag{3.18}$$

Одним із критеріїв, що належить до даної групи і використовується для вибірок, обсяг яких $n \leq 20$ є інформаційний критерій Hannan–Quinn (HQIC) [185], який ґрунтується на припущенні, що адекватною моделлю, яка відповідає реальному процесу є така, що містить мінімальну кількість вільних параметрів статистичної моделі. Цей критерій виражається у вигляді:

$$HQIC(K) = -2\ln(L) + \ln(\ln(n))\,k. \tag{3.19}$$

Однак такий критерій не знайшов широкого застосування на практиці.

Аналогічно з попереднім випадком, використовуючи цей критерій для визначення оптимального порядку K марковської моделі поведінки програмного продукту, отримаємо співвідношення:

$$HQIC(K) = 2N^K(N-1)\ln(\ln(n)) - 2\sum_{i,j,\dots,k,l} n_{ij\dots kl}\ln p_{ij\dots kl}. \qquad (3.20)$$

На відміну від критеріїв, що базуються на мінімізації відстані Кульбака-Лейблера, і за допомогою яких вибирають з поміж заданої множини моделей ефективнішу (вважаючи, що "точної" моделі може і не бути в представленій множині моделей), за допомогою критеріїв Байєса вибирається з певною ймовірністю "правдива" модель. Ймовірність наближення до "точної" моделі прямує до одиниці, при збільшенні обсягу вибірки (тобто дані критерії передбачають наявність "точної" моделі серед множини моделей, що розглядається).

3.3. Узагальнений метод аналізу надійності програмних систем з урахуванням їх складності та етапів життєвого циклу

Як уже зазначалось, основною суперечністю на сучасному етапі розвитку технологій створення програмних систем та засобів аналізу їх надійності є невідповідність існуючих моделей та методів аналізу надійності ПС їх зростаючій складності. Тому основною задачею сучасного етапу розвитку методів аналізу надійності ПС є урахування в моделях надійності ПС їх складності. Крім того, інженерія надійності ПЗ (див. напр. [12, 18]), враховуючи, що виправлення помилок ПС на ранніх етапах життєвого циклу потребує значно менше ресурсів, вимагає як прогностичних моделей на ранніх етапах ЖЦ, так і моделей оцінювання надійності ПС на пізніх його етапах [9]. З урахуванням цих задач, та узагальнюючи розроблені в цій дисертаційній роботі моделі та методи аналізу та оцінювання надійності ПС, пропонується наступний узагальнений метод аналізу надійності ПС з урахуванням їх складності [165] на різних етапах ЖЦ (рис. 3.3).

Оскільки надійність лімітується внесеними помилками, а вартість виправлення помилки на ранніх етапах ЖЦ є меншою ніж на пізніших, задача оцінювання та аналізу надійності на ранніх етапах ЖЦ є важливою для індустрії створення програмних продуктів, розв'язання якої дасть можливість

оптимізувати витрати на створення ПС із заданими показниками якості. Крім того аналіз надійності на етапі проектування дозволить усунути архітектурні помилки, виправлення яких є особливо трудомістким, а в деяких випадках – неможливим.

Наведемо опис методу, зображеного на рис. 3.3 у відповідності з основними етапами ЖЦ ПС.

1. Етап аналізу вимог до ПЗ

Першим кроком узагальненого методу аналізу надійності ПС є отримання інформації про модель її поведінки, що включає оцінки імовірностей виконання різних сценаріїв використання ПС, на ранніх етапах ЖЦ.

Одним із перших етапів ЖЦ розробки ПЗ є збір та аналіз вимог. Так як стандартною практикою аналізу вимог до ПЗ є використання мови UML [195], а саме діаграми випадків використання, доцільним є використати цей інженерний засіб для аналізу надійності ПС, зокрема для наближеного отримання значень матриці ймовірностей переходів між модулями. Оскільки на етапі аналізу вимог наявний мінімальний обсяг інформації, і фактично відсутня архітектура ПС, в роботах [153, 155] описано використання методів системного аналізу для отримання чисельних значень параметрів майбутньої системи, зокрема відносної частоти використання певних сценаріїв, на основі яких можна оцінити значення матриці ймовірностей переходів між модулями ПС.

Якщо q_i – ймовірність використання ПС користувачем u_i , P_{ij} – ймовірність, що користувач u_i використовує функціональну можливість f_j, то ймовірність виконання випадку використання x обчислюється наступним чином: $P(x) = \sum_{i=1}^{m} q_i P_{ix}$, де m – кількість типів користувачів (акторів). На основі ймовірностей (або відносних частот) виконання усіх випадків використання можна оцінити ймовірності виконання модулів ПС та переходів між ними. Необхідно пам’ятати, що на цьому етапі ЖЦ можна тільки наближено говорити про модулі ПС, а згадані імовірності отримують апріорно на основі експертних оцінок.

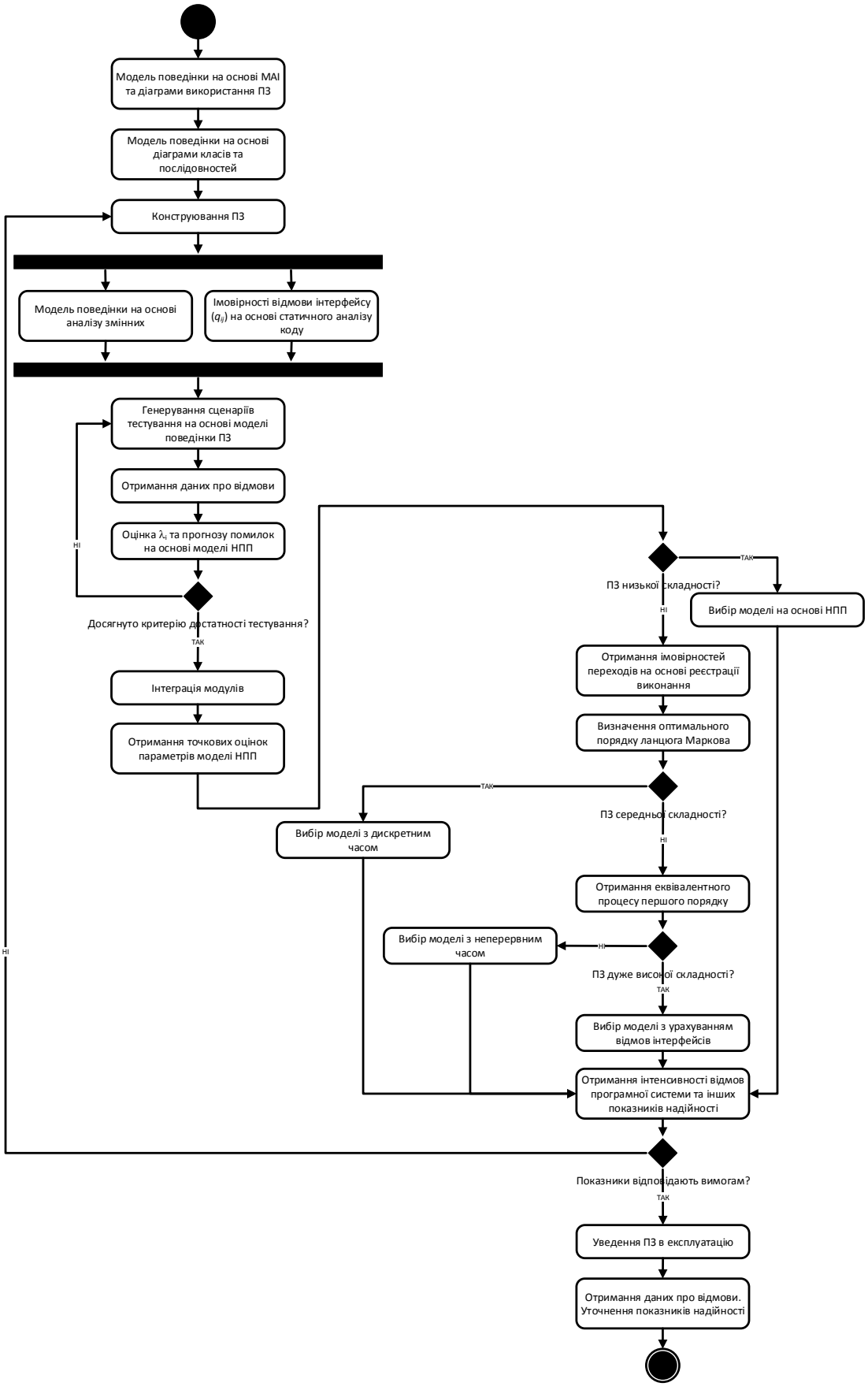


Рис. 3.3. Схема узагальненого методу аналізу надійності ПС з урахуванням їх складності впродовж різних етапів ЖЦ.

Підхід, описаний у [196] потребує знання чисельних значень ймовірностей використання системи кожним типом акторів, а також ймовірностей виконання кожного випадку використання типами користувачів. Проте, на цьому етапі ЖЦ розробник не має необхідних даних, а для замовника, який, як вже зазначалось, є основним експертом з випадків використання системи, задача оцінювання таких ймовірностей виявляється занадто складною. Для усунення даного обмеження в [153, 155] запропоновано використати метод аналізу ієрархій (MAI), який передбачає декомпозицію задачі та використання експертом попарних порівнянь [197].

Відповідно до [153] опитування замовника (експерта) здійснюється у вигляді анкетування за допомогою попарного порівняння згідно з шкалою відносної важливості методу аналізу ієрархій. Опитування передбачає попарне порівняння вузлів рівнів ієрархії, якими є актори системи та варіанти її використання. Під попарним порівнянням тут мають на увазі відносну частоту використання ПС певними акторами та відносну частоту виконання заданим актором певних сценаріїв використання. На основі анкетування заповнюється матриця попарних порівнянь, та здійснюється розрахунок ймовірностей виконання різних випадків використання. Для спрощення опитування замовника можливе використання модифікації MAI, яка базується на порівнянні за стандартами [197], де різним випадкам використання (чи історіям користувача) присвоюється певний пріоритет. Використання подібної системи пріоритетів є поширеною практикою в індустрії розробки ПЗ при побудові специфікацій ПС під час роботи з замовником.

2. Етап проектування архітектури ПЗ

Другий крок методу аналізу надійності (рис. 3.3) є отримання прогнозованого графу станів ПС, який відображає виконання модулів та передачу управління між ними, а також попередніх оцінок матриці імовірності переходів між станами на етапі проектування ПЗ. Оскільки етап проектування належить до ранніх етапів ЖЦ, на якому ще немає коду програми та, відповідно, інформації про відмови, в [155, 158] описано підхід з використанням

стандартного засобу програмної інженерії – універсальної мови моделювання UML.

При розробці архітектури об’єктно-орієнтованих систем використовується UML діаграма класів. За допомогою діаграми класів здійснюється опис модулів ПС, а також зв’язків між ними. З діаграми класів можна сформувати матрицю ймовірностей переходів між модулями, а саме її структуру, проте необхідно зауважити, що у цьому випадку неможливо отримати ймовірності переходів та тривалість перебування в модулях, оскільки така діаграма призначена лише для розробки архітектури ПС, а не відображення поведінки користувачів.

Для проектування модулів ПС та взаємодії між ними використовується UML діаграма послідовності дій. В цій роботі діаграма послідовності дій використовується для оцінки кількості переходів між модулями, з яких можна визначити матрицю ймовірності переходів між модулями, а також час зайнятості [196] – тривалість виконання модуля. Провівши аналіз розроблених UML діаграм послідовності дій під час проектування ПС необхідно порахувати сумарну кількість переходів з модуля в модуль і сформувати матрицю переходів між модулями. Для визначення часу перебування в кожному модулі ПС необхідно просумувати тривалості перебування в модулі. В якості модуля пропонується використовувати клас, який є основної складовою при розробці ПС відповідно до об’єктно-орієнтованого підходу та може бути незалежно протестований. При переході між модулями (класами) виконуються виклики відповідних методів класу, тривалість перебування в модулі визначається часом виконання методу. Для того щоб отримати матрицю ймовірностей переходів усієї ПС, необхідно подібним чином урахувати всі сценарії використання ПС, оскільки діаграма послідовності зазвичай будується для кожного сценарію використання, а імовірність виконання кожного сценарію отримують на першому кроці даного методу.

Крім об'єктно-орієнтованого підходу, для вирішення проблем розроблення, модернізації та супроводу сучасних складних ПС використовують аспектно-орієнтоване програмування (АОП) [203]. При правильному

використанні АОП дозволяє [203]: значно зменшити об’єм програмного коду, покращити дизайн та загальну модульність системи, спростити код системи, спростити тестування та супровід, збільшити кількість повторно використовуваних модулів.

У [203, 205] проведено порівняння ефективності застосування аспектно-орієнтованого підходу з об’єктно-орієнтованим за рядом метрик програмного коду. У [205] показано, що значення АОП метрик в цілому є на 10–40% нижчими ніж у випадку реалізації з використанням ООП. Аспектна реалізація є меншою за кількістю стрічок коду, тоді як кількість унікальних операторів та операндів сильно не відрізняються. Також у АОП нижча цикломатична складність, що свідчить про те, що аспектна реалізація є кращою з точки зору топологічної складності, а також при написанні аспектної реалізації витрачається менше зусиль. Аспектна реалізація має нижчий рівень зв’язності та складності методів, тому вона містить кращу функціональну декомпозицію. Таким чином в [205] зроблено висновок, що застосування АОП дозволяє покращити якість та розширюваність ПС, а нижча топологічна складність, менший об'єм коду, нижчий рівень складності коду та краща функціональна декомпозиція підвищують показники надійності ПС за рахунок зменшення її складності.

3. Етап конструювання ПЗ

На цьому етапі, під час якого створюється працездатна ПС з використанням методів верифікації, кодування та тестування модулів, метод, схема якого зображена на рис. 3.3, дає можливість отримати прогнозований перелік лімітуючих (в сенсі надійності) модулів ПС; оцінити імовірності відмови інтерфейсів ПС на основі статичного аналізу коду та його метрик (3.8) та побудувати модель поведінки ПС на основі аналізу змінних її коду [198].

На початку етапу конструювання ПС можна скористатись виразом (3.4) та отриманими на попередніх кроках методу оцінками імовірностей виконання модулів ПС для моделювання впливу інтенсивності відмов модулів на інтенсивність відмов ПС. При цьому можна отримати верхні оцінки інтенсивності відмов модулів та визначити модулі, внесок яких в загальну

надійність ПС буде найвагомішим, з метою підвищеної уваги до таких модулів під час написання їх програмного коду ти визначення стратегії їх тестування. На цьому кроці методу аналізу надійності недоцільно використовувати математичний апарат ЛМВП внаслідок постійних змін в ПС та відсутності уточнених значень параметрів моделі надійності.

Модель поведінки ПС – це модель програмної системи, що відображає процес її функціонування та деякий набір характеристик. Як правило, модель поведінки ПС представляється у вигляді зваженого графа, вершинами якого є модулі програми, а зваженими ребрами – ймовірності передавання управління між ними. Коректна модель поведінки ПС з урахуванням необхідних характеристик програм дає можливість точніше здійснювати АФСТ, а також оцінювати надійність ПС моделями, побудованими на основі архітектурного підходу. У [198] описано нову математичну модель функціонування ПС у вигляді орієнтованого графа $G = \{C, P\}$, де C – множина модулів ПС, P – множина переходів між відповідними модулями. Кожен вузол графа C_i є набором множин змінних та відповідних класів еквівалентності, які використовуються та змінюються у модулі, а також змінних та відповідних класів еквівалентності, що можуть викликати відмови у модулі C_i, а також характеризується списком інцидентних дуг, що в свою чергу містять ймовірності передавання контролю p_{ij} до іншого вузла C_j $\left(i, j = \overline{1, N}\right)$.

4. Фаза модульного тестування

Під час цієї фази ЖЦ ПЗ здійснюється тестування розроблених модулів (переважно розробниками) без інтеграції їх в систему та виправлення виявлених помилок. Слід зауважити, що розроблений узагальнений метод аналізу надійності ПС передбачає проведення усіх видів тестування [172, 173] на основі моделі поведінки, що базується на змінних коду [198].

Після АФСТ відповідно до стратегії тестування на основі моделі поведінки, отримуються дані про відмови ПС. З отриманих даних про відмови під час процесу відлагодження ПС (який характеризується зміною з часом

кількості помилок в ПС, а відповідно залежністю інтенсивності відмов від часу) з використанням моделі на основі НПП (2.1) отримують інтенсивність відмов модуля та прогнозовану кількість помилок згідно (2.3). Тривалість процесу модульного тестування визначають на основі критерію достатності процесу тестування [128] або ж удосконаленої процедури оцінки кількості помилок в ПС [199]. Як уже зазначалось якість процесу тестування забезпечується використанням методу АФСТ із забезпеченням метрики покриття змінних коду [172, 173]. Після прийняття рішення про припинення тестування і завершення поточної ітерації розробки програмного модуля, переходять до наступного кроку методу аналізу надійності.

5. Фаза інтеграційного тестування

На цьому кроці відбувається інтеграція модулів в ПС, проводиться її тестування вцілому з метою виявлення (і усунення) помилок взаємодії модулів, що передбачає виконання складних тестових сценаріїв, які стосуються декількох модулів системи. Для проведення такого тестування особливо ефективним є використання описаних в [172, 173] засобів АФСТ. Водночас, результати інтеграційного тестування дають інформацію про відмови програмної системи в цілому, яка може бути використана для оцінювання її надійності або для засобів підтримки прийняття рішень при розробленні ПЗ, описаних в четвертому розділі цієї дисертації.

Після отримання даних про відмови програмної системи та виправлення виявлених помилок інтеграції, метод (рис. 3.3) передбачає використання МНПС на основі НПП [66], як першого наближення при аналізі надійності програмної системи. Використання цієї моделі детально описано в методі оцінювання надійності ПС на основі моделі з показником складності в підрозділі 2.2 цієї дисертації. На цьому етапі для уточненої оцінки кількості помилок в програмному продукті метод аналізу надійності передбачає використання удосконаленої процедури оцінки кількості помилок [199]. Після отримання значень параметрів пуассонового процесу (2.1) здійснюють оцінювання складності ПС згідно [133]. При цьому, якщо програмна система відноситься до

класу нескладних ($s < 0{,}66$) рекомендується використовувати для аналізу її надійності модель на основі НПП і перейти до оцінювання показників її надійності. В іншому випадку слід використати одну з моделей на основі ланцюга Маркова вищого порядку, описаних вище в цьому розділі.

Для отримання значень імовірностей передачі потоку управління між модулями ПС, як зазначалось вище (див. рис. 3.1), можна використати моніторинг виконання програми [156] чи формалізм прихованих марковських моделей. Важливо пам'ятати, що при отриманні імовірностей переходів ПС між станами шляхом моніторингу виконання програми (яке здійснюється не в умовах реальної експлуатації, а під час тестування розробниками), сценарії виконання програми повинні бути максимально наближені до сценаріїв її подальшої експлуатації – наприклад до отриманих на першому кроці цього методу на основі методу аналізу ієрархій за результатами експертизи замовника.

Після отримання необхідних значень параметрів марковських моделей (матриці імовірностей переходів між станами, початкового вектору, інтенсивності відмов модулів тощо) наступним кроком методу аналізу надійності (рис. 3.3) на основі одного з інформаційних критеріїв визначають оптимальний порядок марковського процесу. Після цього, якщо ПС відноситься до систем середнього ступеня складності (значення показника складності моделі пуассонового процесу $0{,}66 \leq s < 1{,}6$, оптимальний порядок процесу не дуже високий, а імовірності відмови інтерфейсів системи, оцінені на третьому кроці цього методі згідно (3.8), є нехтувально малими), рекомендується використовувати для аналізу надійності ПС модель на основі ЛМВП з дискретним часом (див. вирази (3.1)–(3.3)) і перейти до оцінювання показників її надійності. Використання моделі з дискретним часом є доцільним внаслідок невеликих обчислювальних витрат на роботу з цією моделлю та незначного зменшення ступеня адекватності при роботі з ПС середнього ступеня складності. В іншому випадку слід використати одну з моделей на основі ЛМВП з неперервним часом.

При використанні моделей з неперервним часом наступним кроком методу (рис. 3.3) є отримання еквівалентного процесу першого порядку для побудови та розв'язку системи рівнянь Колмогорова–Чепмена (3.5). Розв'язки цієї системи рівнянь дають значення імовірностей перебування системи в станах (часові залежності імовірностей виконання модулів ПС), що, разом з отриманими під час модульного тестування значеннями інтенсивності відмов модулів, на основі виразу (3.4) дає можливість отримати функцію інтенсивності відмов ПС високої складності (значення показника складності моделі пуассонового процесу $1{,}6 \leq s < 2{,}28$, а імовірностями відмови інтерфейсів ПС, оціненими на третьому кроці цього методі згідно (3.8) можна знехтувати). В такому випадку переходять до оцінювання показників надійності ПС (рис. 3.3).

Якщо ж складність ПС є дуже високою (значення показника складності моделі пуассонового процесу $s > 2{,}28$, а імовірностями відмови інтерфейсів ПС, оціненими на третьому кроці цього методі згідно (3.8) неможна знехтувати), то використовують модель надійності ПС (3.7), та переходять до оцінювання показників її надійності.

Таким чином узагальнений метод аналізу надійності програмних систем з урахуванням їх складності впродовж різних етапів життєвого циклу, зображений на рис. 3.3 дає можливість отримати функцію інтенсивності відмов ПС на основі виразів (2.1), (3.1), (3.4), (3.7) залежно від її складності. Після отримання часової залежності інтенсивності відмов ПС, інші показники надійності можуть бути розраховані згідно відомих із загальної теорії надійності залежностей [181]. На основі отриманих значень показників надійності приймається рішення про введення цієї версії ПС в експлуатацію, чи продовження тестування та відлагодження (а в окремих випадках і повернення до наступної ітерації розробки модулів системи).

6. Етап експлуатації ПЗ

Після уведення ПС в експлуатацію продовжується збір статистики використання системи та даних про її відмови з використанням сучасних засобів

та інструментів інженерії ПЗ (див. напр. [163]). На основі отриманих даних, згідно обраної на етапі розробки ПС моделі надійності, здійснюється уточнення показників надійності та експлуатаційних показників ПАС, в складі якої експлуатується досліджувана ПС. Інформація про показники надійності та сценарії використання ПС, разом з новими вимогами (як функціональними, так і іншими) лягає в основу процесу розробки нової версії цієї ПС.

3.4. Верифікація компонентних моделей та методу оцінювання надійності програмних систем на основі даних тестування

Для визначення ефективності використання розроблених моделей оцінювання надійності ПС, які використовують теорію марковських процесів вищих порядків, із використанням системи "СОН ПЗ" [200] обчислено значення коефіцієнта готовності (імовірності працездатного стану ПС) для п'яти тестових програмних засобів, які складалися в середньому з 10 модулів [170], за допомогою моделі оцінювання надійності ПС першого та вищого порядку з дискретним часом (рис.3.4).

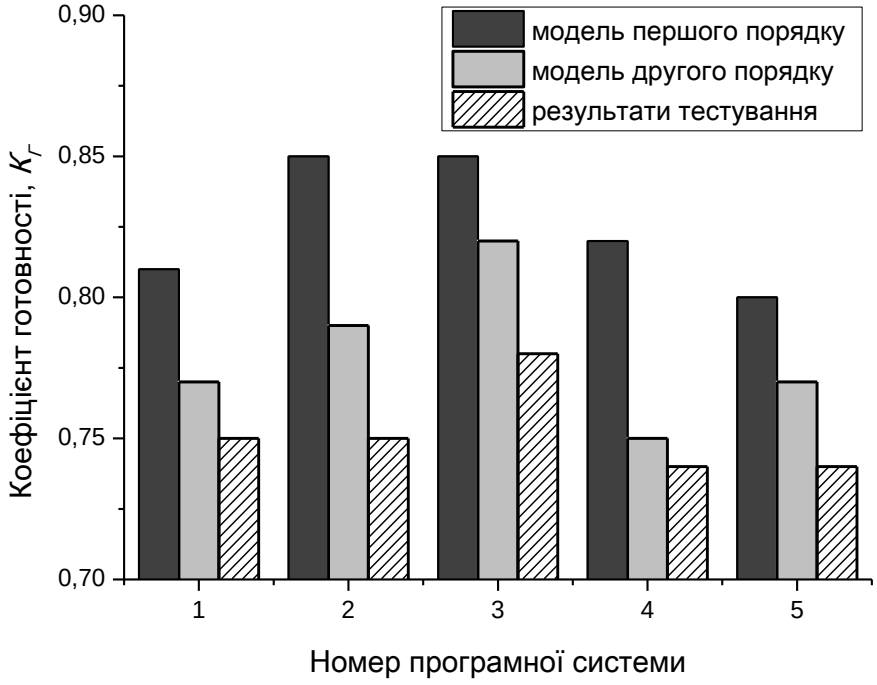


Рис. 3.4. Порівняння значення коефіцієнта готовності ПС, обчисленого з використанням моделей першого та вищого порядків з дискретним часом.

Як видно з рис. 3.4, використання моделі надійності на основі ланцюга Маркова першого порядку дає в усіх випадках завищене значення коефіцієнта готовності системи, що в найгіршому випадку (четверта система на рис. 3.4) дає переоцінку коефіцієнта готовності на 10%, що у випадку реальних ПС відповідального призначення може привести до недооцінки ризику експлуатації системи. Таким чином можна зробити висновок, що використання моделі оцінювання надійності ПС з урахуванням ЛМВП дає можливість навіть для ПС з невеликою кількістю модулів збільшити точність оцінки їх надійності на 6–10%.

Іншою тестовою ПС, що використовувалась для дослідження моделей надійності як з дискретним, так і з неперервним часом була ПС [170], що складалась з чотирьох модулів (класів): *Input, Calculation, Output, Exit*. Початковий граф станів цієї ПС зображено на рис. 3.5, де вершини означають модулі програми, а дуги зображають потік управління програми. Позначення вершин графу на рис. 3.5 відповідають першим літерам модулів програми (**I** – *Input*, **C** – *Calculation*, **O** – *Output*, **E** – *Exit*).

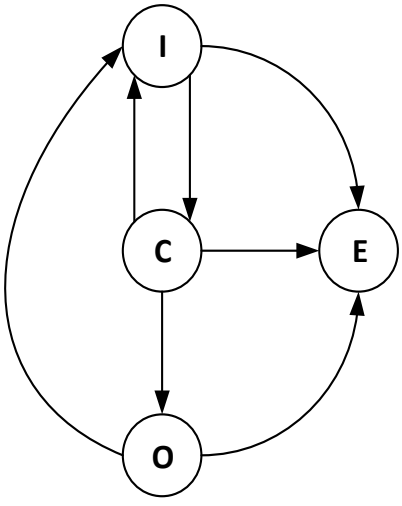


Рис. 3.5. Початковий граф, що відображає модулі та потік управління тестової ПС, що використовувалась для верифікації моделей надійності.

Розглянемо визначення імовірності безвідмовної роботи тестової ПС з використанням моделі надійності на основі ЛМВП з **дискретним часом**, відповідно до методу, описаного в [170].

Визначення інтенсивності відмов кожного модуля. Із використанням результатів тестування ПС, визначено інтенсивності відмов кожного модуля (табл.3.1).

Таблиця 3.1.

Інтенсивність відмов кожного модуля

Модулі	**Інтенсивність відмов, λ_i**
Input	0,11
Calculation	0,18
Output	0,09
Exit	0,01

Знаходження матриці оцінок ймовірностей переходів між модулями ПС та часу їхнього виконання. Дану задачу розв'язано шляхом моніторингу та реєстрації виконання ПС. Із використанням такого підходу отримано оцінки матриці ймовірностей передавання управління між модулями програмної системи та вектора початкового розподілу для випадку першого та другого порядків марковського процесу – табл. 3.2–3.4.

Таблиця 3.2.

Матриця оцінок ймовірностей першого порядку

Модулі	*Input*	*Calculation*	*Output*	*Exit*
Input	0,2987	0,66233	0	0,03897
Calculation	0,2666	0,05	0,6333	0,0501
Output	0,5	0	0,3148	0,1852
Exit	0	0	0	1

Таблиця 3.3.

Початковий вектор оцінок імовірностей першого порядку

Модулі	*Input*	*Calculation*	*Output*	*Exit*
Ймовірність	1	0	0	0

Таблиця 3.4.

Матриця оцінок ймовірностей другого порядку

Модулі	*Input*	*Calculation*	*Output*	*Exit*
Input(Input)	0,038	0,962	0	0
Input(Calculation)	0,16143	0,0806	0,6612	0,09677
Input(Output)	0	0	0	0
Input(Exit)	0	0	0	0
Calculation (Input)	0,4	0,53	0	0,07
Calculation (Calculation)	0,8	0	0,2	0
Calculation (Output)	0,2564	0	0,5	0,2436
Calculation (Exit)	0	0	0	0
Output (Input)	0,3571	0,5357	0	0,1072
Output (Calculation)	0	0	0	0
Output (Output)	0,8888	0	0,1112	0
Output (Exit)	0	0	0	0
Exit (Input)	0	0	0	0
Exit (Calculation)	0	0	0	0
Exit (Output)	0	0	0	0
Exit (Exit)	0	0	0	0

Таблиця 3.5.

Початковий вектор оцінок імовірностей другого порядку

Модулі	**Ймовірність**	**Модулі**	**Ймовірність**
Input(Input)	0,45	*Output (Input)*	0
Input(Calculation)	0,5	*Output (Calculation)*	0
Input(Output)	0	*Output (Output)*	0
Input(Exit)	0,05	*Output (Exit)*	0
Calculation (Input)	0	*Exit (Input)*	0
Calculation (Calculation)	0	*Exit (Calculation)*	0
Calculation (Output)	0	*Exit (Output)*	0
Calculation (Exit)	0	*Exit (Exit)*	0

Визначення порядку марковського процесу. Для визначення матриці ймовірностей передавання управління між модулями було використано критерій AIC (оскільки обсяг вибірки становив 188 даних). Отримані результати наведені у таблиці 3.6.

Таблиця 3.6.

Значення критерію AIC

Порядок процесу	**Значення критерію AIC**
1	223,7
2	203,5
3	352,7
4	1082,8

Як видно з таблиці 3.6, найменше значення критерію AIC відповідає другому порядку марковського процесу, що і є оптимальним порядком ланцюга Маркова для досліджуваної ПС.

Визначення очікуваної кількості виконань модулів ПС. Із урахуванням всіх отриманих даних та порядку марковського процесу обчислюємо за формулою

(3.3) очікувану кількість виконань кожного модуля в залежності від виконання двох попередніх модулів (табл. 3.7).

Таблиця 3.7.

Очікувана кількість виконань модулів ПС

Послідовність модулів	**Очікувана кількість виконань послідовності**
Input(Input)	6
Input(Calculation)	11
Input(Output)	1
Input(Exit)	1
Calculation (Input)	1
Calculation (Calculation)	4
Calculation (Output)	8
Calculation (Exit)	2
Output (Input)	2
Output (Calculation)	0
Output (Output)	0
Output (Exit)	4
Exit (Input)	0
Exit (Calculation)	0
Exit (Output)	0
Exit (Exit)	1

Визначення імовірності безвідмовної роботи кожного модуля. Маючи усі необхідні дані обчислено ймовірність безвідмовної роботи кожного модуля за формулою (3.2), які зведені в табл. 3.8.

Таблиця 3.8.

Ймовірність безвідмовної роботи кожного модуля

Модулі	**Ймовірності безвідмовної роботи**
Input	0,93
Calculation	0,86
Output	0,94
Exit	0,98

Визначення ймовірності безвідмовної роботи ПС в цілому. У попередніх кроках отримано усі необхідні параметри для визначення ймовірності безвідмовної роботи програмної системи згідно (3.1). Ймовірність безвідмовної роботи ПС в цілому, згідно моделі надійності на основі ланцюга Маркова з дискретним часом, в даному випадку становитиме $R = 0{,}74$.

Розглянемо визначення показників надійності тестової ПС з використанням моделі надійності на основі ЛМВП з **неперервним часом.**

Відповідно до процедури, описаної в розділі 3.1.2, для ПС, структура якої описується графом рис. 3.5 та імовірностями переходів табл. 3.2, 3.4, з урахуванням значень критерію AIC (табл. 3.6) побудовано еквівалентний до процесу другого порядку процес першого порядку, який зображено на рис. 3.6. На цьому рисунку позначення вершин графа відповідають рис. 3.5, а індекси означають попередній стан, з якого здійснено перехід в поточний стан (так, стан $\mathbf{O}_{\mathbf{C}}$ означає, що в поточний стан *Output* було передано управління зі стану *Calculation*).

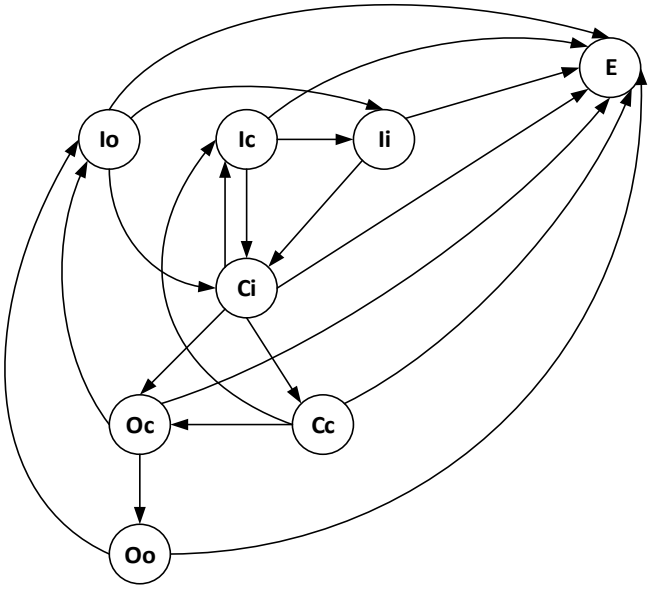


Рис. 3.6. Граф, що представляє еквівалентний процес до ланцюга Маркова другого порядку для ПС, початковий граф якої зображено на рис. 3.5.

З використанням значень табл. 3.2 та 3.4 було побудовано та розв'язано систему рівнянь Колмогорова–Чепмена (3.5) для процесу першого та другого порядку, та обчислено інтенсивність відмов ПС згідно (3.4).

На рис. 3.7 наведено часову залежність інтенсивності відмов тестової ПС, отриману на основі ланцюгів Маркова першого та другого порядку з неперервним часом. Як видно з цього рисунку при малих значеннях часу модель першого порядку дає дещо (на 1–5%) занижене значення інтенсивності відмов, в середньому діапазоні значень часу модель першого порядку дає суттєво (до 20%) завищене значення інтенсивності відмов, в той час як при зростанні значення t більше ніж 60 год. відмінність між значеннями $\lambda(t)$, отриманими на основі обох моделей практично прямує до нуля. Зменшення до нуля відмінностей результатів, отриманих на основі моделей першого та другого порядку, може бути пояснено як зменшенням самого значення $\lambda(t)$, так і відсутністю відмінностей в описі поведінки ПС двома моделями при $t \to \infty$ – в обох моделях передбачається, що в цьому випадку ПС перебуває в стані "*Exit*" і, відповідно, її інтенсивність відмов лімітується інтенсивністю відмов цього модуля. Відмінності ж поведінки залежності $\lambda(t)$ на початкових стадіях еволюції ПС

цілком пояснюються відмінностями в описі поведінки ПС моделями різного порядку, де зокрема модель першого порядку не враховує взаємозалежність виконання модулів, а відповідно відрізняються імовірності перебування в певному стані в момент часу t. Таким чином, можна стверджувати, що модель надійності ПС на основі ЛМВП більш адекватно описує поведінку ПС, та дозволяє більш точно визначати показники її надійності.

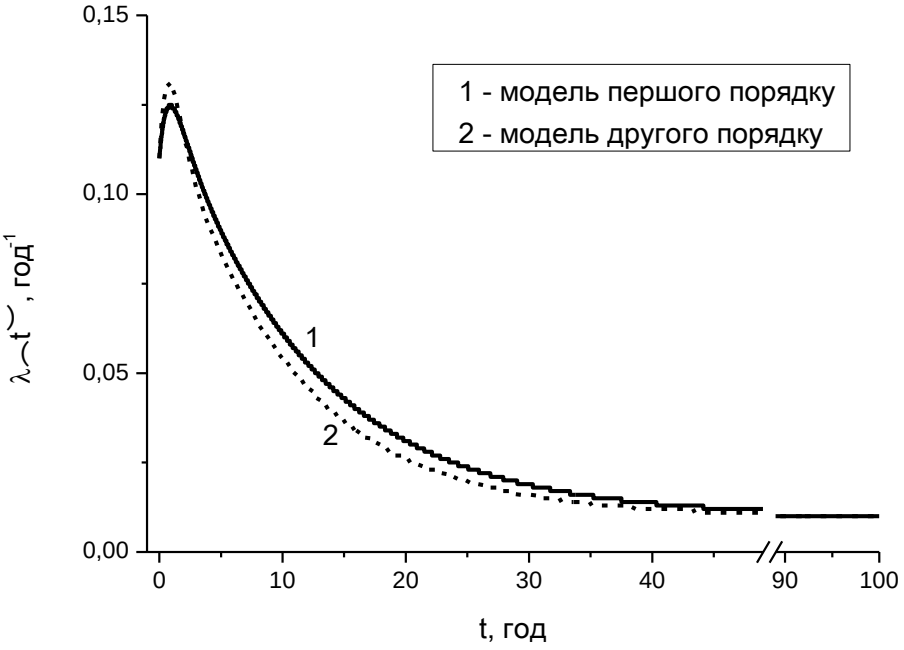


Рис. 3.7. Залежність інтенсивності відмов тестової ПС від часу, отримана на основі моделі першого (1) та другого (2) порядку з неперервним часом.

Такий висновок підтверджується і в результаті аналізу залежності імовірності безвідмовної роботи ПС від часу, зображеної на рис. 3.8 та отриманої з наступного співвідношення [181]:

$$P(t) = \exp\left(-\int_0^t \lambda(\tau)d\tau\right).$$

В даному випадку різниця в значеннях $P(t)$ зростає до ~20% зі зростанням значення часу. Такий результат легко зрозуміти, якщо взяти до уваги, що імовірність безвідмовної роботи це інтервальна оцінка на інтервалі $(0, t]$, і її похибка логічно зростає зі зростанням довжини інтервалу. Отже, неврахування

взаємозалежності виконання модулів може призвести до зростання похибки у визначенні показників надійності ПС до 20%.

Ще одним показником надійності, який використовувався для визначення ефективності та адекватності моделі надійності ПС вищого порядку був середня тривалість безвідмовної роботи:

$$T_1 = \int_0^{\infty} P(\tau)d\tau.$$

Для моделі першого порядку значення середнього часу безвідмовної роботи становить 20,9 год., в той час як значення середнього часу безвідмовної роботи, отримане на основі моделі другого порядку становить 23,3 год. Як бачимо різниця порівняно з результатами, отриманими на основі моделі першого порядку становить 11,5%, що є суттєвим при аналізі надійності складних технічних систем.

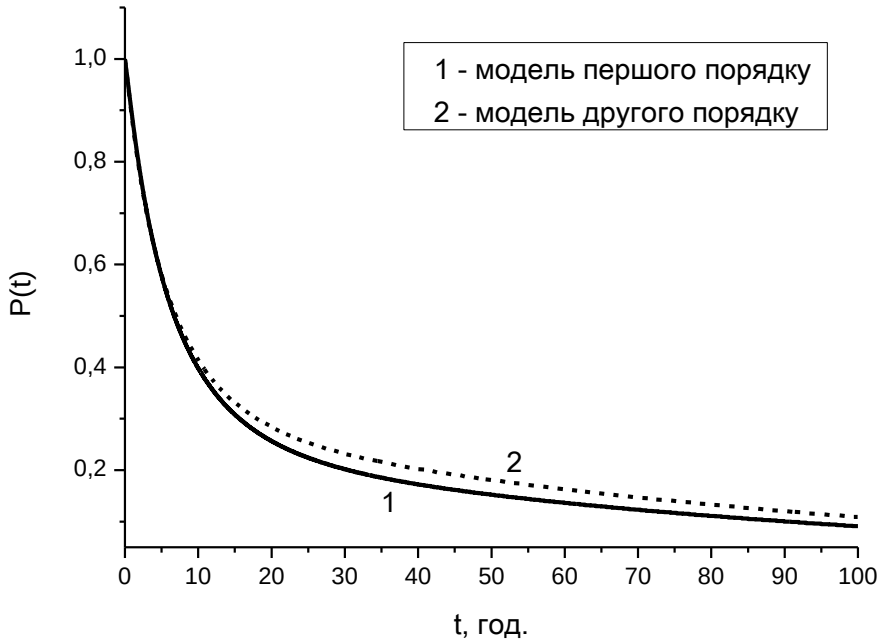


Рис. 3.8. Залежність імовірності безвідмовної роботи тестової ПС від часу, отримана на основі моделі першого (1) та другого (2) порядку з неперервним часом.

Таким чином, використання моделей надійності на основі ЛМВП вже для ПС з невеликою кількістю модулів (в даному випадку – 4) та складністю (в

даному випадку оптимальний порядок моделі – 2) приводить до підвищення ступеня адекватності моделі та збільшення точності оцінки показників надійності ПС на 10–20%. Значення такого підвищення точності оцінки показників надійності особливо важливе у випадку складних ПАС, в яких похибка визначення показників надійності програмної складової може багатократно впливати на похибку визначення показників надійності ПАС в цілому.

3.5. Висновки до розділу 3

Розроблено та описано удосконалені математичні моделі оцінювання надійності ПС з дискретним та неперервним часом, які завдяки використанню ЛМВП дають змогу врахувати взаємозалежність станів виконання модулів ПС, що підвищує адекватність цих моделей у випадку багатокомпонентної ПС зі складними сценаріями виконання. Удосконалено модель надійності Літлвуда та адаптовано її для процесу вищого порядку, а також наведено рекомендації щодо визначення імовірності відмови інтерфейсів програмних модулів.

Розроблено засоби попереднього оцінювання параметрів моделей надійності ПС на основі ЛМВП на етапі проектування ПЗ. Для вирішення цієї задачі запропоновано використовувати такі UML діаграми: випадків використання, класів та послідовності дій. Використання цих діаграм дає можливість оцінити імовірність виконання кожного сценарію використання ПС, отримати структуру марковської моделі на основі діаграми класів, та отримати початкову оцінку значень параметрів моделі на основі діаграми діяльності для кожного сценарію використання. Розроблені засобі дають можливість оцінити значення елементів матриці ймовірностей переходів між модулями ПС та тривалість перебування в кожному модулі. Це дає можливість визначати та уточнювати параметри моделі надійності ПС на ранніх етапах її ЖЦ, що повинно скоротити витрати на розробку ПС із заданим ступенем надійності.

Розроблено метод подання ЛМВП через еквівалентний процес першого порядку, який представляє матрицю інтенсивностей переходів вищого порядку

у вигляді еквівалентної матриці першого порядку з мінімальною кількістю нульових елементів. Перевагою запропонованого методу є простота, можливість застосування до марковських процесів будь-якого порядку (в тому числі змінного) та безпосереднє використання отриманої матриці інтенсивностей переходів для побудови системи рівнянь Колмогорова–Чемпена.

На основі побудованих моделей розроблено узагальнений метод аналізу надійності ПС з урахуванням їх складності впродовж різних етапів ЖЦ. Розроблений метод включає модель надійності ПЗ з показником складності на основі НПП, моделі надійності ПС на основі ЛМВП з дискретним та неперервним часом, а також модель на основі ЛМВП з неперервним часом, яка враховує відмови інтерфейсів програмних модулів. Показано використання цих моделей та допоміжних процедур на різних етапах ЖЦ в залежності від складності ПС. Узагальнений метод аналізу надійності ПС з урахуванням їх складності впродовж різних етапів ЖЦ дає можливість отримати функцію інтенсивності відмов та інші показники надійності ПС.

Наведено результати верифікації розроблених моделей та методів аналізу надійності ПС на основі результатів тестування ПС. Показано, що використання ЛМВП з дискретним часом дає змогу збільшити точність оцінювання показників надійності ПС на 6–10 %, а модель на основі процесу вищого порядку з неперервним часом – на 10–20%. Перевагами використання моделі з дискретним часом у задачах оцінювання надійності є невеликі обчислювальні ресурси, а перевагами використання ЛМВП з неперервним часом є більший ступінь адекватності за рахунок зняття обмеження на час передачі управління між модулями, можливість урахування залежності інтенсивності відмов модулів від часу в загальній інтенсивності відмов ПС, незалежність від кроку дискретизації (що у випадку дискретного часу може бути причиною неточного обчислення матриці ймовірностей переходів), а також автоматичне врахування переходів p_{ii}, що дає змогу уникнути зайвих обчислень.

РОЗДІЛ 4. МОДЕЛІ І МЕТОДИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ В ПРОЦЕСІ РОЗРОБЛЕННЯ ПРОГРАМНИХ СИСТЕМ З УРАХУВАННЯМ ВИМОГ ДО ЇХ НАДІЙНОСТІ

Розділ присвячено засобам підвищення ефективності розробки ПС з урахуванням вимог до його надійності. Зокрема описано критерій достатності процесу тестування та удосконалену процедуру прогнозування кількості помилок ПС, які дають можливість на етапі тестування ПС більш обґрунтовано приймати рішення стосовно досягнення заданих показників надійності та розподілу ресурсів проекту з розробки ПС. Описано нову модель функціонування ПС, яка враховує її архітектуру та змінні програмного коду, та методи автоматизованого формування сценаріїв тестування, які дають можливість підвищити ефективність процесу тестування ПС. Ця модель забезпечує підвищення достовірності оцінювання показників надійності ПС за рахунок більш повного покриття змінних коду та їх класів еквівалентності тестовими сценаріями. Наведено рекомендації стосовно вибору стратегії тестування при розробці ПС в залежності від її складності.

Основні результати і висновки, викладені в цьому розділі, опубліковані в працях [128, 172, 173, 198, 199, 201–207].

4.1. Обґрунтування критерію достатності процесу тестування програмних систем

Важливим прикладним аспектом моделей надійності ПС може стати встановлення кількісного критерію достатності процесу тестування, який би надав можливість керівникам програмних проектів більш обґрунтовано приймати рішення про виділення ресурсів на тестування та про завершення цього етапу розробки ПС. Так, на даний час, у переважній більшості ІТ компаній такий критерій носить скоріше якісний та неформалізований характер на зразок "задоволення замовника", який жодним чином не можна використовувати, наприклад, при розробці ПС відповідального призначення. В даному розділі

наведено опис та формалізацію критерію достатності процесу тестування ПС [128], побудованого на основі МНПС [66].

Для визначення критерію достатності процесу тестування побудовані залежності параметрів трьох досліджуваних моделей надійності в залежності від кількості ітерацій, на яких було припинено тестування (рис. 4.1, 4.2, 4.3) для емпіричних даних першого експерименту з роботи [90] (табл. А.1). На рис. 4.1, 4.2 криві 1 зображають відповідні параметри моделі з показником складності, криві 2 – параметри моделі Goel–Okumoto, криві 3 – параметри S-подібної моделі. Побудова таких залежностей відповідає практиці проведення тестування, коли здійснюється певна ітерація (цикл тестування) після якої оцінюється якість ПС. Потрібно після кожного такого циклу описувати отримані результати моделлю надійності, а параметри моделі зображати графічно чи в таблиці.

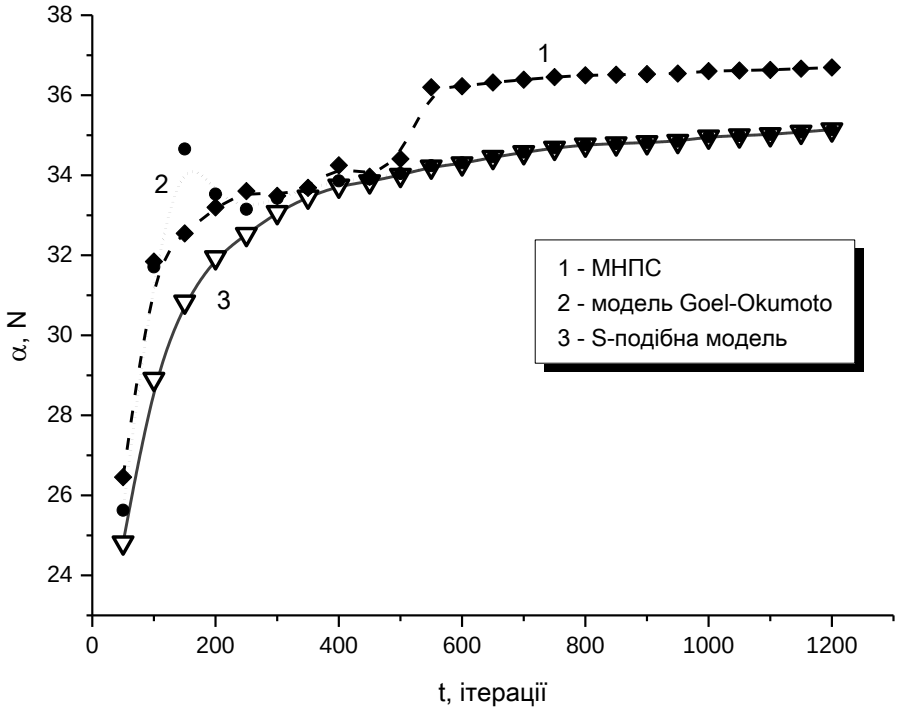


Рис. 4.1. Залежність параметра, що визначає загальну кількість помилок (α та N відповідно) для моделей надійності, від кількості ітерацій, на яких було припинено тестування.

Як показано на рис. 4.1 параметр α для моделі з показником складності та S-подібної моделі чи N для моделі Goel–Okumoto не дає змоги визначити зазначений критерій достатності процесу тестування, оскільки його залежності від кількості ітерацій тестування є практично гладкими функціями без особливостей. Особливість залежності параметру α для моделі з показником складності (рис. 4.1, крива 1) в околі 550 ітерацій добре співвідноситься з висновками, які зроблені на основі аналізу результатів роботи [90] – після 500–600 ітерацій кількість виявлених помилок в цій ПС розподілена за законом Пуассона, тоді як при меншій кількості ітерацій перевірка такої гіпотези дає негативний результат.

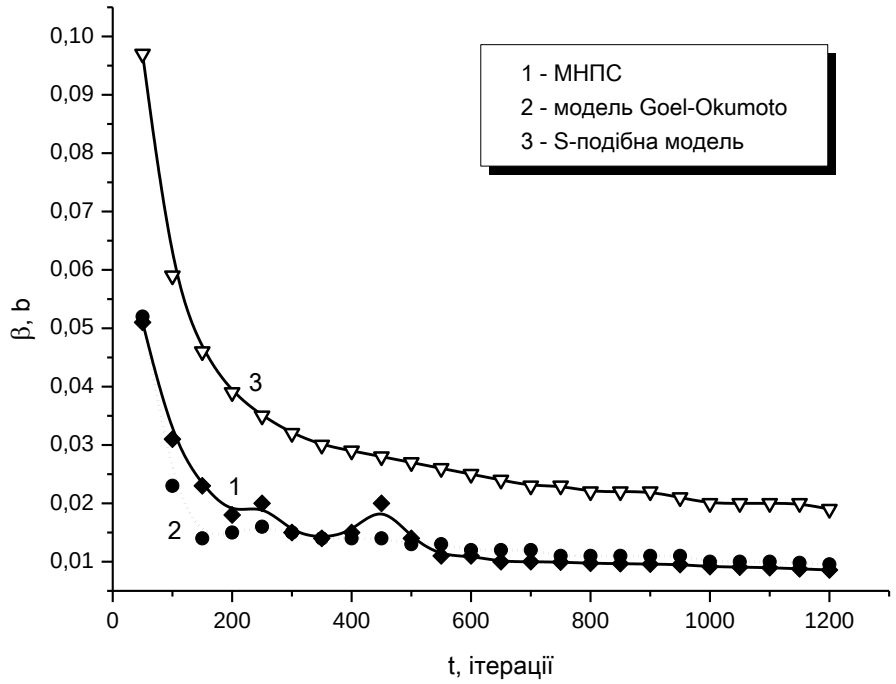


Рис. 4.2. Залежність параметра, що визначає тривалість процесу виявлення помилок (β та b відповідно) для моделей надійності від ітерацій, на яких було припинено тестування.

Отже, МНПС [66], крім іншого, дає можливість встановити точку переходу до пуассонового розподілу кількості виявлених помилок і, таким чином, встановити область експериментальних результатів, які можуть з високим

ступенем достовірності бути описаними моделлю надійності на основі такого розподілу.

Рис. 4.2 показує, що параметр β для моделі з показником складності проекту та S-подібної чи b для моделі Goel–Okumoto так само не може бути основою для визначення критерію достатності процесу тестування, оскільки його залежності від кількості ітерацій тестування є практично гладкими функціями без особливостей. В цьому випадку практично відсутня навіть будь-яка особливість залежності параметра β для запропонованої моделі в околі 550 ітерацій на відміну від параметра α (рис. 4.1). Хоча слід зазначити, що при більших значеннях часу крива є більш гладкою і не виявляє ознак осциляцій, характерних для значень часу менших за 500 ітерацій (рис. 4.2, крива 1).

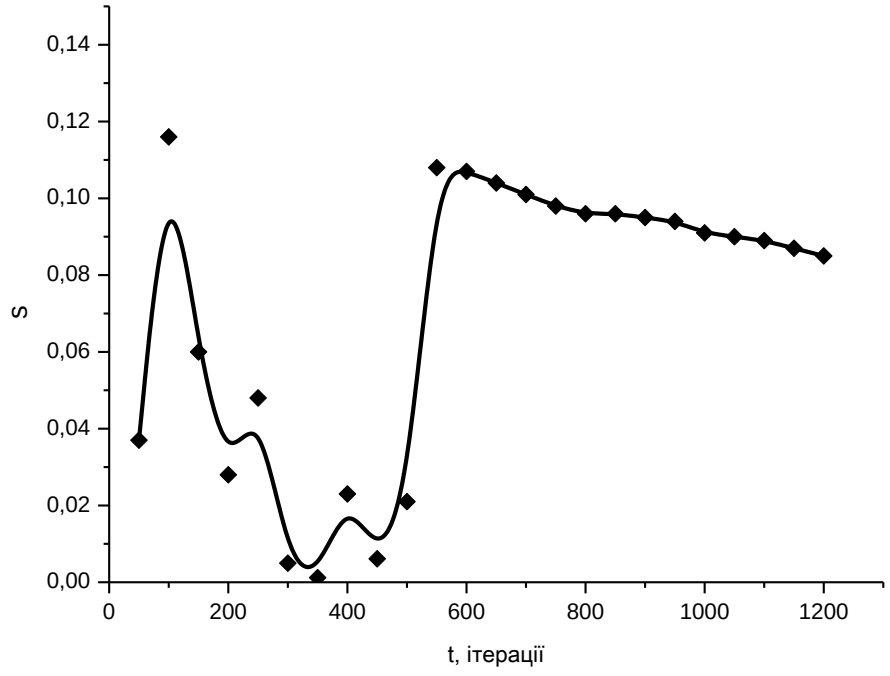


Рис. 4.3. Залежність показника складності ПС s від ітерацій, на яких було припинено тестування.

На противагу до параметрів α та β, залежність показника складності ПС s (який відсутній в усіх інших моделях на основі розподілу Пуассона) виявляє чітку особливість, що покладена в основу критерію достатності процесу

тестування (рис. 4.3). Ця особливість полягає в тому, що при $t \geq 550$ (що знову таки відповідає переходу до пуассонового розподілу кількості виявлених помилок) залежність $s(t)$ стає гладкою, а значення s наближається до постійної величини, на відміну від такої залежності при $t < 550$, яка проявляє різко осцилюючий характер.

Така поведінка залежності $s(t)$ пояснюється тим, що показник складності, який представляє складність ПС в моделі надійності (параметр s) є основним параметром, що визначає форму і функцію розподілу, а відповідно і густину розподілу імовірностей випадкової величини, яка у нашому випадку визначається кількістю відмов.

Отже на пізніх етапах тестування ПС, коли взаємозалежні помилки виявлені та усунені, а кількість відмов відповідає пуассоновому розподілу, оцінка параметра складності ПС s в моделі надійності вже практично не змінюється, а змінюються в основному кількісні характеристики (параметри α та β), що дає можливість формалізувати критерій достатності процесу тестування ПС таким чином [128]:

$$\frac{ds(t)}{dt} \underset{t\to\infty}{\longrightarrow} 0. \tag{4.1}$$

Ще раз необхідно зауважити, що усі інші відомі моделі надійності ПС на основі НПП не дають можливості отримати такий критерій внаслідок відсутності необхідних параметрів моделей, які б описували якісну зміну форми розподілу.

Залежність функції (4.1), отриману шляхом чисельного диференціювання даних з рис. 4.3, наведено на рис. 4.4. Така залежність наочно ілюструє критерій достатності процесу тестування ПС (4.1) і показує різку зміну форми кривої при переході експериментальних даних до пуассонового розподілу.

Для ілюстрації та аналізу критерію достатності процесу тестування були побудовані залежності параметрів трьох досліджуваних моделей надійності в залежності від ітерацій, на яких було припинено тестування (рис. 4.5, 4.6, 4.7) також для результатів другого експерименту з роботи [90] (табл. А.2). Як уже зазначалось вище, побудова таких залежностей відповідає практиці проведення

тестування, коли здійснюється ітерація (цикл тестування), після якої оцінюється якість програмного продукту. У [128] пропонується після кожного такого циклу описувати отримані результати моделлю надійності, та визначати досягнення критерію достатності процесу тестування, який використовувати як важливий параметр при прийнятті рішень стосовно проекту.

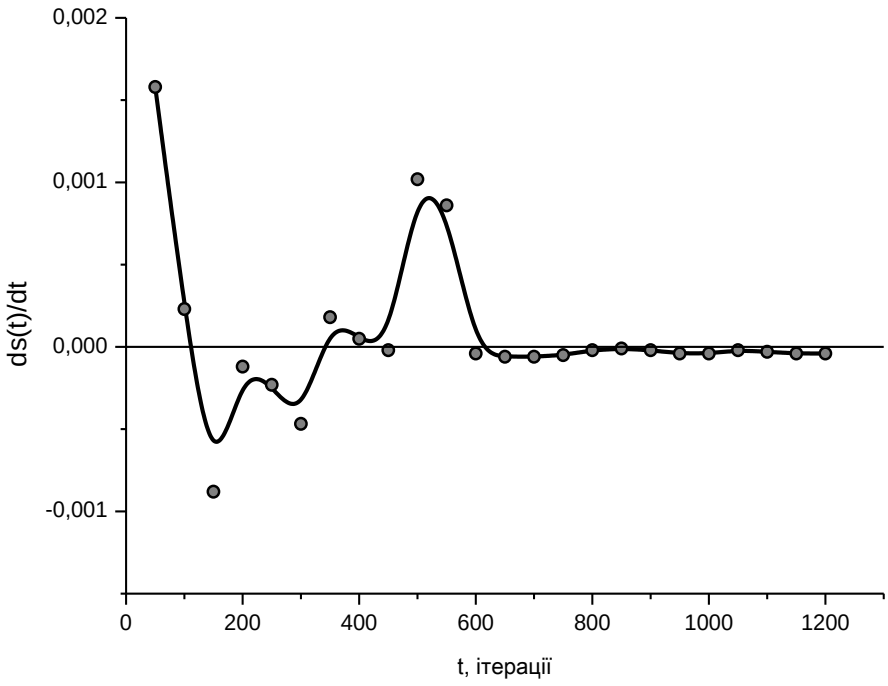


Рис. 4.4. Залежність критерію достатності процесу тестування ПС від кількості ітерацій, на яких було припинено тестування

На рис. 4.5 зображено залежність параметрів α і s моделі з показником складності від тривалості процесу тестування. Особливість залежності параметру α моделі з показником складності (на відміну від інших моделей – рис. 4.7) в діапазоні 450–550 ітерацій відповідає висновкам, які можна зробити з аналізу результатів роботи [90] – після 500 ітерацій кількість відмов реальної ПС розподілена за Пуассоном, тоді як в області менших значень кількості відмов перевірка такої гіпотези дає негативний результат. Отже залежність параметру α моделі з показником складності, крім іншого, дозволяє встановити точку переходу до пуассонового розподілу кількості відмов. Так само, як і в роботі

[128] залежність показника складності продукту від тривалості процесу тестування виявляє чітку особливість, яку було покладено в основу критерію достатності процесу тестування. Ця особливість полягає в тому, що при $t \geq 550$ (що знову таки відповідає переходу до пуассонового розподілу відмов) значення s наближається до постійної величини, на відміну від такої залежності при $t < 500$. Залежність критерію достатності процесу тестування (ds/dt), отриману шляхом чисельного диференціювання даних з рис. 4.5, наведено на рис. 4.6 (б). Як бачимо, така залежність наочно ілюструє критерій достатності процесу тестування ПС [128] (4.1) і показує наближення цього критерію до нуля при переході експериментальних даних до пуассонового розподілу.

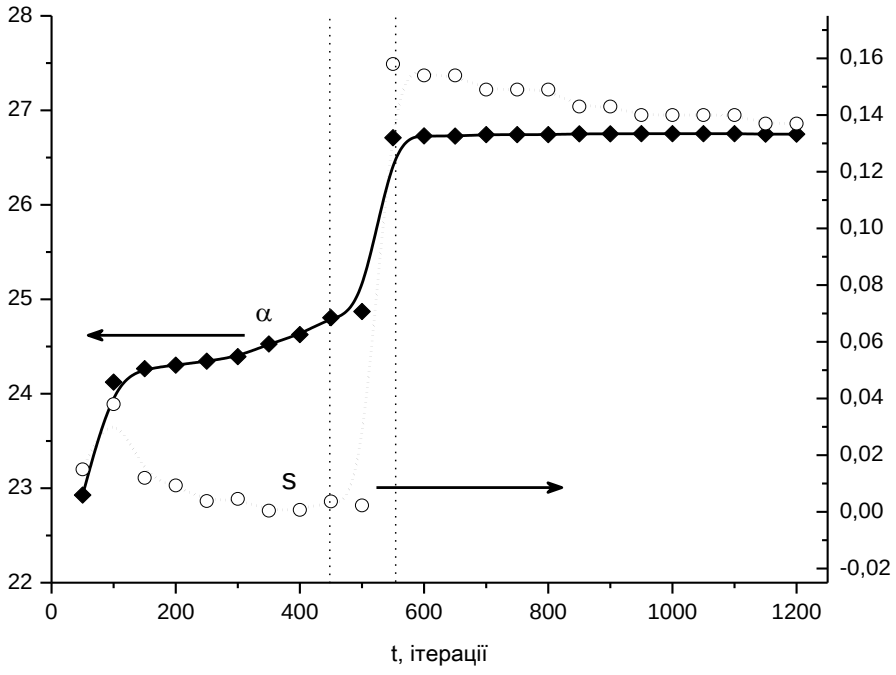
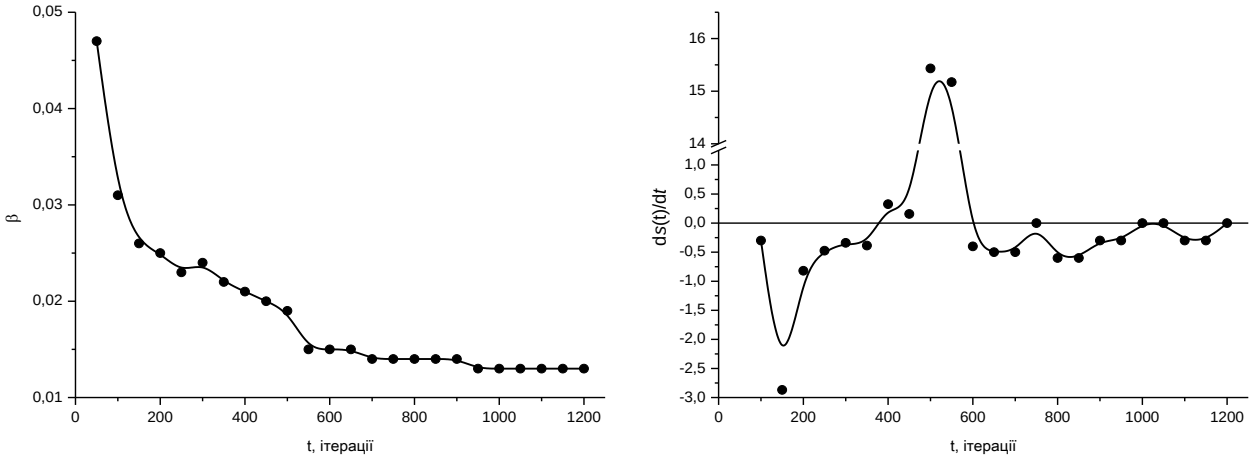


Рис. 4.5. Залежність параметрів α і s МНПС від тривалості процесу тестування.

На рис. 4.6 зображено залежність параметра β (а) та критерію достатності процесу тестування (б) МНПС від тривалості процесу тестування (номеру ітерації, на якій було припинено тестування). Так само, як і в роботі [128] для інших експериментальних даних, параметр β МНПС не може бути основою для визначення критерію достатності процесу тестування, оскільки його залежність від ітерацій тестування є практично гладкою функцією без особливостей. В

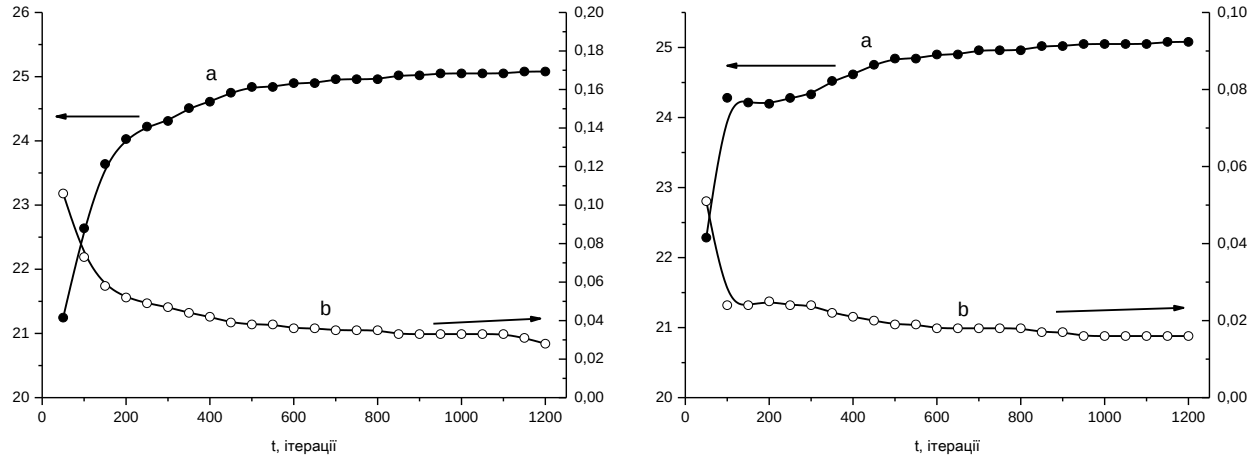
цьому випадку також практично відсутня навіть будь-яка особливість залежності параметру β для запропонованої моделі в околі 500 ітерацій на відміну від параметру α (рис. 4.5). З рис. 4.5 та рис. 4.6 видно, що на пізніх етапах тестування ПЗ, коли взаємозалежні помилки виявлені та усунені, а кількість відмов відповідає пуассоновому розподілу, параметр s практично не змінюється, а змінюються в основному кількісні характеристики (параметри α та β).



а). *б*).

Рис. 4.6. Залежність параметра β (*а*) і критерію достатності процесу тестування (*б*) МНПС від тривалості процесу тестування.

На противагу моделі з динамічним показником складності, S-подібна модель та модель Goel–Okumoto, залежності параметрів яких від тривалості процесу тестування наведені на рис. 4.7 (а) і (б) відповідно, не виявляють характерних особливостей, які можна було в використати в ролі критерію достатності процесу тестування чи для визначення точки переходу до пуассонового розподілу вхідних величин.



а). *б*).

Рис. 4.7. Залежність параметрів S-подібної моделі (*а*) та моделі Goel–Okumoto (*б*) від тривалості процесу тестування.

Як показано на рис. 4.7 залежності усіх параметрів обох моделей від тривалості процесу тестування є гладкими кривими, при цьому залежності параметрів моделі Goel–Okumoto проявляють стрибок при $t = 100$ (рис. 4.7 (б)) на відміну від параметрів S-подібної моделі (рис. 4.7 (а)), залежність яких є плавною в усьому дослідженому діапазоні.

Таким чином з використанням критерію достатності процесу тестування можна визначити загальну кількість помилок в ПС за допомогою рівняння (4.1) і, порівнявши її з кількістю вже виявлених та виправлених помилок, прийняти обґрунтоване рішення про розподіл ресурсів проекту.

4.2. Удосконалена процедура прогнозування кількості помилок ПС

Критерій достатності процесу тестування [128], отриманий з узагальненої МНПС на основі НПП [66] дає можливість надати практичні рекомендації керівникам проекту стосовно розподілу виробничих ресурсів між етапами життєвого циклу ПС, а також, після досягнення цього критерію, зі значним ступенем достовірності оцінити кількість невиявлених помилок в ПС (рівняння (2.3)). Значення кількості невиявлених помилок в ПС може бути використано як окремий критерій для прийняття рішень в процесі розробки ПС, так і, наприклад, в моделях вартості ПС, розглянутих в підрозділі 1.6. З метою підвищення

ступеня точності оцінки кількості невиявлених помилок в ПС та/або можливості отримати такі оцінки на більш ранніх стадіях тестування, в [199, 207] було описано удосконалену процедуру підтримки прийняття рішення стосовно досягнення заданого рівня надійності ПС за рахунок підвищення точності прогнозу кількості помилок на ранніх етапах тестування.

Для побудови удосконаленої процедури підтримки прийняття рішення стосовно досягнення заданого рівня надійності ПС [199, 207], з метою отримання інформації про поведінку прогнозованої загальної кількості помилок в ПС (див. рівняння (2.3)) було досліджено її залежність від тривалості етапу тестування. Для цього здійснено нелінійний регресійний аналіз критеріальної змінної μ_∞, отриманої з рівняння (2.3) відносно предикторів T_i , які означають час припинення тестування, а точкові оцінки параметрів моделі $\hat{\alpha}, \hat{\beta}, \hat{s}$, використані в рівнянні (2.3) для отримання відповідного значення $\mu_\infty = f(T_i)$, отримували методом максимальної правдоподібності з системи (2.16) на інтервалі $(0; T_i]$. В якості рівняння регресії було обрано вираз Вейбулівського типу:

$$\mu_\infty = A\left[1 - \exp\left(-\left(k(T_i - T_c)\right)^d\right)\right]. \qquad (4.2)$$

З виразу (4.2) можна отримати прогнозоване значення кількості відмов (помилок) ПС за припущення, що тривалість процесу тестування буде безмежною ($T_i \to \infty$). Як видно з (4.2) це значення дорівнює значенню параметра регресії A. Таким чином, отримавши методом найменших квадратів параметри рівняння регресії, за значенням параметру A можна отримати уточнений прогноз граничної кількості відмов ПС.

Графік залежності $\mu_\infty = f(T_i)$ для обох експериментів, описаних в [90] (див. розділ 2.4.1) та відповідні лінії регресії наведено на рис. 4.8. Як видно з цього рисунку розкид значень прогнозу є більшим для першого експерименту, тоді як у випадку другого експерименту прогноз є практично стабільним починаючи приблизно з 200 ітерацій тестування. Величина зміни прогнозу зі збільшенням тривалості процесу тестування, імовірно, залежить від складності ПС, однак остаточне вияснення цього питання потребує подальших досліджень.

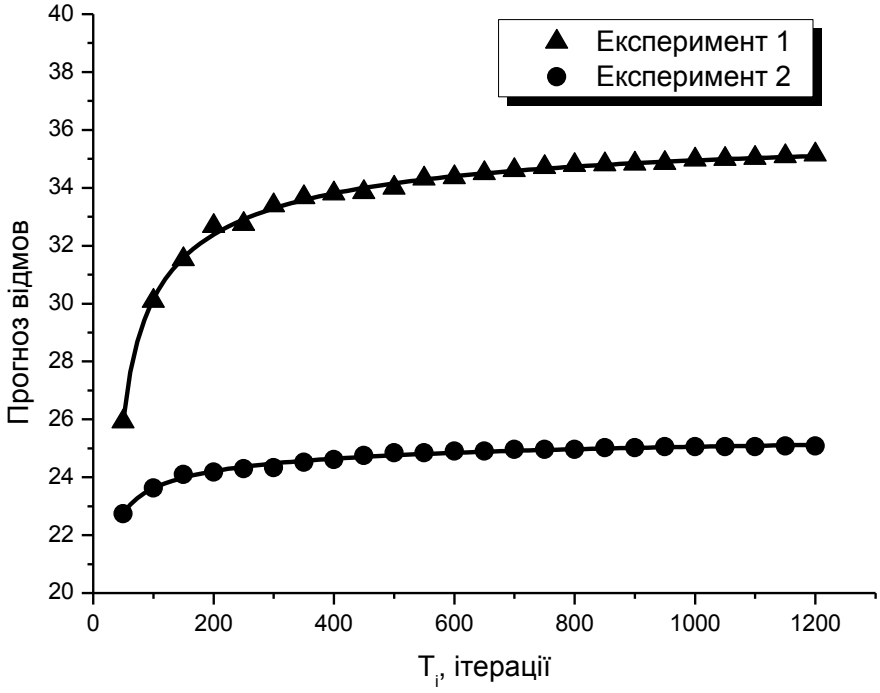


Рис. 4.8. Прогноз загальної кількості відмов ПС згідно МНПС в залежності від тривалості процесу тестування (точки) та лінія регресії за рівнянням (4.2) для першого і другого експериментів (табл. А.1 та А.2).

Після отримання чисельних значень параметрів рівняння регресії, для визначення якості опису експериментальних точок рівнянням регресії було розраховано коефіцієнт детермінації (R^2) експериментальних значень та значень, отриманих з рівняння (4.2). Значення цього коефіцієнта та прогнозовані значення граничної кількості відмов за рівняннями (2.3) та (4.2) наведені в табл. 4.1 та 4.2 для першого та другого експериментів (табл. А.1 та А.2) відповідно.

Таблиця 4.1

Результати прогнозування загальної кількості помилок для експерименту 1

Кількість ітерацій	**Прогноз за (2.3)**	**Прогноз за (4.2)**	R^2	**Різниця прогнозу, %**
1	2	3	4	5
300	33,4	34,6	0,992	3,7
400	33,8	35,0	0,996	3,5

1	2	3	4	5
500	34,0	34,6	0,997	1,7
600	34,4	35,1	0,997	2,1
700	34,6	35,4	0,997	2,4
800	34,8	35,7	0,997	2,7
900	34,8	35,7	0,997	2,6
1000	35,0	35,7	0,997	2,2
1100	35,0	35,8	0,998	2,1
1200	35,1	35,8	0,998	2,0

Як видно з табл. 4.1, використання описаної процедури та рівняння (4.2) дає змогу покращити прогноз на 2–3 %, або ж скоротити тривалість (а відповідно і витрати) тестування в даному прикладі вдвічі – з 1200 до 600 ітерацій з отриманням практично однакового прогнозу (35,1 помилки). Нагадаємо, що згідно з [128] для цього експерименту критерій достатності тестування, а отже мінімальна тривалість процесу тестування, досягався щонайменше при 650 ітераціях тестування. Однак у випадку такої мінімальної тривалості процесу тестування модель з індексом складності [66] прогнозує кількість невиявлених помилок в програмі на рівні 34–34,5, а використання запропонованого регресійного аналізу вже на цьому етапі дає прогноз на рівні 35,0–35,5 (див. табл. 4.1). З іншого боку, при проведенні повного циклу тестування, описаного в [90] використання моделі з індексом складності дає граничне значення біля 35 помилок, тоді як використання регресійного аналізу прогнозів моделі – 36 помилок, що ближче до реального значення.

Таблиця 4.2

Результати прогнозування загальної кількості помилок для експерименту 2

Кількість ітерацій	**Прогноз за (2.3)**	**Прогноз за (4.2)**	R^2	**Різниця прогнозу, %**
300	24,3	24,3	0,994	0,0
400	24,6	26,5	0,984	7,6
500	24,8	30,5	0,982	22,8
600	24,9	30,6	0,986	22,8
700	25,0	29,9	0,989	19,8
800	25,0	30,3	0,989	21,3
900	25,0	27,5	0,989	9,9
1000	25,1	26,5	0,990	5,7
1100	25,1	25,9	0,989	3,3
1200	25,1	25,7	0,990	2,4

Аналіз даних табл. 4.2 не такий однозначний, як у випадку першого експерименту (особливо у випадках, коли тестування припинялось після 500–800 ітерацій), однак також дозволяє зробити висновок про те, що використання запропонованої процедури регресійного аналізу [199, 207] з використанням виразу (4.2), покращує прогноз загальної кількості відмов ПС на 2–5 %, або ж дозволяє скоротити тривалість процесу тестування в даному випадку практично на третину – з 1200 до 900 ітерацій.

Таким чином, використання описаної процедури дає змогу покращити точність прогнозу на 3–5 %, або ж скоротити тривалість тестування в середньому на 30–40%.

4.3. Засоби автоматизованого генерування сценаріїв тестування програмних систем

4.3.1. Тестування на основі моделі функціонування ПС

Однією з найновіших технологій для автоматизованого формування сценаріїв тестування ПС є тестування на основі моделі функціонування [208–211] – тестування ПС, в якому тестові сценарії частково або цілком формуються з моделі, яка описує деякі аспекти (частіше функціональні) тестованої системи. Моделі функціонування програмної системи можуть відображати її поведінку або використовуватися для створення тестових стратегій чи середовища тестування.

У загальному модель ПС, що описує об'єкт тестування, як правило є абстрактною і описує часто лише частину його функціональності та властивостей. Тести, що генеруються з таких моделей, теж абстрактні і не можуть безпосередньо використовуватися для тестування об'єкта. На основі таких сценаріїв абстрактних тестів необхідно виконати їхню реалізацію для програмної системи, що тестується. Зараз уже деякі середовища розроблення моделі ПС (Conformiq Designer, BPM-Xchange) містять достатню кількість інформації для генерування виконуваних тестів.

Існує ряд моделей функціонування ПС, що використовуються засобами автоматизованого формування тестових сценаріїв ПС. У загальному вони поділяються на чотири великі групи, які у свою чергу під час моделювання відповідно використовують [212]: скінченні автомати, граф станів, UML діаграми та марковські ланцюги. Крім цього існують моделі, побудовані на основі дерева прийняття рішень, таблиць рішень та граматик, але вони є новими і не достатньо дослідженими [212].

Граф станів, який ще іноді називають "потік тестування", використовується для моделювання ПС [213, 214] та формування тестових сценаріїв і відображає реальне використання таких систем. Він точніший за використання скінченних автоматів. Перевагами моделей даного класу є

можливість їхнього застосування для багатопотокових програм, але все-таки такі моделі важко спроектувати та реалізувати для складних ПС.

Широкого застосування набуло тестування ПС на основі діаграм уніфікованої мови моделювання UML [215–218]. Як правило, автоматизоване генерування тестових сценаріїв відбувається на основі діаграми станів [216], прецедентів [217], послідовності [218]. Основними перевагами такого підходу є можливість його використання для складних систем та моделювання ПС з паралельним виконанням.

Моделі на основі скінченних автоматів є відомими динамічними моделями для автоматизованого формування тестів [219, 220]. Формально скінченні автомати, що відтворюють програмну систему, є сукупністю п’яти параметрів (I, S, T, F, L), де I – множина вхідних параметрів ПЗ, S – множина всіх станів системи, T – це функція, яка визначає, чи відбувається перехід системи з деякого стану, F – множина станів системи, які є кінцевими (у них система завершує свою роботу) та L – стани, у яких система починає свою роботу. Основними недоліками використання даних моделей є легкість їхнього проектування та підтримування лише для ПС із малим рівнем складності, неможливість використання даного класу моделей для паралельних обчислювальних систем та програм з нескінченною кількістю станів [221–223].

Ще однією великою групою моделей для автоматизованого формування тестових сценаріїв є моделі на основі дискретного марковського ланцюга [224–227]. За структурою марковські ланцюги подібні до скінченних автоматів, але очевидною перевагою використання такого підходу є не лише генерація тестів, але й аналіз відмов для оцінювання та прогнозування надійності ПС. Сценарії тестування отримуються через послідовні переходи між модулями ПС у відповідності до ймовірностей передавання контролю. Генерування тестових сценаріїв може бути автоматизоване за допомогою генератора випадкових чисел будь-якою мовою програмування високого рівня.

Перевагами використання даного підходу для автоматизованого формування сценаріїв тестування ПС є:

- зменшення витрат – можливо одна з найважливіших причин застосування тестування на основі моделі функціонування ПС. У працях [209, 228, 229] описано, як за допомогою такого підходу покращується якість програмного забезпечення, а також ефективність етапу тестування ПС. У роботі [228] показано, що з використанням тестування на основі наведених моделей можна виявити в 6 разів більше помилок, ніж традиційними методами тестування ПС;
- досягнення найкращого покриття коду тестами – тестування на основі моделі функціонування ПС дає змогу домогтися кращого покриття тестами [229];
- легкість документування – згідно праці [230], завдяки такому підходу легко здійснювати детальне документування, яке можна використати як навчальний посібник для нових тестерів у команді, адже вони можуть дізнатися і зрозуміти властивості та поведінку ПС;
- повторне використання засобів – ще одна перевага такого підходу, тому що тести, сформовані на основі моделі функціонування ПС можуть легко повторно використовуватись та відтворюватись на етапі регресійного тестування [209];
- краще розуміння системи – розроблення та використання моделей функціонування ПС дає змогу покращити розуміння системи в цілому. У відповідності до роботи [231] створення моделі функціонування ПС дало змогу виявити неточності в специфікаціях і в деяких випадках помилки на ранніх стадіях процесу розробки ПС;
- зменшення вартості ризику відмов при експлуатації ПС за рахунок урахування середовища функціонування ПС під час внутрішнього тестування.

Крім переваг можна виділити ряд основних обмежень та проблем, що виникають під час автоматизованого формування сценаріїв тестування (АФСТ) на основі моделі функціонування ПС:

- створення ефективної моделі функціонування ПС вимагає додаткового часу;
- якість розробленої моделі функціонування ПС залежить від глибокого розуміння ПС розробником;
- якість тестів залежить від коректності розробленої моделі функціонування ПС.

4.3.2. Модель функціонування програмної системи з урахуванням змінних коду та архітектури

Модель функціонування відображає процес виконання ПС та деякий набір її параметрів. Як правило, модель функціонування ПС представляється у вигляді зваженого графа, вершинами якого є модулі програми, а зваженими ребрами – ймовірності передавання управління між ними [232–234]. Коректна модель функціонування ПС з урахування необхідних параметрів ПС дає можливість точніше здійснювати АФСТ, а також оцінювати надійність ПС моделями, побудованими на основі архітектурного підходу.

Проте, на основі відомих моделей функціонування ПС не можна сформувати набори тестів, які забезпечують високий показник метрики покриття значень параметрів коду. Тому важливим завданням є врахування змінних та їхніх класів еквівалентності під час створення нових моделей. Крім цього, важливість врахування використання змінних ПС випливає з того, що багато моделей аналізу надійності ПС використовують її метрики, які в свою чергу беруть до уваги використання змінних ПС. Наприклад, Bieman та Kang запропонували метрики зв'язності класу, які ґрунтуються на прямих і непрямих з'єднаннях між парами методів [235], що реалізується використанням однакових змінних. Ott, Bieman, Kang та Mehra розробили модель секціонування класу [236], що ґрунтується на екземплярних змінних класу. Крім цього, метрики

складності програм поділяються на три основні групи [237]: метрики розмірності програм; метрики складності потоку керування програм; метрики складності потоку даних програм. Метрики третьої групи базуються на оцінці використання, конфігурації і розміщення даних в програмі. У першу чергу це стосується глобальних змінних. До даної групи належать метрики Chapin [238], суть яких полягає в оцінці окремо взятого програмного модуля за допомогою аналізу характеру використання змінних зі списку вводу-виводу; метрика складності McClure [239], що базується на кількості можливих шляхів проходження програми, кількості керуючих конструкцій і змінних; метрика звернення до глобальних змінних [239].

Варто зауважити, що помилки, які виникають на пізніших етапах розробки ПС: регресійного тестування, альфа- і бета-тестування, застосування, як правило, зумовлені складними сценаріями, першими кроками яких є нестандартна ініціалізація певних змінних, об’єктів або модулів, а подальшими кроками – їх використання. При цьому такі помилки проявляються тільки при використанні обмеженої підмножини значень змінних. Такі сценарії складно покрити автоматизованим чи ручним тестуванням у зв’язку з тим, що кожний етап складного сценарію може мати певну кількість степенів свободи, тобто можливих підмножин значень набору змінних, що він використовує. Відповідна кількість усіх сценаріїв буде добутком усіх степенів свободи кожного етапу, що є доволі великим числом. Саме тому окремі підмножини значень змінних не можуть бути відтестовані за допомогою стандартних тестових сценаріїв, оскільки тестування ПС часто не покриває такі випадки, бо час відведений на тестування є обмеженим. Помилки, що виникають у цьому випадку, є вагомими для корекції, оскільки часто вони відслідковуються вже на етапі регресійного тестування, впровадження або підтримки ПС. Згідно з дослідженнями Кембріджського університету, вартість помилок, що виникають на етапі експлуатації ПС, за рік становить близько 312 мільярдів доларів [240]. Також емпірично встановлено діаграму зростання вартості виправлення помилок на відповідному етапі ЖЦ ПС (рис. 4.9) [241]. Тому важливо відслідковувати такі

сценарії ще на етапі розроблення ПС, що надасть змогу зменшити економічні та трудові ресурси.

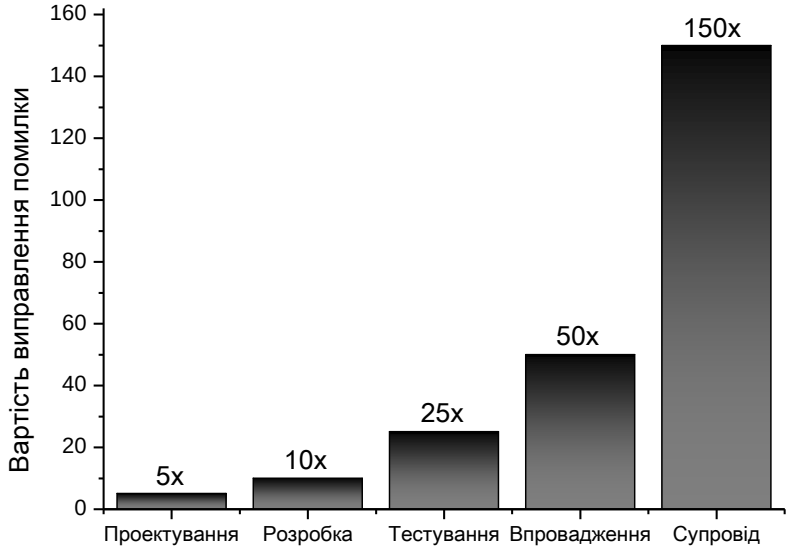


Рис. 4.9. Вартість виправлення помилок ПС на різних етапах її ЖЦ.

Опис архітектурної моделі функціонування ПС, що ґрунтується на варіативності змінних коду.

Для розв'язання описаних вище завдань у дисертаційній роботі розроблено математичну модель функціонування ПС [198] у вигляді орієнтованого графу $G = \{C, P\}$, де C – множина модулів ПЗ, P – множина переходів між відповідними модулями.

Процес виконання ПС зображено за допомогою пройдених шляхів такого графа, кожну вершину якого подано як модуль ПС, що змінює значення множини змінних, а ребра відповідають послідовностям виклику модулів.

Основні позначення, що використовуються в моделі:

- V – множина усіх змінних ПЗ;
- V_i – пара (v, EC), де v – змінна програмного продукту, а $EC = \{EC^{cor}, EC^{incor}\}$ – множина відповідних правильних EC^{cor} та неправильних EC^{incor} класів еквівалентності такої змінної у модулі C_i;

- V_i^{used} – множина змінних та відповідних класів еквівалентності, які використовуються у модулі C_i. Вважається, що змінні та їхні значення є основним джерелом помилок ПЗ;
- $V_i^{change} = V_i^{change\_user} \cup V_i^{change\_program}$ – список змінних, що змінюються у модулі C_i. Ця множина є об'єднанням множини змінних, які можуть змінюватись користувачем $V_i^{change\_user}$ (змінні, які можна протестувати за допомогою стратегії "чорної" скриньки), а також множини змінних, які можуть бути змінені лише внутрішньою логікою програми $V_i^{change\_program}$ (тестуються лише методами "білої" скринькою);
- V_i^{fail} – список змінних та відповідних неправильних класів еквівалентності, що можуть викликати помилки у модулі C_i.

Тоді кожен вузол графа C_i є набором відповідних множин $\left(V_i^{used}, V_i^{change}, V_i^{fail}\right)$, а також характеризується списком інцидентних дуг, що в свою чергу містять ймовірності передавання контролю p_{ij} до іншого вузла C_j $\left(i, j = \overline{1, N}\right)$.

Для визначення кількості необхідних сценаріїв тестування для повного покриття тестами функції/методу потрібно обчислити добуток усіх можливих змінних функції/методу на добуток кількості класів еквівалентності кожної змінної. У відповідності до цього у [170] введено вираз:

$$S_f = \prod_{i=1}^{N_f} \prod_{v \in V_i^{used} \cup V_i^{changed}} card(EC), \tag{4.3}$$

де N_f – кількість вхідних змінних методу, EC – множина класів еквівалентності відповідної змінної v.

Відповідно, кількість можливих сценаріїв тестування для модуля «клас» запропоновано визначати як суму сценаріїв тестування кожної функції/методу:

$$S_o = \sum_{f=1}^{N_M} S_f, \tag{4.4}$$

де N_M – кількість методів.

У свою чергу кількість сценаріїв тестування, необхідних для повного покриття пакетів ПС, визначається як сума потрібних сценаріїв кожного класу даного пакету за співвідношенням:

$$S_p = \sum_{o=1}^{N_o} S_o, \tag{4.5}$$

де N_o – кількість екземплярів усіх класів під час виконання програми, S_o – визначається за формулою (4.4).

Декомпозицію модулів ПС необхідно проводити, враховуючи відповідну кількість тестових сценаріїв ПС та кількість її модулів. У випадку, якщо операції забиратимуть надто велику кількість ресурсів, можна спростити цей процес шляхом аналізу лише окремих модулів. Отже, врахування не лише основних характеристик модулів, але і показника кількості необхідних тестів для повного покриття усіх модулів ПС є важливим під час декомпозиції ПС для побудови її моделі функціонування.

Практичні питання побудови моделі функціонування ПС

Аналізуючи вигляд моделі функціонування, зрозуміло, що при її створенні необхідно розв’язати три основні задачі: визначення набору модулів ПС; обчислення ймовірностей передавання управління між модулями; пошук інформації про можливі відмови. На даний час відомими засобами, які можуть вирішити ці питання, є:

- Реєстрація – система моніторингу повідомлень та подій, що виникли у ПС.
- UML діаграми, які поділяються на структурні та поведінкові, що в свою чергу дають змогу визначити сукупність модулів та взаємозв’язків між ними ще на ранніх етапах ЖЦ ПС.

- Документація – сукупність документів, у яких містяться вимоги до структури ПС або його функціональні можливості (зокрема специфікація вимог).
- Аналіз коду – як статистичний так і динамічний, що дає змогу визначати повніше усі характеристики для формування множини модулів, які повинні бути відображені в моделі функціонування ПС.

Реєстрація. Результатом реєстрації ПС є файл з даними про те, які модулі ПС виконувались і в якому порядку, а також які під час цього виникали відмови.

Визначення сукупності модулів. Із файлу протоколу, що утворюється під час користування ПС, можна легко побудувати множину модулів, які під час цього виконувались. Але варто зауважити, що передумовою використання даного підходу для формування множини модулів є виконання всіх модулів ПС.

Обчислення ймовірностей передавання контролю між модулями ПС. У результаті виконання ПС управління передається між модулями і, маючи результати передавання управління, можна визначити показники переходів між модулями. Для визначення ймовірностей переходів в модуль C_j з C_i необхідно визначити відношення кількості переходів з модуля C_i в модуль C_j до загальної кількості переходів з модуля C_i.

Інформація про помилки. Файл протоколу, як правило, містить детальну інформацію про рівень складності, час, причину та місце виникнення відмов.

Для перевірки запропонованого підходу у [156] розроблений реєстратор на основі мови програмування Java. Реалізований засіб дозволяє досліджувати ПС без внесення змін в програмний код. Такий підхід дає можливість визначати та уточнювати матрицю ймовірностей переходів між модулями ПС навіть на етапі її впровадження та експлуатації з урахуванням реальних сценаріїв використання ПС різними класами користувачів.

UML діаграми. Найчастіше такий підхід використовується на ранніх етапах життєвого циклу ПС, але необхідно пам’ятати, що на цьому етапі можна

тільки наближено говорити про модулі ПС, а згадані ймовірності отримуються апріорно на основі експертних оцінок.

Визначення сукупності модулів. Під час розроблення архітектури об'єктно-орієнтованих систем використовується UML діаграма класів, за допомогою якої здійснюється опис модулів ПС, а також зв'язків між ними. З діаграми класів можна сформувати лише структуру матриці ймовірностей передачі управління між модулями, проте необхідно зауважити, що неможливо отримати оцінки ймовірності передавання управління між модулями.

Обчислення ймовірностей передавання контролю між модулями ПЗ. Із використанням діаграми прецедентів на основі ймовірностей (або відносних частот) виконання усіх випадків використання можна оцінити ймовірності передавання управління між модулями ПС. Крім цього, використовуючи UML діаграму послідовностей, можна отримати кількість переходів між модулями, з яких відповідно легко визначається матриця ймовірностей передавання управління між ними. Здійснивши аналіз розроблених UML діаграм послідовностей під час проектування ПС, необхідно обчислити сумарну кількість переходів з модуля в модуль та сформувати матрицю передавання контролю між ними.

Інформація про помилки. Жодна з UML діаграм не містить інформації про відмови, які можуть виникнути у ПС.

Документація. Як правило, на ранніх етапах ЖЦ ПС розробляється специфікація вимог, яка містить усі функціональні завдання майбутньої ПС і основні модулі ПС. На етапі проектування ПС розробниками паралельно створюється детальна документація про модулі проекту, його змінні, результати роботи для подальшої підтримки та супроводу.

Визначення сукупності модулів. Із документації легко отримати перелік усіх модулів, що існують в ПС з відповідними змінними, але за умови, що документація складена правильно з дотриманням усіх вимог. Щодо специфікації вимог, то вона надає змогу визначити лише наближений набір основних модулів, який буде неповним.

Обчислення ймовірностей передавання контролю між модулями ПС. Визначення ймовірності передавання контролю між модулями ПС з використанням документації можна здійснити за допомогою описаних основних сценаріїв користувача. Але, як правило, у документації не описані усі сценарії використання ПС, тому матриця ймовірностей переходів буде неточною.

Інформація про помилки. В основному у документації описані відмови, що можуть виникнути унаслідок виконання ПС, але розробники намагаються їх передбачити, тому з використанням даного підходу інформацію про помилки неможливо проаналізувати.

Аналіз коду. Такий підхід забезпечує отримання повнішої інформації про набір модулів моделі функціонування ПС та їхні параметри.

Визначення сукупності модулів. Шляхом аналізу коду ПС є можливість визначити весь набір існуючих модулів в проекті.

Обчислення ймовірностей передавання контролю між модулями ПС. З аналізу коду неможливо визначити ймовірності переходів між модулями, але можна визначити наявність ребра між двома вузлами графу моделі функціонування ПС.

Інформація про помилки. За допомогою даного підходу можна проаналізувати змінні програми та ті їхні значення, що можуть бути джерелом відмови, тому, відповідно, є змога отримати інформацію про помилки, але варто зауважити, що це надзвичайно складний та тривалий процес.

Таким чином, підсумовуючи можна зазначити, що окремо жоден із існуючих підходів не забезпечує повного визначення наборів даних для створення моделі функціонування ПС, тому варто поєднувати дані варіанти для отримання найбільш повної вхідної інформації. Для формування більш адекватної моделі функціонування ПС, у якій на відміну від відомих, враховується множина змінних та відповідних значень коду, у роботі [198] поєднується одночасно реєстрація та аналіз коду, адже з файлу протоколу можна точніше обчислити ймовірності передавання контролю, а за допомогою аналізу коду визначити сукупність усіх модулів та змінних ПС.

У [170] процес розроблення моделі функціонування ПС в загальному складається із двох основних частин: додавання реєстрації та відповідних викликів модулів (preprocessor), парсинг лог-файлу, створеного після користування перекомпільованою ПС (postprocessor).

Препроцесор. Зчитується вхідний код програми. Після цього додаються виклики лог-повідомлень у програмний код на початку та вкінці відповідного модуля (функції, класу, пакету). Унаслідок цього програма буде змінена таким чином, що у відповідний лог-файл записуватимуться повідомлення про виклик відповідного модуля під час виконання ПС.

Компіляція та виконання програми. Програма запускається та виконується багато раз тестерами та кінцевими користувачами для отримання повнішої інформації для визначення ймовірностей переходів між модулями ПС.

Постпроцесор. Зчитується вхідний код ПС для аналізу модулів та їхніх змінних, а також лог-файл для обчислення ймовірностей переходів між модулями ПС.

Загальна схема такого підходу зображена на рис. 4.10 [170].

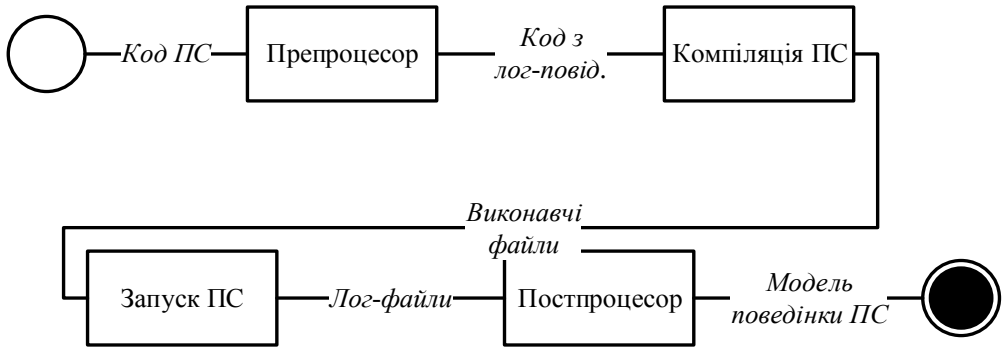


Рис. 4.10. Схема створення моделі функціонування ПС.

4.3.3. Автоматизоване формування сценаріїв тестування програмної системи на основі моделі її функціонування

Сценарієм тестування або тестовим випадком (тестом) будемо називати послідовність виконання модулів ПС з певними значеннями змінних. Тест може

виконатись успішно або завершитись через виникнення помилки у деякому модулі.

Існує два найважливіші методи тестування програмного продукту [249]:

- методи "чорної скриньки", які розглядають системні характеристики програм, ігноруючи їхню внутрішню логічну структуру;
- методи "білої скриньки" – це детальне дослідження внутрішньої логіки і структури коду, під час якого відома повна інформація про вихідний код ПС.

З урахуванням моделі функціонування ПС для формування сценаріїв тестування взаємозв’язки та переходи між станами подаються у вигляді графа, як було зазначено у попередньому розділі. Цей граф можна побудувати методом білої скриньки шляхом аналізу коду, що найбільше відповідатиме реальному характеру ПЗ, або створити граф методом чорної скриньки за допомогою реєстрації виклику методів, проте у ньому можливий варіант, що деякі з переходів не будуть враховані (рис. 4.11).

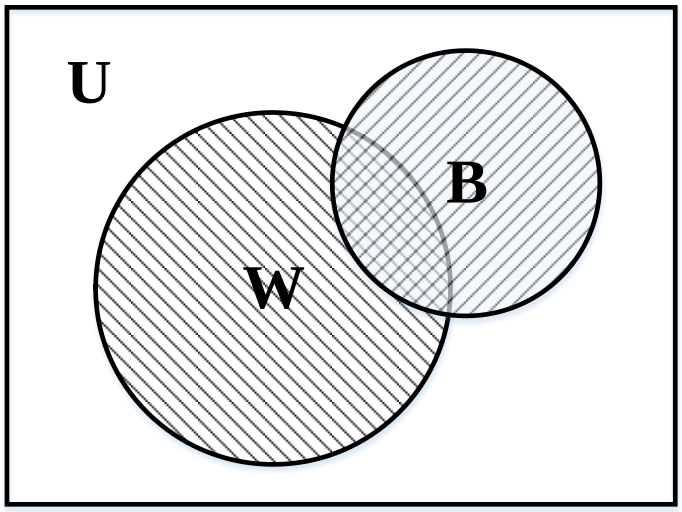


Рис. 4.11. Множини переходів у графі під час його створення методом $\boldsymbol{B}$ – чорної скриньки, $\boldsymbol{W}$ – білої, $\boldsymbol{U}$ – множина всіх переходів у графі.

Метод АФСТ за стратегією "чорної скриньки"

У відповідності до описаної в [198] моделі функціонування ПС вважається, що кожен сценарій тестування ПС складається з декількох кроків $T^k =$

$\{T_0^k, T_1^k, \dots, T_n^k\}$, де кожен крок сценарію $T_j^k = (C_j^k, V_j^k)$ визначається парою: модулем C_j^k та множиною змінних V_j^k з відповідними класами еквівалентності, що фактично змінюються в цьому модулі. Змінні набувають своїх значень на кожному кроці тесту внаслідок введення даних користувачем, зчитуванням з бази даних тощо, через що можуть виникнути помилки під час виконання даного сценарію тестування.

У відповідності до реального процесу тестування, набори тестів будуються ітераційно, крок за кроком виконуються в ПС з урахуванням виявлених відмов під час формування наступного набору тестів.

У [173] розроблено метод АФСТ на основі описаної в [198] моделі функціонування ПС за стратегією "чорної скриньки", який складається із двох основних частин: початкова генерація тестів без інформації про відмови та решти K наборів тестів, що її враховують.

При описі методу формування множини тестів використано такі позначення: $F_j(C_i)$ – кількість тестів, під час яких відбулась відмова, і які включають модуль C_i на j-тій ітерації. $Num(C_i)$ – множина номерів усіх модулів, у які можна передати контроль із модуля C_i; $Com(C_i)$ – множина всіх модулів, у які можна передати контроль із модуля C_i; N – кількість всіх вершин графа функціонування ПС; TS – набір усіх сценаріїв тестування; $Cov(C_i)$ – міра покриття тестами модуля C_i, v_i – змінна модуля C_i.

Основними кроками методу є [173]:

Крок 1. *Ініціалізація.* Вибір мінімальної кількості α покриття всіх модулів ПЗ та середньої довжини β тестового сценарію з досвіду тестувальника. Рекомендованим діапазоном початкових значень цих параметрів може бути $\alpha \in \overline{5,10}$, $\beta \in \overline{5,7}$. Установлення $TS = \{\}$, 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 вибирається початковий крок тесту з ймовірністю $p_j^k = 1/N$;
3. Із ймовірністю $p_{j+1}^k = 1/card\left(Com(C_j)\right)$ вибирається наступний крок T_{j+1}^k у k-ому сценарії; $j++$;
4. Якщо $j \le \beta$, тоді здійснюється пункт 3;
5. Приєднання сформованого тесту до набору: $TS = TS \cup \{T^k\}$;

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

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

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 вибирається номер T_j^k першого кроку тесту, з ймовірністю $p_j^k = \max\limits_{i=1\ldots N} {F_{m-1}(C_i)}\Big/{\sum_{l=1}^{N} F_{m-1}(C_l)}$;
3. Із ймовірністю $p_{j+1}^k = \max\limits_{i \in Num(C_j)} {F_{m-1}(C_i)}\Big/{\sum_{l \in Com(C_j)} F_{m-1}(C_l)}$ обирається наступний крок у T_{j+1}^k сценарії; $j++$;
4. Якщо $j \le \beta$, тоді здійснюється перехід на пункт 3;
5. Приєднання побудованого тесту до набору: $TS = TS \cup \{T^k\}$.

Крок 7. Перевірка покриття усіх модулів. Якщо $Cov(C_i) < \alpha, i = \overline{1,N}$, тоді $k++$, тобто здійснюється перехід на Крок6.

Крок 8. Перехід на крок 4.

Крок 9. Кінець.

Схема описаного методу АФСТ наведена на рис. 4.12.

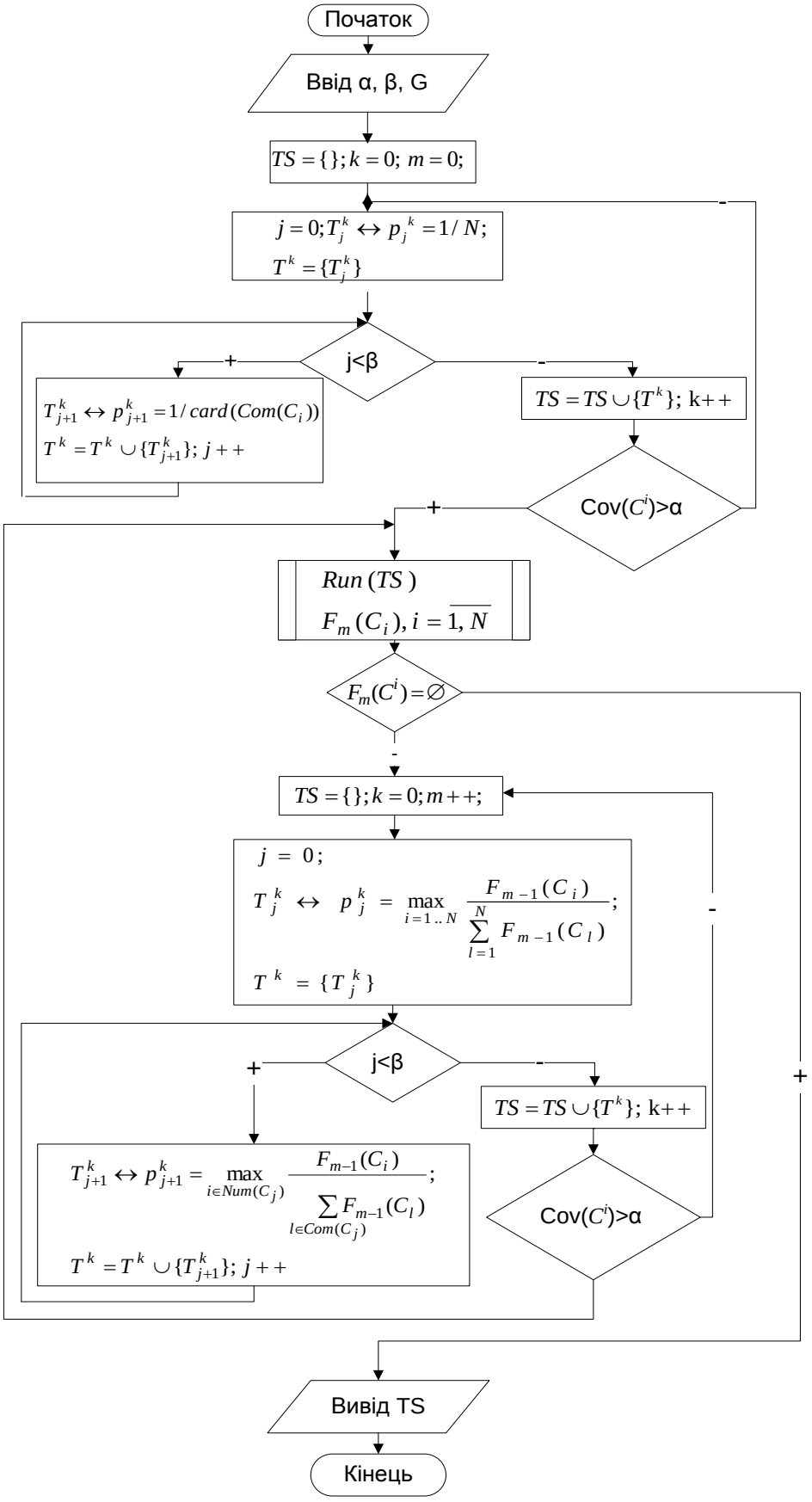


Рис. 4.12. Схема методу автоматизованого формування сценаріїв тестування ПС за стратегією "чорної" скриньки.

Побудовані тести виконуються для різних вхідних даних таким чином, щоб кожний вхідний параметр набував принаймні одного значення з кожного класу еквівалентності. Класи еквівалентності змінних формуються не з аналізу коду, а відповідно до досвіду тестувальника. При цьому вхідні дані можуть і не набувати усіх можливих значень і деякі відмови будуть пропущені, що є недоліком такої стратегії тестування.

Для того, щоб цілком покрити код ПС, необхідно для кожного модуля C_i здійснити $\|v_1\| \cdot \|v_2\| \cdot \ldots \cdot \|v_n\|$ перевірок, де $\|v_i\|$ – кількість класів еквівалентності змінної $v_i \in C_i, i = \overline{1, N}$. Але через обмеження часових та людських ресурсів може бути неможливо провести таку велику кількість перевірок. Крім цього не всі перевірки відповідають реальній логіці виконання ПС (певних значень змінних не можливо досягнути при використанні ПС), тому існує задача покриття коду тестами так, щоб повністю перебрати усі можливі набори змінних, допустимих внутрішньою логікою, або, якщо таких випадків дуже багато, то перебрати усі кортежі класів еквівалентності змінних довжиною до деякого ступеня $Step$, який визначається співвідношенням вимог до якості ПС та затратами на процес тестування. Стандартно ($Step = 1$) обмежуються перебором усіх класів еквівалентності для кожної змінної у модулі $C_i - \|v_1\| + \|v_2\| + \ldots + \|v_n\|$. Для $Step = 2$ необхідно перевірити всі пари наборів класів еквівалентності у модулі $C_i - \sum_{k,l=1}^{N} \|v_k\| \cdot \|v_l\|$ і т.д. Тобто з використанням методу "чорної" скриньки неможна повністю перебрати усі можливі випадки значень змінних, що є суттєвим недоліком даного підходу.

Таким чином, множини тестів, що отримуються на кожній ітерації описаним методом "чорної скриньки", можуть бути використані під час тестування, що у свою чергу пришвидшує даний процес та зменшує фінансові та людські витрати. Крім цього, така генерація сценаріїв забезпечує рівномірне покриття коду, що покращує ефективність тестування, та відповідно підвищує якість ПС. Також завдяки пропорційному покриттю коду та використанню класів

еквівалентності, зростає ймовірність виявлення відмови під час тестування програми, що в результаті збільшує надійність ПС на етапі експлуатації.

Метод АФСТ ПС за стратегією "білої скриньки"

Першою перевагою стратегії "білої скриньки" є аналіз змінних, які використовуються та змінюються у модулях. Відкинувши змінні, що не використовуються, можна зменшити перебір можливих V_i^{used} – підмножина змінних, яку використовує модуль C_i. Також, проаналізувавши код, можна чітко визначити V_i^{change} – множину, яку може змінювати відповідний модуль C_i. Очевидно, що множина змінних модуля C_i дорівнює $V_i^{used} \bigcup V_i^{change}$. Якщо розглянути тестовий сценарій, який складається лише з одного кроку $T = (C, V)$, то множина змінних $\alpha(T)$, яка покривається даним сценарієм дорівнює $V^{used} \bigcap V$. Значення інших змінних можна покрити тільки тестовими сценаріями, які матимуть два або більше кроків. Множина змінних модуля C_1, яка покривається сценарієм тестування, що містить два кроки $T = \{T_0, T_1\} = \{(C_0, V_0), (C_1, V_1)\}$ та враховує два послідовних модуля C_0 та C_1 має вигляд: $\alpha(T) = V_1^{used} \bigcap (V_0 \bigcup V_1)$. У загальному випадку для тестового сценарію $T = \{T_0, T_1, \dots, T_N\} = \{(C_0, V_0), (C_1, V_1), \dots, (C_N, V_N)\}$ множина змінних, яку він покриває та впливає на процеси останнього модуля C_N дорівнює: $\alpha(T) = V_N^{used} \bigcap (V_N \bigcup V_{N-1} \bigcup \dots \bigcup V_1 \bigcup V_0)$. Для того, щоб цілком покрити модуль C_N різними значеннями змінних, необхідно, щоб $\alpha(T) = V_N^{used}$.

Зауважимо також, що сценарій тестування покриває множини змінних також і попередніх модулів. Наприклад, для проміжного модуля C_i $(i < N)$ цього сценарію T множина покриття визначається так [172, 173, 198]:

$$\alpha(T, C_i) = V_i^{used} \bigcap \left(V_0 \bigcup V_1 \bigcup \dots \bigcup V_{i-1}\right). \quad (4.6)$$

Для останнього модуля: $\alpha(T, C_N) = \alpha(T)$.

Очевидно, що чим більше кроків у сценарії, тим більше можливостей для задання різних значень змінних. З іншого боку, зі збільшенням кількості кроків у сценаріях тестування зменшується ймовірність того, що тестовий сценарії буде

пройдений до кінця, оскільки він може бути зупинений через виникнення відмови. Для зменшення трудоємності процесу тестування та підвищення його ефективності необхідно вибрати такий набір сценаріїв та кроків у ньому, які би цілком покривали значення усіх змінних модулів та мали найменшу кількість кроків. Варто також зауважити, що у ПС буде багато змінних, які набувають своїх значень лише в одному модулі (на формі, у конфігурації, і т.д.), тому під час формування тестових сценаріїв необхідно враховувати цей факт. Позначимо максимальну довжину тестового сценарію для даного ПС через β_{max}. Тоді міра покриття PVC (Parameter Value Coverage – значення параметра покриття змінних коду, яке забезпечує, щоб змінні ПС набували усіх своїх значень) визначається співвідношенням [198]:

$$PVC(TS) = \sum_{i=1}^{N} \sum_{v_i \in V_i^{used}} Ind_{v_i}\left(\bigcup_{T \in TS} \alpha(T, C_i)\right), \tag{4.7}$$

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

Також у [172, 173] вводиться поняття складності виконання тестів. Оскільки з точки зору тестувальника складність виконання переходу до наступного кроку є значно меншою у більшості випадків, ніж задання даних, то складність визначається як кількість даних, що вносить користувач під час формування тесту:

$$CM(TS) = \sum_{i=1}^{card(TS)} \sum_{j=1}^{card(T^i)} V_j^i. \tag{4.8}$$

Таким чином умови формування набору сценаріїв тестування ПЗ записуються у вигляді:

$$\begin{cases} CM(TS) \to min; \\ \text{з обмеженнями } PVC(TS) = PVC_{max}(TS) = \sum_{i=1}^{N} card\left(V_i^{used}\right). \end{cases} \tag{4.9}$$

Алгоритм знаходження оптимальних покриттів є алгоритмом перебору з відсіканням гілок, який є надлишковим згідно міри покриття $PVC(TS)$ та

складності виконання $CM(TS)$. Отже, розроблений метод формування сценаріїв тестування за стратегією "білої скриньки" [172] на основі описаної в [198] моделі функціонування ПС, передбачає такі кроки:

Крок 1. *Ініціалізація*. Формується початкова множина тестових сценаріїв, що складаються лише з одного кроку $TS = \{(C_0, V_0^{change}), (C_1, V_1^{change}), \dots, (C_N, V_N^{change})\}$. На цьому кроці тестового сценарію здійснюється перебір усіх можливих параметрів, оскільки вони впливають на всі подальші кроки.

Крок 2. *Знаходження повного покриття параметрів змінних коду.* Множина тестових сценаріїв, сформована на попередньому кроці, розширюється таким чином: для сценарію $T = \{(C_0, V_0), (C_1, V_1), \dots, (C_K, V_K)\}$ з множини TS знаходяться усі модулі C_{K+1}, які суміжні компоненту C_K і приєднується новий крок (C_{K+1}, V_{K+1}) до сценарію T, де $V_{K+1} = V_{K+1}^{change} \setminus \cup_{i=1}^{K} V_i$. Додається новий сценарій до множини TS, якщо при цьому зростає $PVC(TS)$. Якщо $PVC(TS) = PVC_{max}(TS)$, то здійснюється перехід на крок 3, інакше повторюється крок 2.

Крок 3. *Мінімізація складності виконання сценарію тестування.* Для кожного сценарію з множини TS здійснюється перевірка: якщо під час вилучення його з множини TS величина $PVC(TS)$ не змінюється, то видаляємо його, інакше перевіряємо кожен крок (C_K, V_K), видаляючи з множини V_K ті змінні, які не впливають на $PVC(TS)$. Це проводиться для мінімізації $CM(TS)$.

Крок 4. *Перебір класів еквівалентності змінних.* Сформовані тестові сценарії з множини TS виконуються для різних вхідних даних таким чином, щоб кожний вхідний параметр набував принаймні одного значення з кожного класу еквівалентності. Для підвищення ефективності тестування необхідно проаналізувати взаємозв’язані змінні та перебирати усі кортежі відповідних значень класів еквівалентності, що є значно складнішою задачею. Іншим підходом є більша дискретизація вузлів графу функціонування ПС до рівня програмних виразів та операторів управління, що дає змогу глибоко проаналізувати покриття та взаємний вплив змінних.

Крок 5. *Ітераційне виконання тестових сценаріїв.* Після формування та виконання тестових сценаріїв знаходиться множина модулів $F(C_i)$, у яких виникли помилки. Якщо $F(C_i) = \emptyset$, то алгоритм закінчується, інакше, починаючи з кроку 1, будується нова множина тестових сценаріїв TS з новим покриттям:

$$PVC(TS) = \sum_{C_i \in F(C_j)} \sum_{v_i \in V_i^{used} \cap V_i^{fail}} Ind_{v_i} \left(\bigcup_{T \in TS} \alpha(T, C_i) \right). \quad (4.10)$$

Описаний алгоритм дає змогу формувати сценарії тестування ПС з максимальним покриттям коду при мінімальній складності їхнього виконання, а це підвищує ефективність процесу тестування та усунення помилок ПС, що в свою чергу, покращує її показники надійності.

4.3.4. Залежність показників надійності ПС від параметрів моделі функціонування ПС з урахуванням змінних коду та архітектури

Для підтвердження адекватності розроблених методів та отримання залежностей кількості виявлених відмов від характеристик ПС, що надасть можливість сформувати рекомендації щодо вибору стратегії тестування у [172, 198] проведено дослідження впливу характеристик ПС (параметрів моделі функціонування) на її надійність.

Спочатку було сформовано множини тестів методом АФСТ для ПС після їхнього тестування відомими методами "чорної", а потім "білої" скриньки. Порівняння кількості виявлених помилок ПС відомими методами та методом на основі моделі функціонування [172] наведено на рис. 4.13. Застосування розробленого методу формування тестів дає можливість підвищити ефективність виявлення відмов на 7–10% (рис. 4.13).

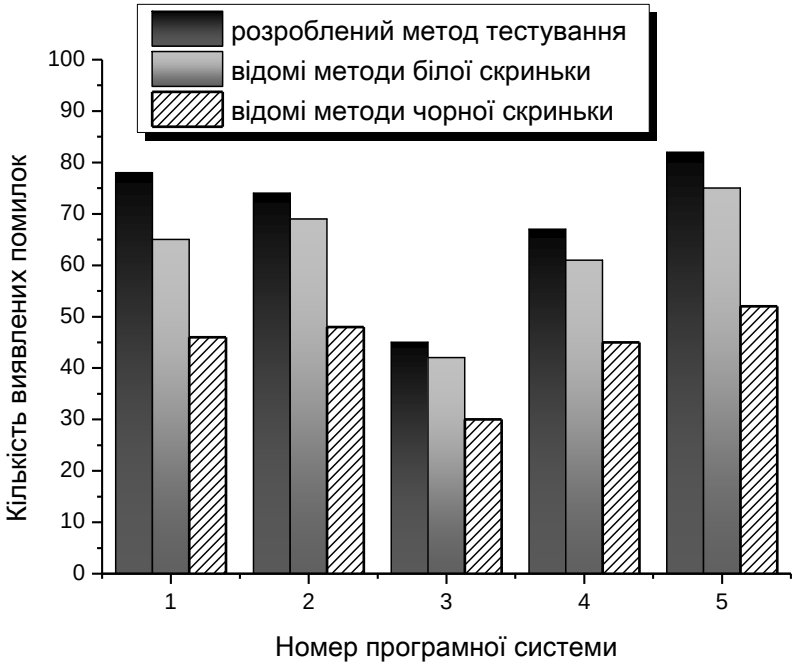


Рис. 4.13. Порівняння кількості виявлених помилок за допомогою відомих методів тестування та методу тестування на основі аналізу змінних коду [172].

1. Залежність кількості відмов від типу стратегії тестування.

На основі розробленого методу формування сценаріїв тестування реалізовано програмний засіб для аналізу надійності ПС "СОН ПЗ" [200], вхідними даними для якого є параметри моделі функціонування ПС. Проведено числові експерименти, в яких задавались різні параметри моделі функціонування ПС (кількість модулів; кількість змінних з відповідними класами еквівалентності, що використовуються та змінюються методами; ймовірності переходів між методами та інформація про помилки) та досліджувався їх вплив на показники надійності ПС [172, 198]. З використанням методу АФСТ [172] будувались набори тестових сценаріїв для двох стратегій тестування. Кількість відмов, виявлених на етапі тестування, в залежності від кількості тестових сценаріїв з урахуванням двох стратегій наведені на рис. 4.14.

Як видно з рис. 4.14, під час тестування за стратегією "білої скриньки" було виявлено в ПС більше відмов, ніж під час тестування за стратегією "чорної скриньки", адже тестування за стратегією "білої скриньки" враховує особливості

архітектури ПС та особливості її відмов. Цей факт, а також якісний вигляд залежності відповідає відомим даним тестування реальних ПС [250, 251].

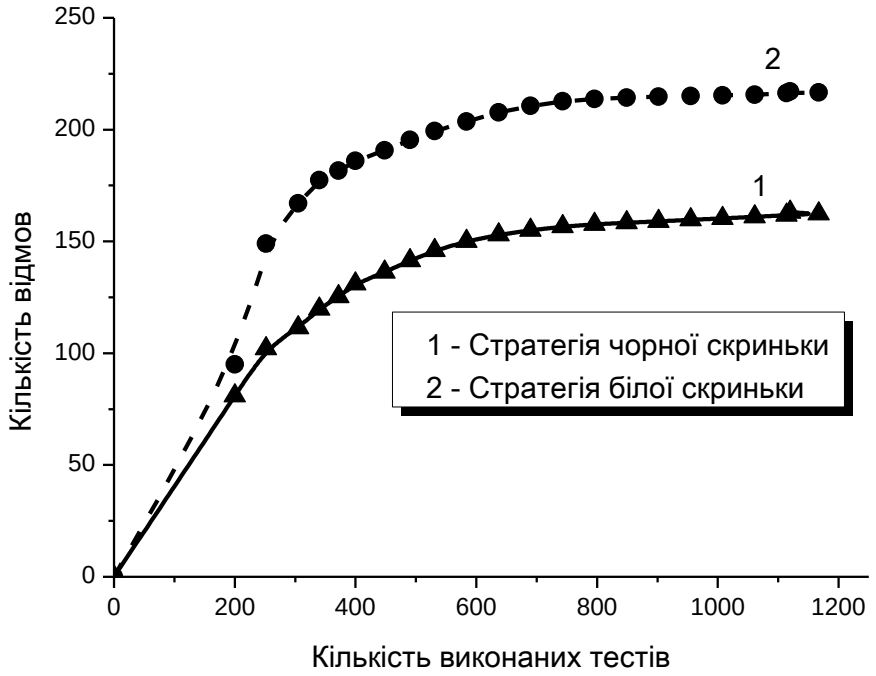


Рис. 4.14. Графік залежності кумулятивної кількості виявлених на етапі тестування відмов ПС від кількості тестів для двох стратегій тестування.

2. Залежність кількості відмов від кількості модулів ПС.

Графік залежності кількості відмов, що були зафіксовані під час процесу тестування, від кількості модулів ПС (рис. 4.15) ілюструє лінійний зв’язок між ними для обох стратегій тестування. Такий самий вигляд залежності кількості відмов від кількості модулів ПС був показаний в [251], що підтверджує правильність запропонованого у [172] методу.

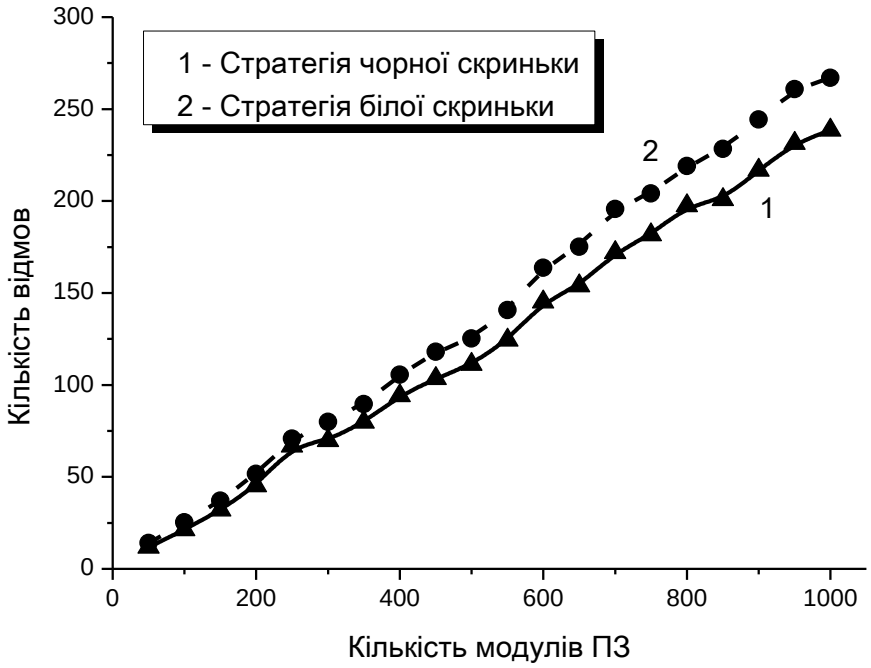


Рис. 4.15. Графік залежності кількості виявлених під час тестування відмов від кількості модулів ПС, для двох стратегій тестування.

3. Залежність кількості відмов від середньої кількості змінних, що використовуються у модулі ПС.

Вплив середньої кількості змінних, що використовуються у модулях ПС на кількість виявлених на етапі тестування відмов наведено на рис. 4.16. Кількість змінних, що використовуються модулем ПС, вимірюється відсотком від загальної кількості змінних. Як видно з графіка рис. 4.16, зі збільшенням відсотка кількості таких змінних кількість відмов під час тестування за стратегією "білої скриньки" дещо зростає, що обумовлено можливістю аналізувати усі змінні та їхні значення під час формування тестів і в результаті виявляти більше відмов. Під час тестування за стратегією "чорної скриньки" зі зростанням кількості змінних, що використовуються модулем ПС, кількість відмов зменшується. Це пояснюється тим, що стратегія "чорної скриньки" не дає змоги проаналізувати внутрішні змінні ПС, які впливають на її виконання, а лише ті, які може змінювати користувач (кількість таких змінних залишається незмінною), тому

ефективність цієї стратегії погіршується з ростом середньої кількості змінних, що використовуються модулем ПС.

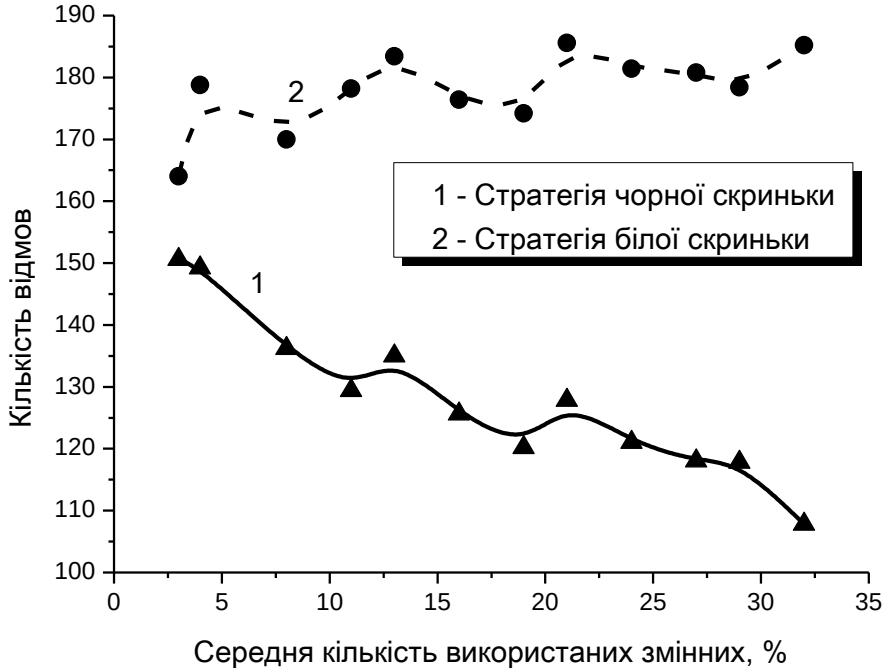


Рис. 4.16. Графік залежності кількості виявлених відмов від середнього значення кількості використаних змінних, для двох стратегій тестування.

4. Залежність кількості відмов від складності тестування ПС.

Досліджено залежності складності виявлення відмов та їх кількості під час процесу тестування за обома стратегіями. У загальному помилки, що виникають у результаті виконання ПС, можуть бути двох видів:

- прості, які виникають внаслідок належності однієї змінної до якогось неправильного класу еквівалентності (наприклад, $v_1 \in EC_1^{incor}$). Просту помилку можна знайти, виконавши один сценарій, оскільки вона завжди знаходиться в одному модулі.
- складні, які виникають у результаті належності двох і більше змінних до неправильних класів еквівалентності (наприклад, $v_1 \in EC_1^{incor}, v^2 \in v_1 \in EC_2^{incor}$). Даний тип помилок знайти дуже

складно, адже вони можуть міститись в різних модулях і тому потрібно реалізувати декілька сценаріїв для знаходження помилки.

За даними емпіричного досвіду розробників промислових ПС, частка складних помилок експоненційно зменшується зі зменшенням кількості задіяних змінних коду, а розподіл таких помилок у ПС представлений на рис. 4.17.

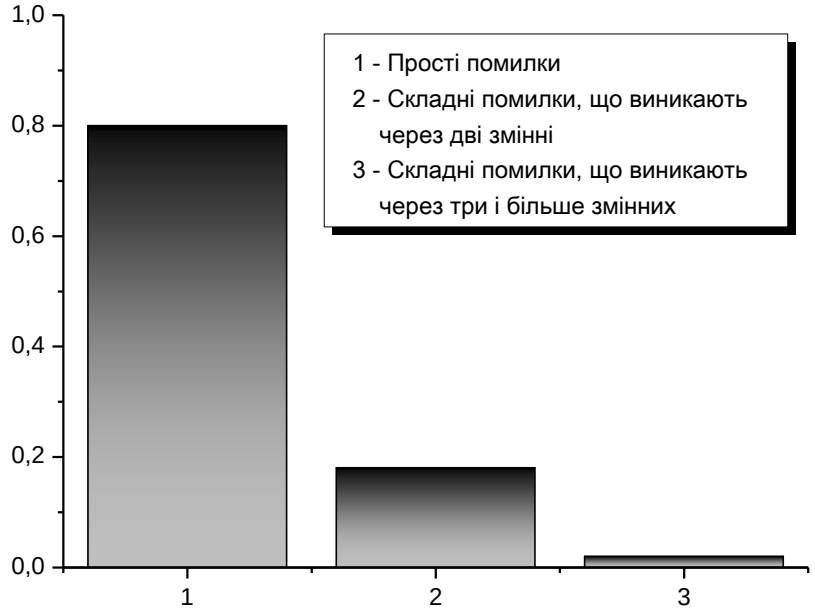


Рис. 4.17. Частка помилок різного типу у програмних системах.

Як показано на рис. 4.18, зі зростанням складності тестування зменшується кількість відмов під час тестуваннями за обома стратегіями тестування, що відповідає реальному характеру функціонування ПС.

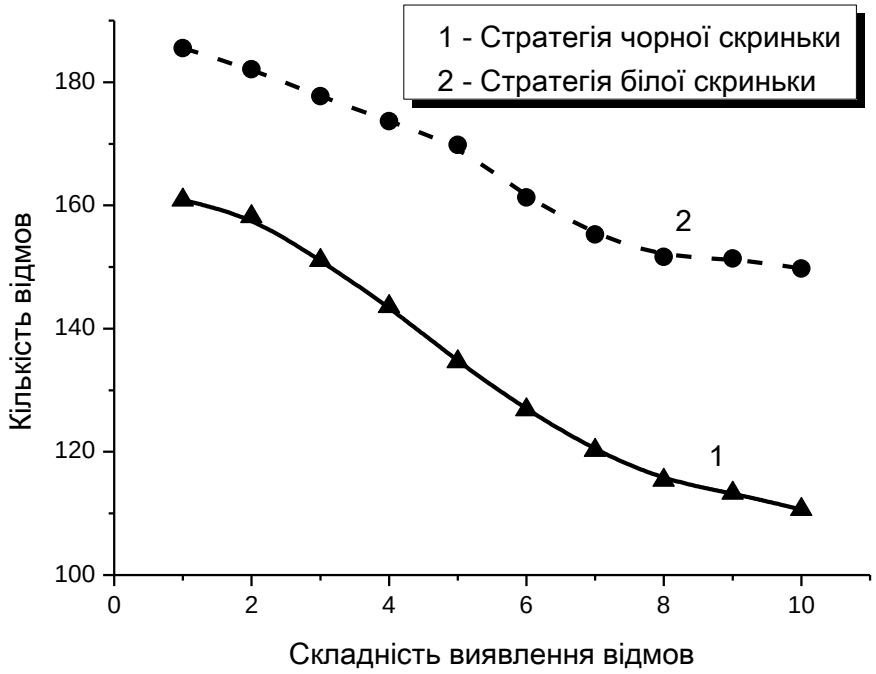


Рис. 4.18. Графік залежності кількості виявлених під час тестування відмов від складності виконання тестів ($CM(TS)$) для двох стратегій тестування.

5. Залежність кількості відмов від складності помилок.

Графік на рис. 4.19 показує вплив складності помилок на кількість виявлених відмов, як за стратегією "чорної", так і за стратегією "білої скриньки". Під складністю помилки розуміється кількість невзаємопов’язаних місць у програмному коді, де потрібно виправляти помилку. Виправивши її тільки в одному місці, можна все одно отримати відмову за іншим сценарієм. Якщо помилка виправляється у тому виразі (англ. *statement*) де виникає, то повторюваність або складність такої відмови дорівнює одиниці, інакше вона є більшою за одиницю.

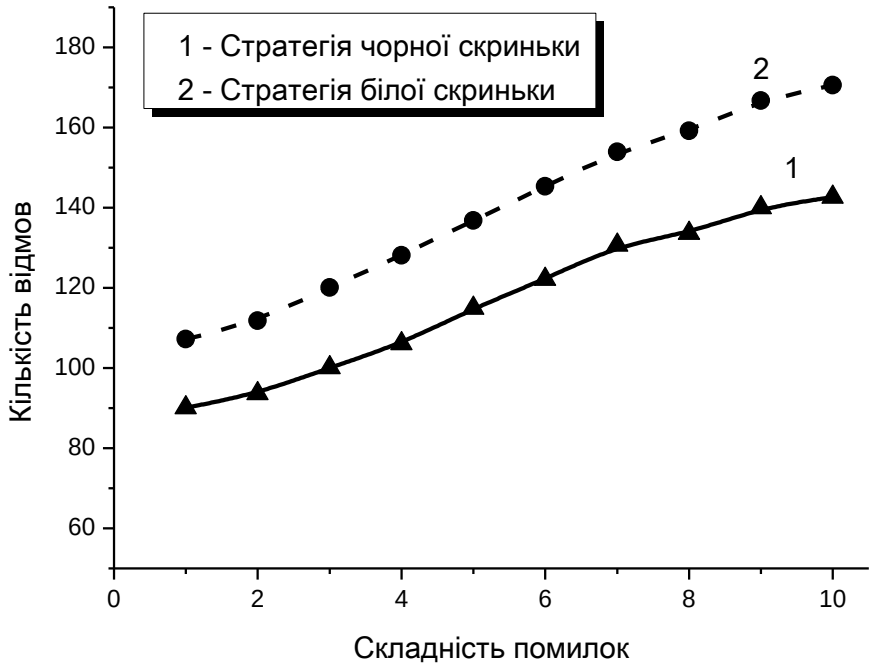


Рис. 4.19. Графік залежності кількості виявлених під час тестування відмов від складності помилок для двох стратегій тестування.

6. Залежність кількості відмов від середньої зв'язності модулів.

Вплив середньої зв'язності модулів ПС на кількість виявлених на етапі тестування відмов зображено на рис. 4.20. Зв'язність модулів ПС вимірюється середньою кількістю взаємозв'язків цього модуля з іншими. Як видно з рис. 4.20, зі збільшенням зв'язності модулів кількість відмов під час тестування обома стратегіями зростає, що обумовлено більшою кількістю можливих шляхів виконання ПС, а, отже, більшою кількістю розроблених тестів.

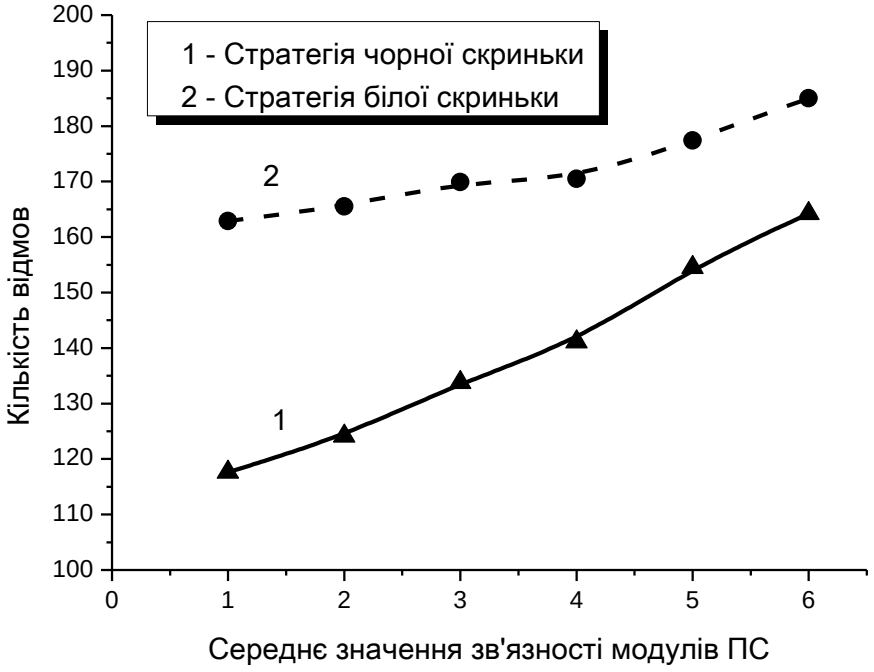


Рис. 4.20. Графік залежності кількості виявлених під час тестування відмов від зв’язності модулів ПС для двох стратегій тестування.

4.3.5. Рекомендації вибору типу стратегії тестування на основі моделі функціонування ПС

На основі створеної моделі функціонування ПС [198] та отриманих результатів тестування з використанням запропонованих методів АФСТ [172] розроблені такі рекомендації стосовно вибору стратегії тестування під час виконання проектів з розробки ПС:

1. *Вибір типу модуля під час декомпозиції* ПС *для побудови моделі її функціонування.*

Для попереднього аналізу ПС не рекомендується вибирати такий тип модуля (пакет, клас, метод), кількість яких перевищує 1000. Оскільки тривалість аналізу може перевищувати допустимі норми попереднього аналізу ПС. Для визначення кількості модулів також можна використовувати формули (4.3–4.5).

Із використанням комп’ютера з такими характеристиками: процесор Intel Atom N2800 (1.86 GHz), оперативна пам’ять 2 Гб, тип системи – 32-х розрядна,

у [170] проведено дослідження залежності часу аналізу моделі функціонування ПС від кількості її модулів (рис. 4.21).

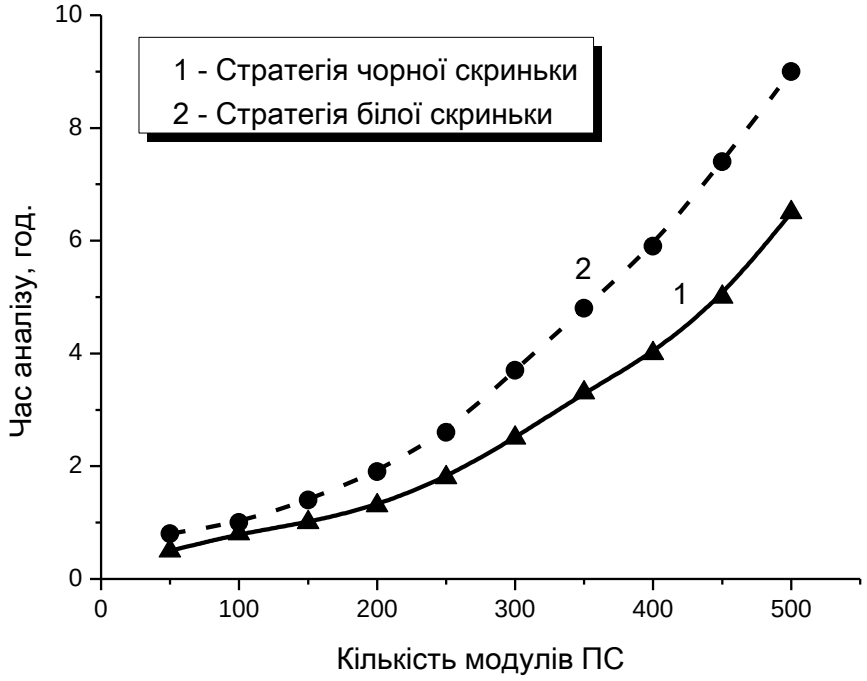


Рис. 4.21. Графік залежності часу попереднього аналізу ПС від кількості модулів для двох стратегій тестування.

2. Визначення характеристик моделі функціонування ПС аналізом коду.

До важливих характеристик моделі належать: кількість модулів, кількість змінних коду ПС, зв'язність модулів. Ці параметри знаходяться за допомогою синтаксичного аналізу програмного коду.

3. Визначення стратегії тестування на основі середньої кількості використаних змінних.

На основі отриманих залежностей (рис. 4.14–4.20) можна стверджувати, що найкритичнішим параметром для формування набору тестів є середня кількість змінних у модулі та середня зв'язність. На основі отриманих результатів можна зробити такі висновки:

- Як видно з рис. 4.16, якщо середня кількість змінних перевищує 30, то тестування за стратегією "чорної скриньки" слабо впливає на

достовірність оцінювання показників надійності ПС і виконується лише для покриття основних сценаріїв та базового функціоналу.

- Якщо середня кількість змінних модуля ПС не більша за 7, то тестування за стратегією "чорної скриньки" є достатньо ефективним і тоді тестування за стратегією "білої скриньки" можна практично не проводити (за винятком написання unit-тестів), оскільки затрати на підготовку середовища тестування та тестових випадків за стратегією "білої скриньки" будуть неефективними.
- Якщо середня кількість змінних модуля ПС більша за 7 і менша за 30, потрібно проводити аналіз інших характеристик і ефективне тестування матиме комбінований характер.

4. *Визначення характеристик помилок моделі з попереднього тестування ПС.*

До прихованих характеристик моделі функціонування ПС відносяться складність помилок та складність їхнього виявлення. Такі характеристики можна оцінити за частотою виникнення відмов та складності відповідного коду.

Перед тестуванням ПС програмний код опрацьовується для того, щоб отримати файли протоколу з інформацією про переходи між модулями ПС та зафіксованими відмовами. На основі цього тестування визначається частота виявлення відмов та складність помилок. Додатково потенційні відмови при певних значеннях змінних можна внести в модель функціонування ПС з аналізу коду (арифметичні помилки, помилки ініціалізації та ін.).

5. *Аналіз тестування за стратегією "чорної скриньки".*

Оскільки в більшості випадків тестування включає стратегії як "білої", так і "чорної скриньки", то необхідно визначити оптимальну кількість тестів для цих стратегій. Із урахуванням емпіричного досвіду тестувальників ПС випливає, що формування та виконання тестів за стратегією "чорної скриньки" є у 10–20 разів менш затратним ніж за стратегією "білої скриньки", але воно і менш ефективне. З урахуванням цього, як правило, спочатку проводиться тестування за стратегією

"чорної скриньки", а потім – "білої". Більшість відмов будуть знайдені ще на етапі менш затратного тестування за стратегією "чорної скриньки", а складніші помилки будуть виявлені на етапі тестування за стратегією "білої скриньки". Очевидно, що тестування стратегією "чорної" скриньки варто проводити до тих пір, поки частота знаходження відмов є достатньо високою в плані ефективності.

Для отримання критерію переходу із етапу тестування за стратегією "чорної скриньки" до стратегії "білої скриньки" досліджували ПС із середньою кількістю модулів $N = 1000$. Для урахування факторів кількості вузлів та складності помилок пропонується наступна формула для граничної кількості знайдених відмов, при якій рекомендовано переходити на тестування за стратегією "білої скриньки" [170]:

$$\lambda < \frac{\lambda_0}{w_{WB}} \frac{\ln(N)}{\ln(1000)\, f_e}, \tag{4.11}$$

де λ_0 – початкова інтенсивність відмов ПЗ, w_{WB} – складність формування тесту за стратегією "білої скриньки" в порівнянні до стратегії "чорної скриньки", що знаходиться емпіричними методами в залежності від ПС, яка розглядається, f_e – складність помилок.

6. Закінчення тестування за стратегією "білої скриньки".

Важливим фактором, який впливає на надійність ПС є складність помилок (рис. 4.18). Для того, щоб гарантувати високий ступінь надійності за наявності відмов, які залежать від кількох змінних та їхніх класів еквівалентності, рекомендується закінчувати тестування за стратегією "білої скриньки" з урахуванням процентного відношення відмов α залежних від трьох та більше змінних (складності помилок):

- якщо $\alpha > 1\%$, то рекомендовано проводити додатковий аналіз коду для визначення складних відмов;
- якщо $\alpha < 1\%$, то рекомендовано проводити тестування за стратегією "чорної скриньки".

Це зумовлено тим, що такий тип помилок є критичний, та їх дуже важко виявити. Крім цього ці помилки можуть бути знайдені лише на етапі тестування

за стратегією "білої скриньки", коли решта "простіших" помилок, як правило, уже виявлені та виправлені.

4.4. Висновки до розділу 4

У цьому розділі наведено засоби підвищення ефективності розробки ПС (зокрема на етапі тестування) на основі розроблених моделей і методів аналізу надійності ПС з урахуванням їх складності.

Наведено опис і властивості критерію достатності процесу тестування ПС, що базується на узагальненій МНПС на основі НПП. Показано, що показник складності, уведений в цій моделі надійності, якісно міняє форму розподілу в процесі тестування ПС, і при переході вхідних даних про відмови до розподілу Пуассона, практично не змінює своє значення. Така властивість показника складності ПС дала змогу використати описаний критерій достатності в якості чисельної міри під час прийняття рішень щодо розподілу ресурсів проекту на етапі тестування ПС. Використання цього критерію при розробці ПС дає змогу зменшити похибку оцінювання залишкової кількості помилок в ПС.

З метою удосконалення процесу прийняття рішення стосовно досягнення заданого рівня надійності ПС розроблено метод підвищення точності визначення кількості помилок на ранніх етапах тестування. Цей метод базується на нелінійному регресійному аналізі прогнозованої кількості помилок, відносно часу припинення тестування. З параметрів рівняння регресії можна отримати новий, уточнений прогноз кількості помилок досліджуваної ПС. Використання описаного методу дає змогу покращити точність прогнозу на 3–5 %, або ж скоротити тривалість тестування в середньому на 30–40%.

Важливим завданням при оцінюванні показників надійності ПС, крім вибору адекватної моделі надійності, є отримання вхідних статистичних даних у вигляді результатів тестування. При цьому, підвищення якості таких вхідних даних (відповідність тестового профілю операційному, максимально повне покриття тестами усіх модулів, гілок, змінних ПС та їх класів еквівалентності

тощо) дає можливість суттєво підвищити достовірність оцінювання показників надійності ПС.

Описано нову модель функціонування ПС, яка враховує її архітектуру та змінні програмного коду, та методи автоматизованого формування сценаріїв тестування, які дають можливість підвищити ефективність процесу тестування ПС, та покращити достовірність оцінювання показників її надійності за рахунок більш повного покриття змінних коду та їх класів еквівалентності тестовими сценаріями. Розроблені методи автоматизованого формування сценаріїв тестування базуються на моделі функціонування ПС та придатні для генерування сценаріїв тестування як у випадку використання за стратегією "чорної скриньки", так і у випадку тестування за стратегією "білої скриньки". Таке автоматизоване формування сценаріїв тестування забезпечує рівномірне покриття коду ПС тестами та дає можливість зменшити часові, фінансові та людські ресурси на етапі тестування ПС. Досліджено вплив характеристик процесу тестування та ПС (таких як кількість тестових сценаріїв, кількість та зв'язність модулів ПС, середня кількість змінних кожного модуля, складність помилок та ін.) на ефективність процесу тестування та достовірність оцінювання показників надійності ПС. Наведено рекомендації стосовно вибору стратегії тестування при розробці ПС в залежності від її складності з метою підвищення ефективності розроблення.

РОЗДІЛ 5. ПРОГРАМНІ ЗАСОБИ ОЦІНЮВАННЯ ТА ПРОГНОЗУВАННЯ ПОКАЗНИКІВ НАДІЙНОСТІ ПРОГРАМНИХ СИСТЕМ

В цьому розділі представлено розроблені засоби оцінювання та прогнозування показників надійності ПС, придатні для використання в процесі їх розроблення. Зокрема спроектовано та реалізовано програмний засіб попереднього опрацювання даних про відмови ПС, інтегрований з системою JIRA, який є препроцесором для моделі надійності з показником складності. Описано програмний засіб для емпіричного оцінювання ймовірностей переходів між модулями ПС, написаної мовою Java, який базується на використанні аспектно-орієнтованого реєстратора виконання ПС. Отримані значення ймовірностей використання модулів ПС (початкового вектору ймовірностей та матриці переходів) використовуються для оцінювання її надійності на основі компонентного підходу. Описано використання розробленого програмного засобу оцінювання показників надійності ПС на основі архітектурних моделей надійності. Розглянуто програмний засіб прогнозування відмов ПС на основі нейронних мереж (НМ). Проведено дослідження впливу параметрів та архітектури НМ на ефективність прогнозування відмов ПС та надано рекомендації, щодо належного вибору параметрів НМ.

Основні результати і висновки, викладені в цьому розділі, опубліковані в працях [200, 252–264].

5.1. Програмний засіб попереднього опрацювання даних про відмови програмних систем

В цьому підрозділі описано реалізацію та застосування програмного засобу попереднього опрацювання даних про відмови ПС, які є вхідною інформацією для моделей надійності типу "чорної скриньки", зокрема МНПС, а також використовуються для отримання показників надійності модулів ПС у випадку використання архітектурних моделей. Цей програмний засіб призначений для організації попереднього опрацювання даних тестування ПС, отриманих з

системи Atlassian JIRA, з метою подальшого оцінювання показників надійності ПС шляхом використання відповідних моделей надійності.

Основними функціональними можливостями програмного засобу є:

- Автоматизоване отримання результатів тестування з систем JIRA (наприклад з https://jira.springsource.org);
- Опрацювання результатів;
  - Створення запису про відмови ПС;
  - Видалення запису про відмови ПС;
  - Нормування результатів за кількістю тестерів;
  - Розбиття статистики відмов на задану кількість часових інтервалів;
  - Опрацювання статистики за типами відмов (критичні, блокуючі, тривіальні тощо);
  - Обчислення кумулятивної кількості відмов;
- Формування звітів;
  - Сортування та фільтрування при відображенні таблиці з результатами відмов ПС;
  - Виведення графічної залежності відмов ПС від часу;
  - Збереження звіту у файл;
- Вивід сформованих звітів.

Програмний засіб написано на мові Java з використанням середовища розробки Eclipse Kepler. Інтеграцію з системою JIRA здійснено за допомогою JIRA REST API.

Відповідно до поставлених вимог розроблено архітектуру програмного модуля для здійснення інтеграції з системою обліку виявлених відмов ПС, яку зображено на рис. 5.1.

- *Модуль аналізу запитів користувача, керування процесом імпортування та інтеграції з системою обліку помилок ПС* – основний модуль, який призначений для керування процесом

інтеграції з системою обліку помилок, а також виконує роль головного модуля, за допомогою якого здійснюється інтеграція між усіма модулями.

- *Модуль інтеграції з JIRA REST API* – призначений для здійснення інтеграції з REST API системи Atlassian JIRA. Виконує функції аутентифікації та авторизації користувача в JIRA, виконання запитів, їх обробку та конвертування з JSON в об’єкти Java.

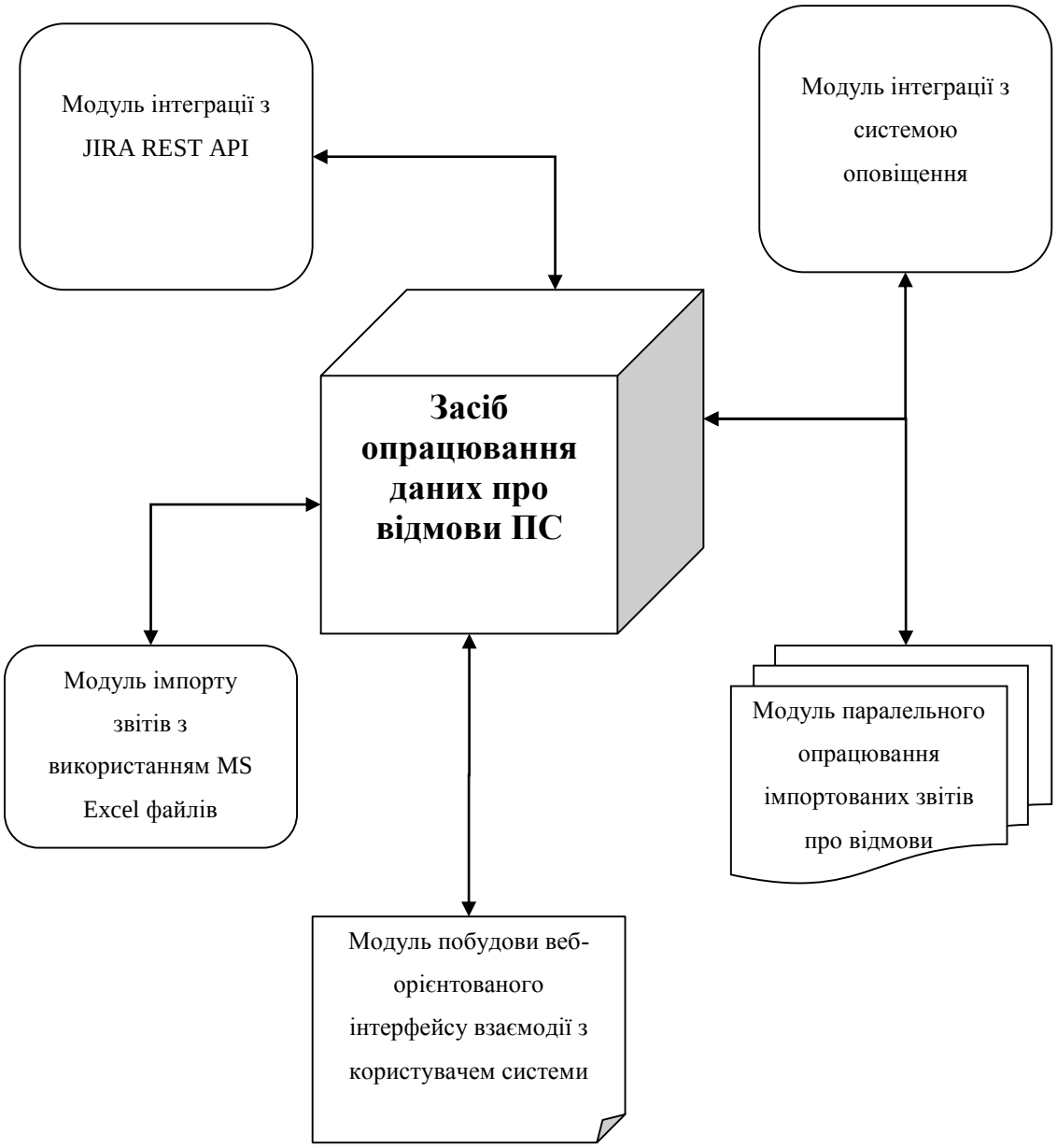


Рис. 5.1. Загальна архітектура засобу опрацювання даних про відмови ПС.

- *Модуль інтеграції з системою оповіщення* – здійснює інтеграцію в загальну підсистему подій та оповіщення в програмний засіб аналізу та прогнозування надійності ПС.
- *Модуль імпорту звітів з використанням MS Excel файлів* – забезпечує можливість імпортування звітів про виявлені помилки в роботі ПС за допомогою файлів в форматі MS Excel.
- *Модуль паралельного опрацювання імпортованих звітів про помилки* – призначений для розпаралелення процесу аналізу імпортованих звітів, що дозволяє пришвидшити роботу модуля імпортування даних.
- *Модуль побудови веб-орієнтованого інтерфейсу взаємодії з користувачем системи* – за допомогою даного модуля здійснюється процес взаємодії користувача з системою під час імпортування звітів про відмови ПС.

Для інтеграції з системою Atlassian JIRA розроблено декілька основних функціональних елементів:

1. Авторизації користувача в системі JIRA.
2. Формування пошукового запиту для системи.
3. Формування та перевірка HTTP запиту для отримання даних.
4. Виконання запиту.
5. Отримання результату, його перевірка та інтерпретація.
6. Передача отриманих звітів в модуль керування.

На рис. 5.2 продемонстровано архітектуру засобу інтеграції з Atlassian JIRA.

- *Модуль керування процесом інтеграції з JIRA* – здійснює процес управління та делегування запитів при інтеграції з системою обліку помилок.
- *Модуль аутентифікації та авторизації в Atlassian JIRA* – призначений для здійснення аутентифікації та авторизації

користувача в системі обліку помилок, визначення його прав доступу до визначеного проекту.

- *Модуль виконання HTTP запитів* – забезпечує коректне формування та виконання HTTP запитів.
- *Модуль формування пошукових запитів* – виконує формування пошукових запитів відповідно до вимог JIRA Search REST API.
- *Модуль конвертування JSON* – призначений для конвертування Java об’єктів в JSON та навпаки.

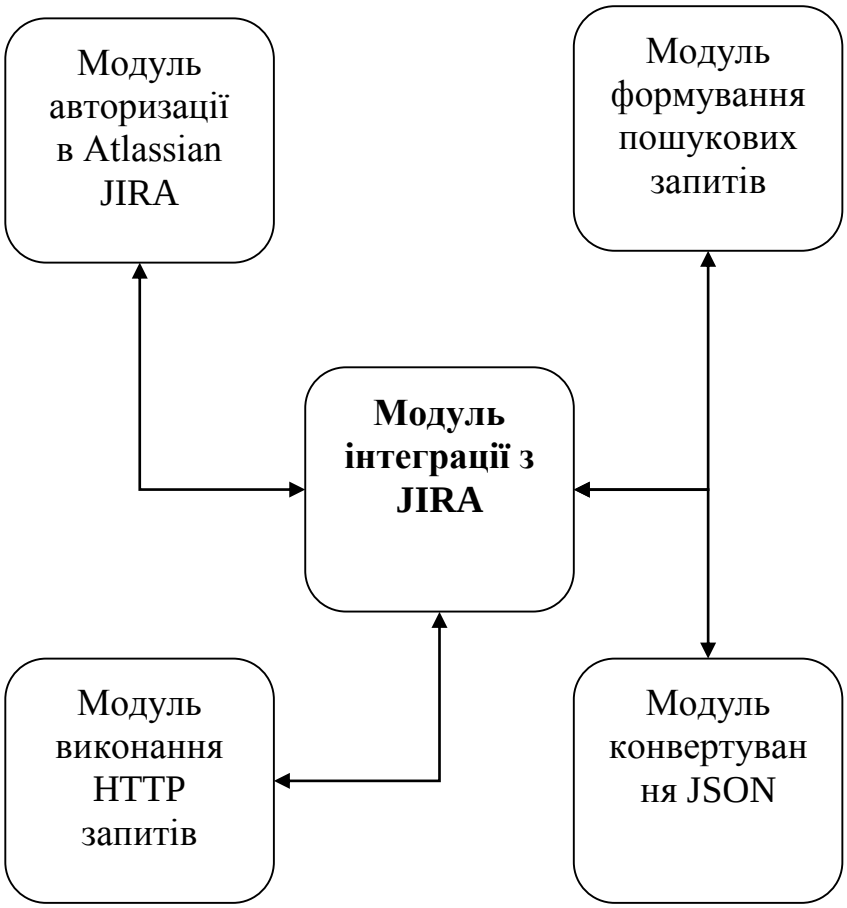


Рис. 5.2. Архітектура модуля інтеграції з Atlassian JIRA.

Засіб опрацювання даних про відмови ПС розроблено відповідно до об’єктно-орієнтованого підходу. Розроблено основні класи модуля та взаємозв’язки між ними. На рис. 5.3 наведено діаграму класів розробленого програмного засобу.

Основними класами цього засобу є:

- *JiraService* (рис. 5.4) – інтерфейс яким описано основні функціональні можливості сервісу.
  - *getIssueByProject* – метод повинен надавати можливість отримати звіти про помилки для заданого проекту;
  - *importIssueForProject* – метод надає можливість імпортування звітів про помилки для заданого проекту;

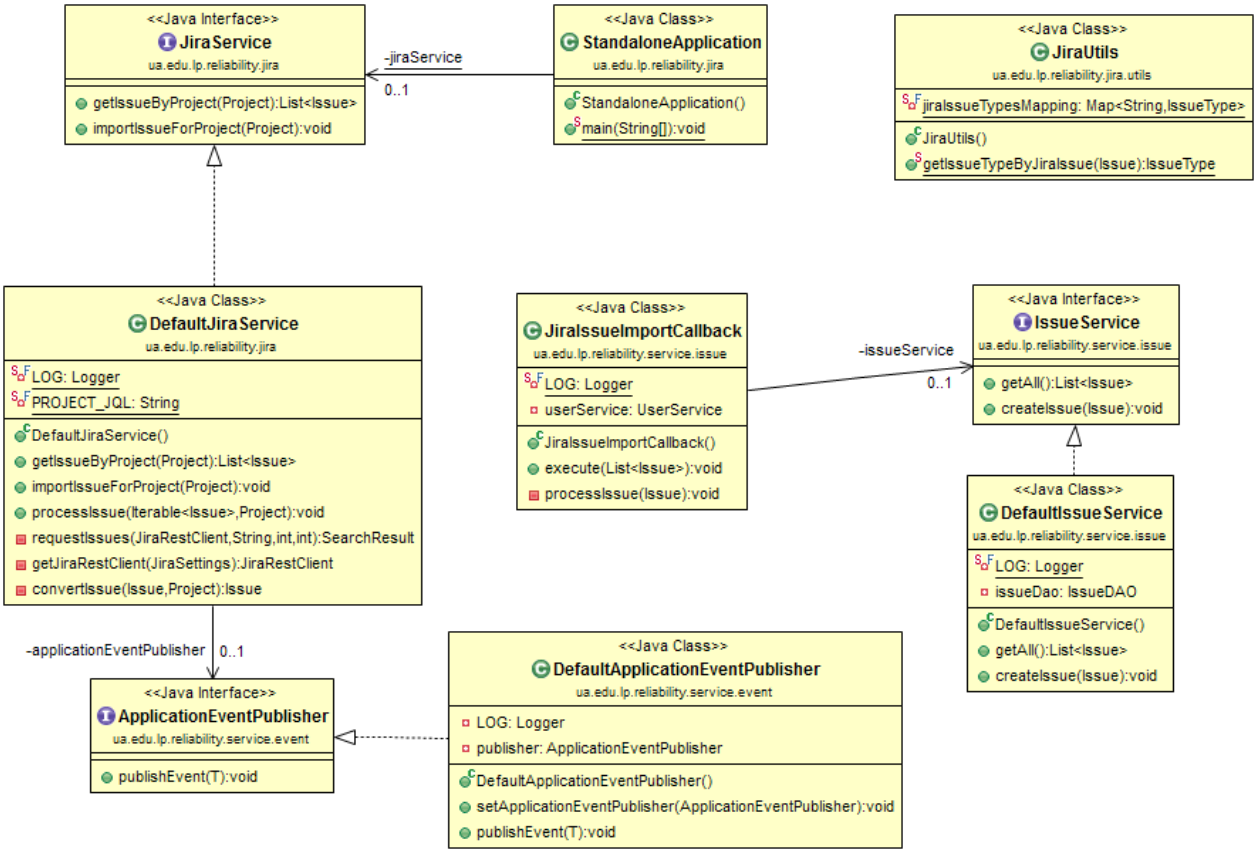


Рис. 5.3. UML діаграма класів засобу опрацювання даних про відмови ПС.

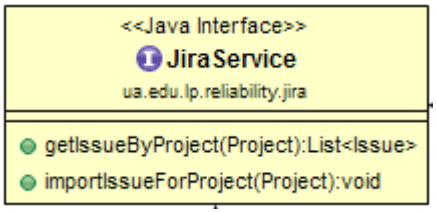


Рис. 5.4. Інтерфейс JiraService.

- DefaultJiraService – клас (рис. 5.5) який реалізовує інтерфейс JiraService та власне реалізовує функціональні вимоги до сервісу.

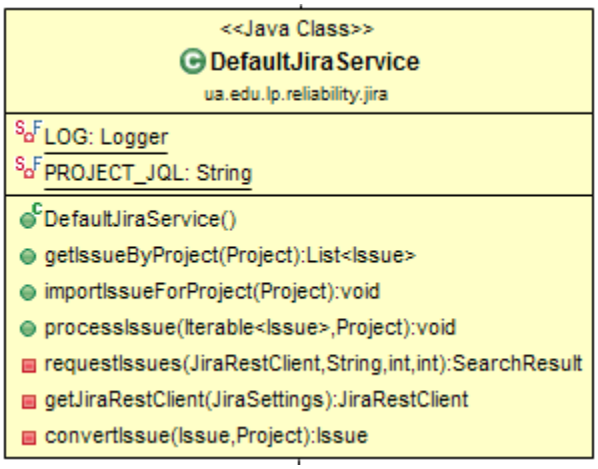


Рис. 5.5. Структура класу DefaultJiraService.

- StandaloneApplication – клас (рис. 5.6), що надає можливість виконання цього модуля автономно від програмного засобу аналізу та прогнозування надійності ПС.

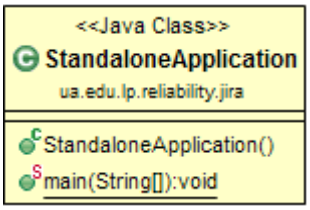


Рис. 5.6. Структура класу StandaloneApplication.

- JiraUtils – клас (рис. 5.7) реалізовує допоміжні функції, які необхідні в процесі імпортування та конвертації звітів про відмови.

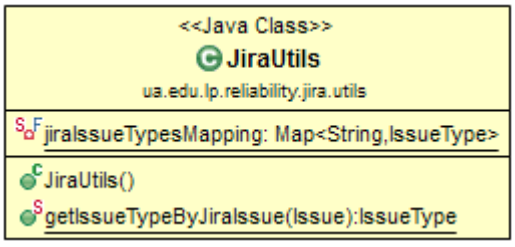


Рис. 5.7. Структура класу JiraUtils.

- ApplicationEventPublisher – інтерфейс (рис. 5.8), що описує систему подій та оповіщень.

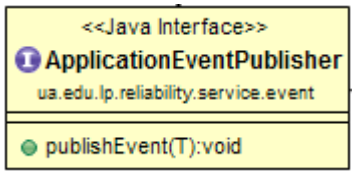


Рис. 5.8. Структура інтерфейсу ApplicationEventPublisher

- DefaultApplicationEventPublisher – реалізація (рис. 5.9) подій та оповіщення в системі аналізу та прогнозування надійності ПС.

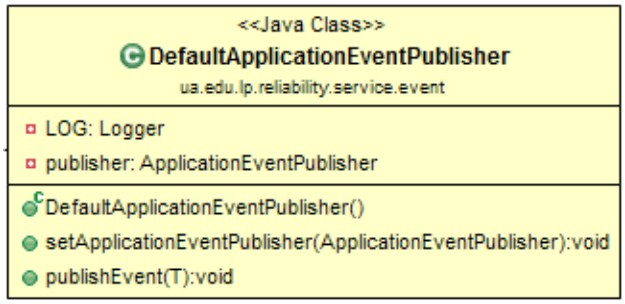


Рис. 5.9. Структура класу DefaultApplicationEventPublisher.

- JiraIssueImportCallback – клас (рис. 5.10) призначений для опрацювання результатів отриманих з запиту JIRA REST API.

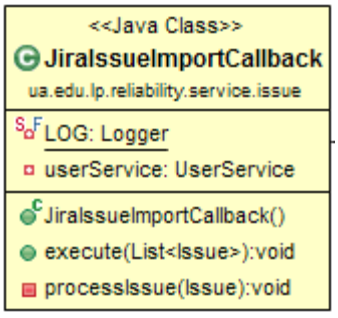


Рис. 5.10. Структура класу JiraIssueImportCallback.

- *IssueService* – призначений для проведення аналізу та опрацювання звітів про помилки ПС (рис. 5.11).

Рис. 5.11. Структура інтерфейсу IssueService.

Для збереження інформації про відмови ПС використовується реляційна база даних (БД). На рис. 5.12 продемонстровано ER-діаграму розробленої БД.

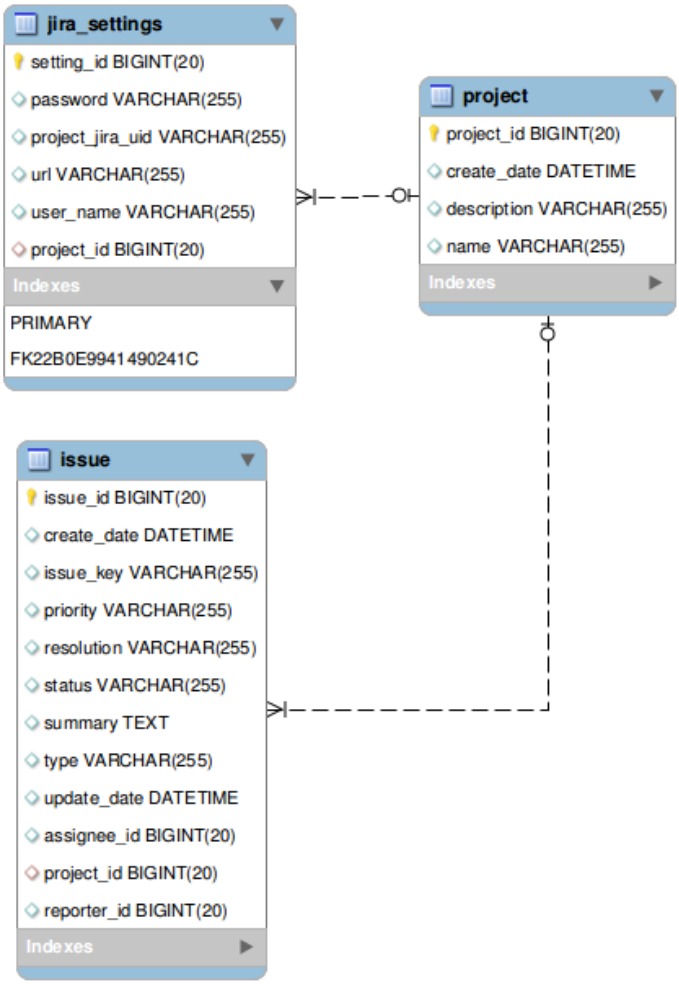


Рис. 5.12. ER-діаграма архітектури бази даних.

Таблиця jira_settings – призначена для збереження інформації про дані користувача та адресу сервера з системою JIRA для конкретного проекту в системі аналізу та прогнозування надійності ПС (рис. 5.13).

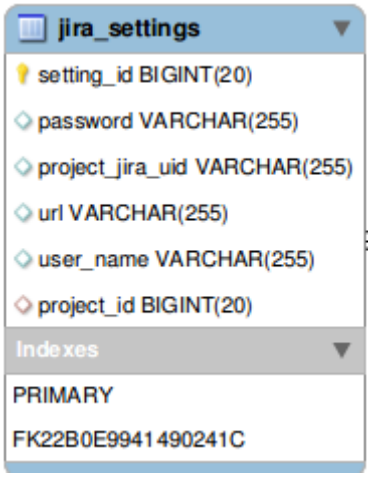


Рис 5.13. Структура таблиці jira_settings.

Основними полями цієї таблиці є:

- setting_id – унікальний ідентифікатор запису налаштувань, первинний ключ таблиці;
- user_name – назва користувач JIRA;
- password – пароль доступу користувача;
- url – адреса сервера JIRA;
- project_id – ідентифікатор проекту для якого призначені налаштування, зовнішній ключ таблиці.

Таблиця ptoject (рис. 5.14) призначена для зберігання інформації про проект для якого здійснюється імпортування звітів про відмови ПС, а також аналіз та прогнозування її надійності;

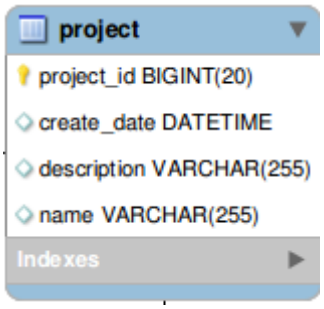


Рис. 5.14. Структура таблиці project.

Основними полями таблиці project є:

- project_id – унікальний ідентифікатор проекту в системі, первинний ключ таблиці;
- name – назва проекту;
- description – опис проекту;
- create_date – дата створення.

Таблиця issue (рис. 5.15) призначена для зберігання інформації про відмови ПС.

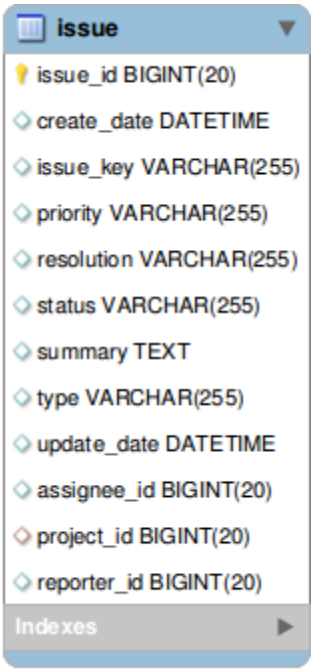


Рис. 5.15. Структура таблиці issue.

Основними полями таблиці issue є:

- issue_id – унікальний ідентифікатор відмови в системі, первинний ключ таблиці;
- create_date – дата створення запису про відмову;
- issue_key – ідентифікатор відмови в системі обліку відмов ПС;
- priority – пріоритет відмови;
- resolution – висновок;
- status – статус виявленої відмови;
- summary – опис відмови;
- type – тип відмови;
- update_date – дата останнього оновлення інформації про відмову;
- assignee_id – зовнішній ключ таблиці, посилання на користувача, який буде аналізувати виявлену відмову;
- reporter_id – зовнішній ключ таблиці, посилання на користувача, що виявив відмову;
- project_id – зовнішній ключ, посилання на проект в якому буде здійснюватись аналіз виявленої відмови.

На рис. 5.16 наведено вигляд головного вікна розробленого програмного засобу опрацювання даних про відмови ПС.

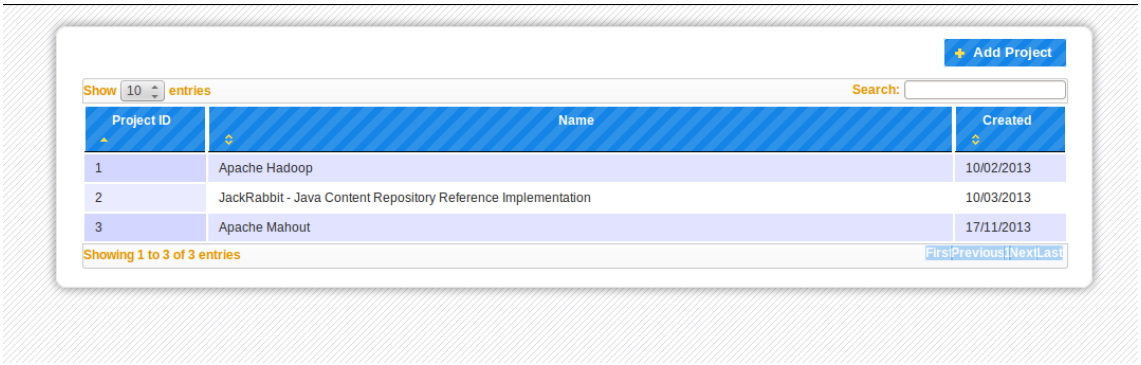


Рис. 5.16. Головне вікно засобу опрацювання даних про відмови ПС.

Вікно перегляду існуючих записів звітів про відмови та меню можливих дій зображено на рис. 5.17.

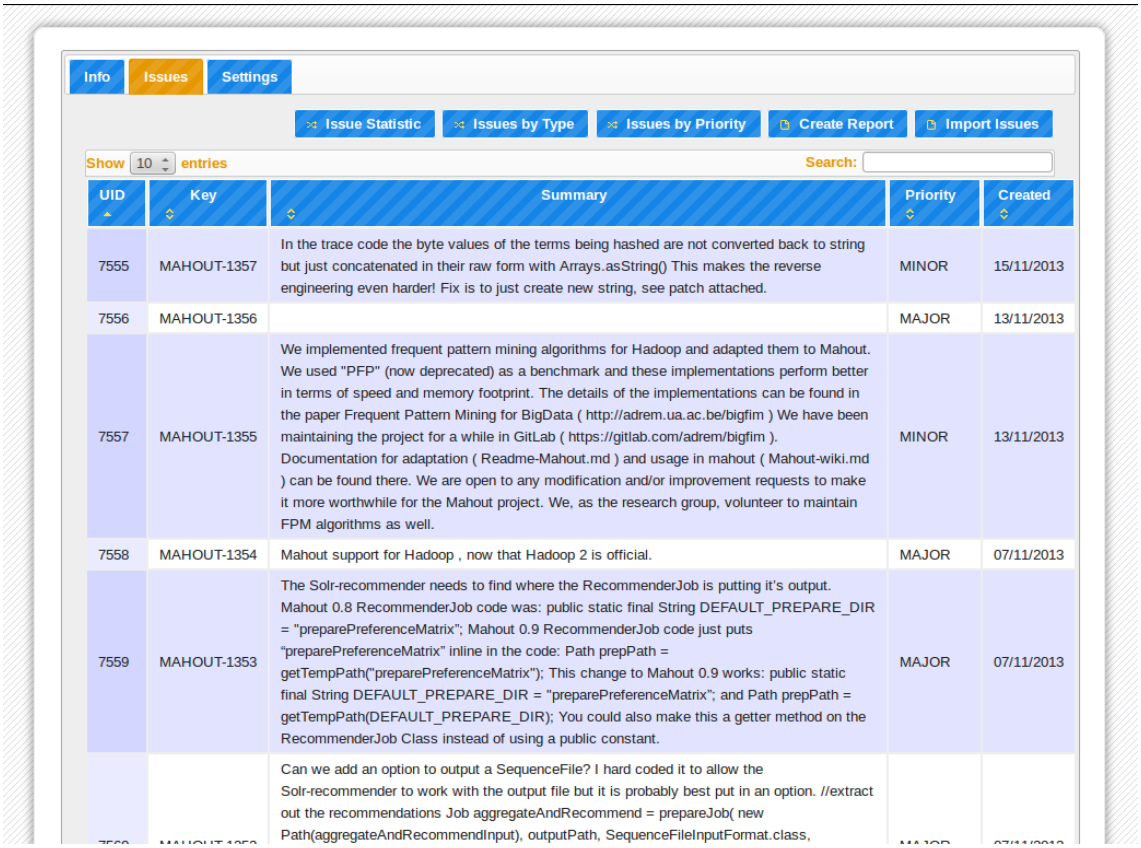
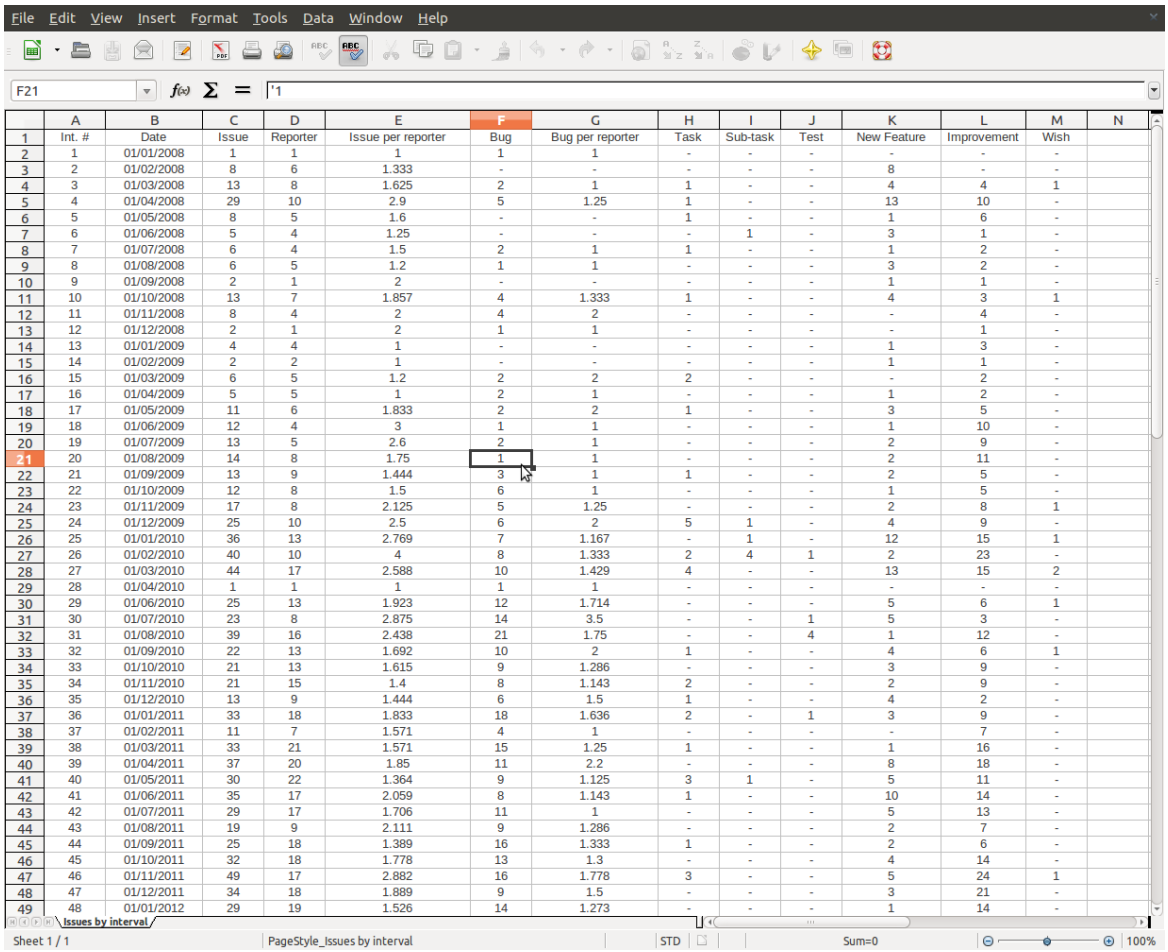


Рис. 5.17. Список звітів про відмови ПС.

Засіб передбачає експорт звіту про відмови ПС у форматі MS Excel. Вигляд такого звіту наведено на рис. 5.18.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Int. #	Date	Issue	Reporter	Issue per reporter	Bug	Bug per reporter	Task	Sub-task	Test	New Feature	Improvement	Wish	
2	1	01/01/2008	1	1	1	1	1	-	-	-	-	-	-	
3	2	01/02/2008	8	6	1.333	-	-	-	-	-	8	-	-	
4	3	01/03/2008	13	8	1.625	2	1	1	-	-	4	4	1	
5	4	01/04/2008	29	10	2.9	5	1.25	1	-	-	13	10	-	
6	5	01/05/2008	8	5	1.6	-	-	1	-	-	1	6	-	
7	6	01/06/2008	5	4	1.25	-	-	-	1	-	3	1	-	
8	7	01/07/2008	6	4	1.5	2	1	1	-	-	1	2	-	
9	8	01/08/2008	6	5	1.2	1	1	-	-	-	3	2	-	
10	9	01/09/2008	2	1	2	-	-	-	-	-	1	1	-	
11	10	01/10/2008	13	7	1.857	4	1.333	1	-	-	4	3	1	
12	11	01/11/2008	8	4	2	4	2	-	-	-	-	4	-	
13	12	01/12/2008	2	1	2	1	1	-	-	-	-	1	-	
14	13	01/01/2009	4	4	1	-	-	-	-	-	1	3	-	
15	14	01/02/2009	2	2	1	-	-	-	-	-	1	1	-	
16	15	01/03/2009	6	5	1.2	2	2	2	-	-	-	2	-	
17	16	01/04/2009	5	5	1	2	1	-	-	-	1	2	-	
18	17	01/05/2009	11	6	1.833	2	2	1	-	-	3	5	-	
19	18	01/06/2009	12	4	3	1	1	-	-	-	1	10	-	
20	19	01/07/2009	13	5	2.6	2	1	-	-	-	2	9	-	
21	20	01/08/2009	14	8	1.75	1	1	-	-	-	2	11	-	
22	21	01/09/2009	13	9	1.444	3	1	1	-	-	2	5	-	
23	22	01/10/2009	12	8	1.5	6	1	-	-	-	1	5	-	
24	23	01/11/2009	17	8	2.125	5	1.25	-	-	-	2	8	1	
25	24	01/12/2009	25	10	2.5	6	2	5	1	-	4	9	-	
26	25	01/01/2010	36	13	2.769	7	1.167	-	1	-	12	15	1	
27	26	01/02/2010	40	10	4	8	1.333	2	4	1	2	23	-	
28	27	01/03/2010	44	17	2.588	10	1.429	4	-	-	13	15	2	
29	28	01/04/2010	1	1	1	1	1	-	-	-	-	-	-	
30	29	01/06/2010	25	13	1.923	12	1.714	-	-	-	5	6	1	
31	30	01/07/2010	23	8	2.875	14	3.5	-	-	1	5	3	-	
32	31	01/08/2010	39	16	2.438	21	1.75	-	-	4	1	12	-	
33	32	01/09/2010	22	13	1.692	10	2	1	-	-	4	6	1	
34	33	01/10/2010	21	13	1.615	9	1.286	-	-	-	3	9	-	
35	34	01/11/2010	21	15	1.4	8	1.143	2	-	-	2	9	-	
36	35	01/12/2010	13	9	1.444	6	1.5	1	-	-	4	2	-	
37	36	01/01/2011	33	18	1.833	18	1.636	2	-	1	3	9	-	
38	37	01/02/2011	11	7	1.571	4	1	-	-	-	-	7	-	
39	38	01/03/2011	33	21	1.571	15	1.25	1	-	-	1	16	-	
40	39	01/04/2011	37	20	1.85	11	2.2	-	-	-	8	18	-	
41	40	01/05/2011	30	22	1.364	9	1.125	3	1	-	5	11	-	
42	41	01/06/2011	35	17	2.059	8	1.143	1	-	-	10	14	-	
43	42	01/07/2011	29	17	1.706	11	1	-	-	-	5	13	-	
44	43	01/08/2011	19	9	2.111	9	1.286	-	-	-	2	7	-	
45	44	01/09/2011	25	18	1.389	16	1.333	1	-	-	2	6	-	
46	45	01/10/2011	32	18	1.778	13	1.3	-	-	-	4	14	-	
47	46	01/11/2011	49	17	2.882	16	1.778	3	-	-	5	24	1	
48	47	01/12/2011	34	18	1.889	9	1.5	-	-	-	3	21	-	
49	48	01/01/2012	29	19	1.526	14	1.273	-	-	-	1	14	-	

Рис. 5.18. Приклад експортованого звіту про відмови ПС.

5.2. Програмний засіб для оцінювання ймовірностей переходів між модулями програмної системи

В цьому підрозділі описано реалізацію та застосування програмного засобу, який дає змогу реєструвати в журнал шляхи і тривалість виконання модулів ПС, написаної мовою Java. Результатом роботи цього засобу є матриця оцінок імовірностей переходів між модулями ПС та початковий вектор розподілу, які є вхідною інформацією для архітектурних моделей надійності ПС разом з показниками надійності модулів ПС. Програмний засіб оцінювання ймовірностей переходів між модулями ПС призначений для ведення протоколу виконання програми та статистичного підрахунку використання модулів ПС, зберігання та візуалізації отриманих даних з метою подальшого їх використання для розрахунку надійності досліджуваної ПС.

Основними функціональними вимогами до програмного засобу є:

- введення списку або групи модулів, які необхідно відслідковувати;
- запис: часу початку, завершення та тривалості виконання; ідентифікатора модуля, який виконує звертання, та ідентифікатора модуля, до якого звертаються;
- дані необхідно зберігати в загальному сховищі для подальшого аналізу;
- необхідно розробити зручний інтерфейс доступу та перегляду статистики виконання модулів.

Враховуючи постановку задачі та функціональні вимоги, було прийнято рішення використати елементи аспектно-орієнтованого програмування, а реалізацію програмного засобу здійснити мовою програмування Java за допомогою середовища Spring Tools Suite. Модуль доступу та візуалізації результатів роботи програми реалізовано у вигляді веб-додатку, для можливості отримання зручного доступу до даних з будь-якої платформи.

Для забезпечення описаних специфікацій було спроектовано і реалізовано наступні функціональні одиниці:

- засіб для реєстрації, який буде записувати результати в БД;
- веб-додаток, для перегляду протоколу і статистики виконання ПС;
- спільний фреймворк для засобу реєстрації та веб-додатку перегляду результатів.

Архітектуру програмного засобу оцінювання ймовірностей переходів між модулями ПС представлено на рис. 5.19.

1. Аспектно-орієнтований реєстратор – призначений для здійснення запису потоку виконання ПС за допомогою фреймворку реєстрації. На рис. 5.20 продемонстровано архітектуру такого реєстратора.

Аспектно-орієнтований реєстратор складається з:

- *модуля взаємодії з фреймворком реєстрації та аналізу виконання ПС* – даний модуль призначений для виконання операцій з записом

статистики виконання в сховище даних за допомогою фреймворку реєстрації та аналізу виконання ПС.

- *модуля зчитування та запису налаштувань реєстратора* – призначений для зчитування налаштувань, до яких входять: шаблон пакету, модулі якого будуть модифікуватись та реєструватись, налаштування підключення до сховища даних;
- *файлу з налаштуваннями реєстратора* – тут зберігаються налаштування, необхідні під час роботи реєстратора;
- *модуля завантаження класів для модифікації* – даний модуль здійснює завантаження класів для модифікації байт-коду при завантаженні їх віртуальною машиною Java (JVM) для виконання в програмі.
- *модуля модифікації байт*-коду – призначений для зміни байт-коду модуля, здійснює додавання програмного коду для реєстрації в класи, які необхідно аналізувати.

Рис. 5.19. Загальна архітектура засобу оцінювання ймовірностей переходів між модулями ПС.

2. Фреймворк для реєстрації виконання ПС та аналізу результатів – призначений для здійснення збереження результатів реєстрації в сховищі даних та надання спільного API для реєстрації та аналізу результатів. На рис. 5.21 наведено архітектуру такого фреймворку.

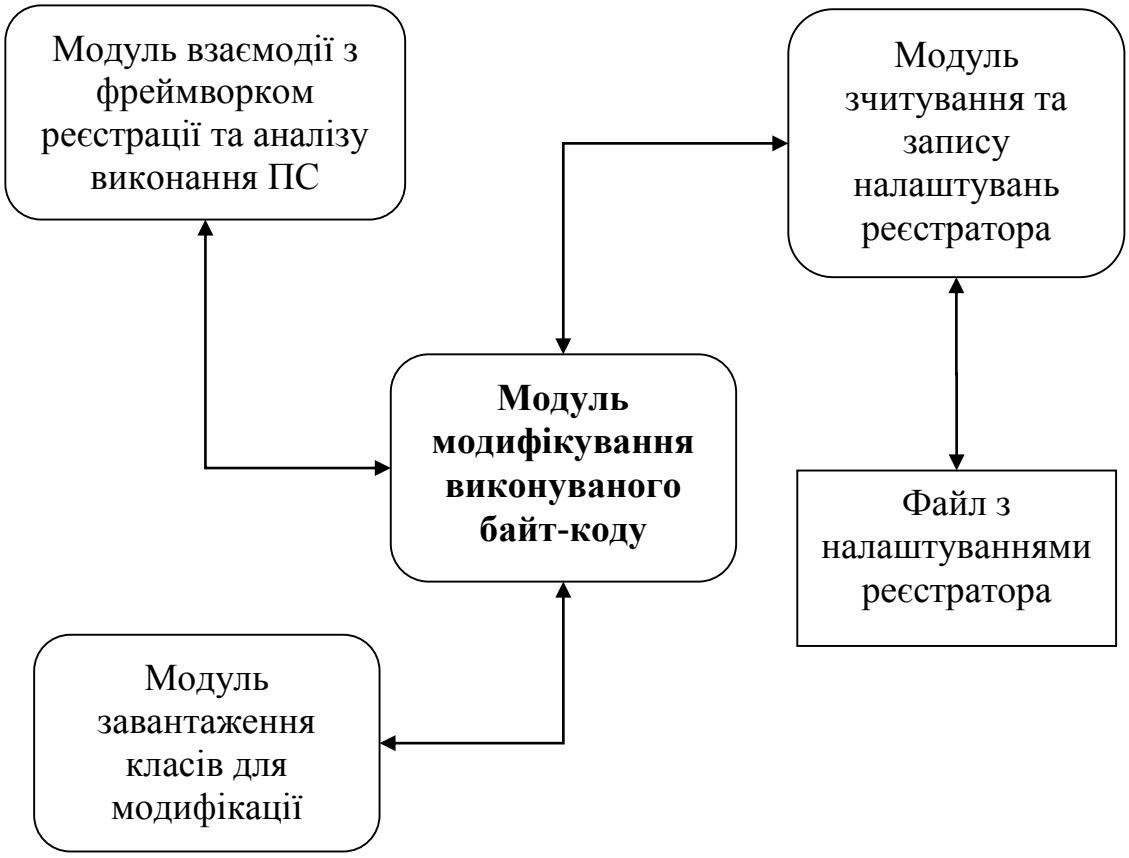


Рис. 5.20. Архітектура аспектно-орієнтованого реєстратора.

Фреймворк для реєстрації виконання ПС та аналізу результатів складається з наступних модулів:

- *модуль збереження результатів реєстрації* – призначений для забезпечення збереження результатів реєстрації;
- *модуль взаємодії з сховищем даних* – призначений для взаємодії з сховищем даних (БД);
- *модуль формування звітів реєстрації* – відповідає за формування звітів реєстрації, таких як: статистика виконання модулів ПС, статистика по окремих модулях тощо;
- *модуль аналізу результатів реєстрації* – даний модуль здійснює аналіз результатів реєстрації виконання ПС.
- *модуль управління запитами* – призначений для прийняття запитів від інших програмних засобів та їх делегування іншим модулям;

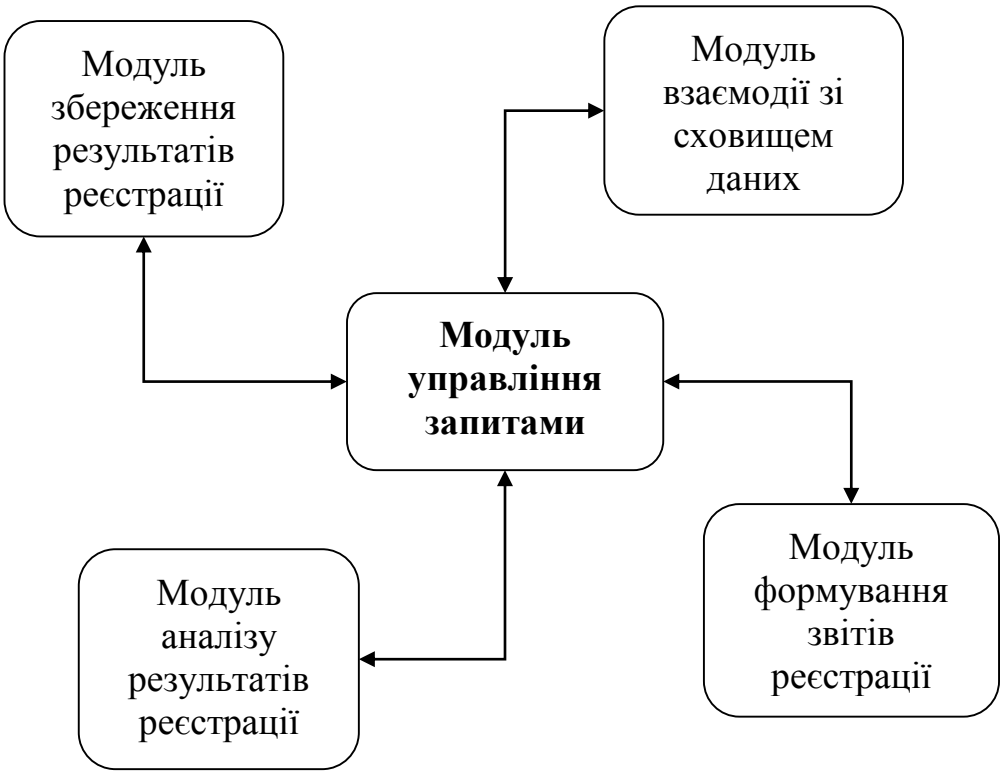


Рис. 5.21. Архітектура фреймворку для реєстрації виконання ПС та аналізу результатів.

3. Веб-додаток – призначений для зручного багатокористувацького та платформонезалежного використання та аналізу результатів реєстрації виконання ПС. На рис. 5.22 продемонстровано архітектуру такого веб-додатку.

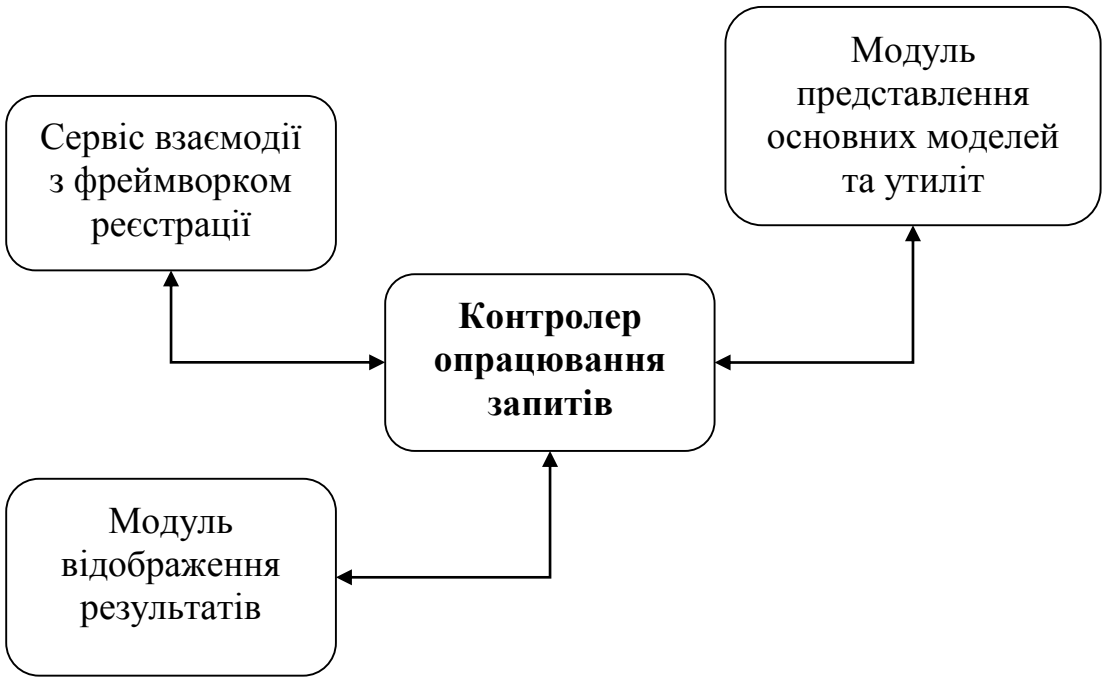


Рис. 5.22. Архітектура веб-додатку.

Веб-додаток складається з наступних елементів:

- *сервіс взаємодії з фреймворком реєстрації* – призначений для забезпечення взаємодії з фреймворком реєстрації результатів виконання ПС;
- *модуль представлення основних моделей та утиліт* – забезпечує представлення основних моделей ПС та утиліт для виконання різних допоміжних операцій;
- *модуль відображення результатів* – відповідає за формування сторінок та відображення їх користувачам веб-додатку;
- *контролер опрацювання запитів* – призначений для прийняття запитів від користувачів та їх делегування іншим модулям.

В якості сховища даних використовуватиметься реляційна БД, яка складається з таблиці log_item, структуру якої наведено на рис. 5.23.

Призначення полів таблиці log_item наступне:

- callerClassUID – унікальна назва класу, що здійснює запит на виконання;

- callerClassMethod – назва методу класу, з якого здійснюється запит на виконання;
- currentClassUID – унікальна назва класу, що здійснює виконання;
- currentClassMethod – назва методу класу, що здійснюється виконання;
- startDate – початок виконання;
- endDate – кінець виконання;
- duration – тривалість виконання в мілісекундах.

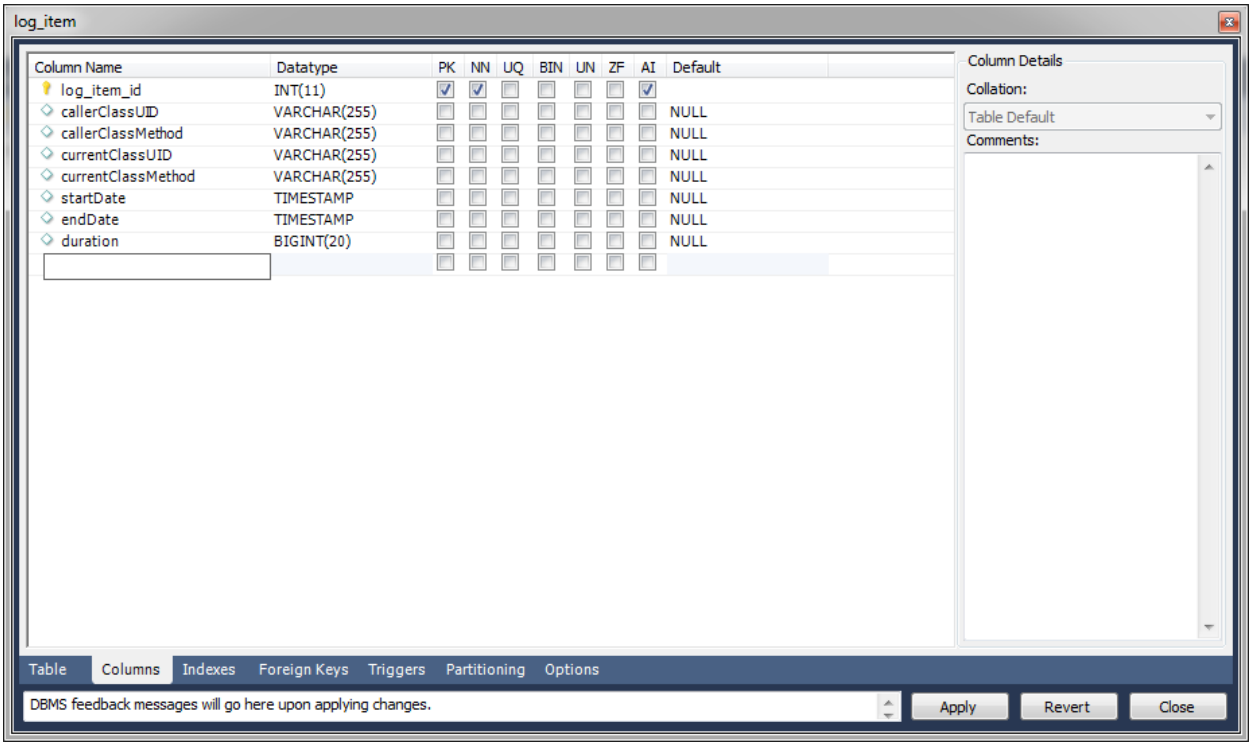


Рис. 5.23. Структура таблиці log_item.

Для перегляду результатів використовується веб-додаток, головний інтерфейс якого складається з двох частин: меню та таблиці з результатами реєстрації (рис. 5.24).

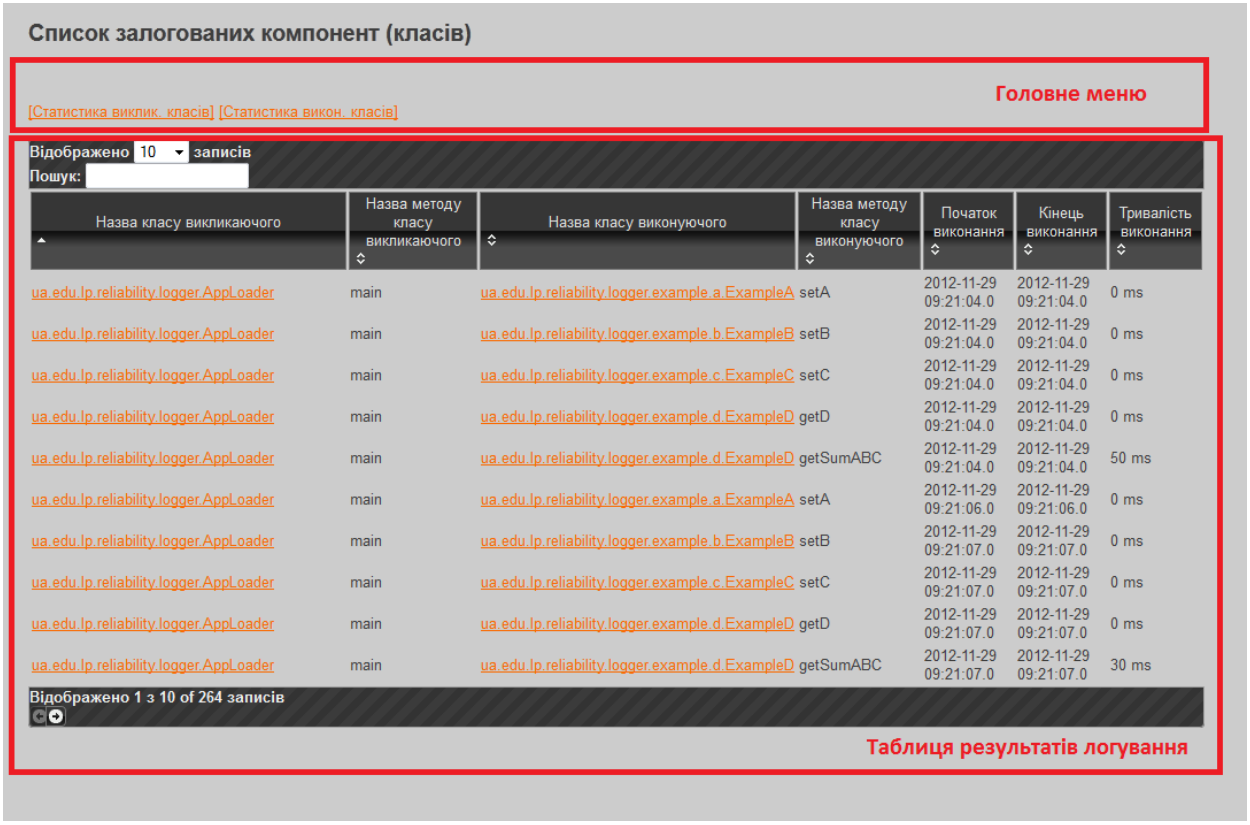


Рис. 5.24. Структура інтерфейсу веб-додатку.

Фреймворк для реєстрації виконання ПС та аналізу результатів, а також веб-додаток розроблено з використанням об'єктно-орієнтованої парадигми. Діаграму класів фреймворку для реєстрації виконання ПС та аналізу результатів зображено на рис. 5.25, а на рис. 5.26 наведено діаграму класів веб-додатку.

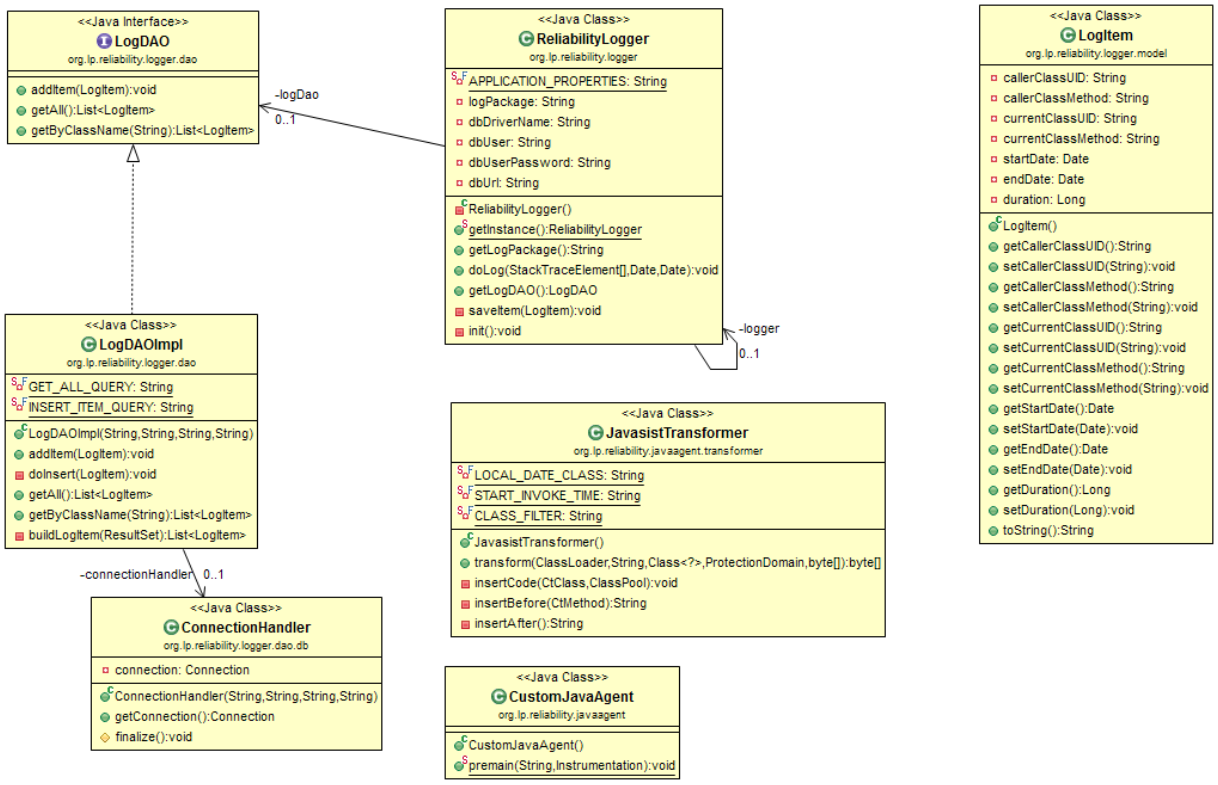


Рис. 5.25. Діаграма класів фреймворку реєстрації виконання ПС та аналізу результатів.

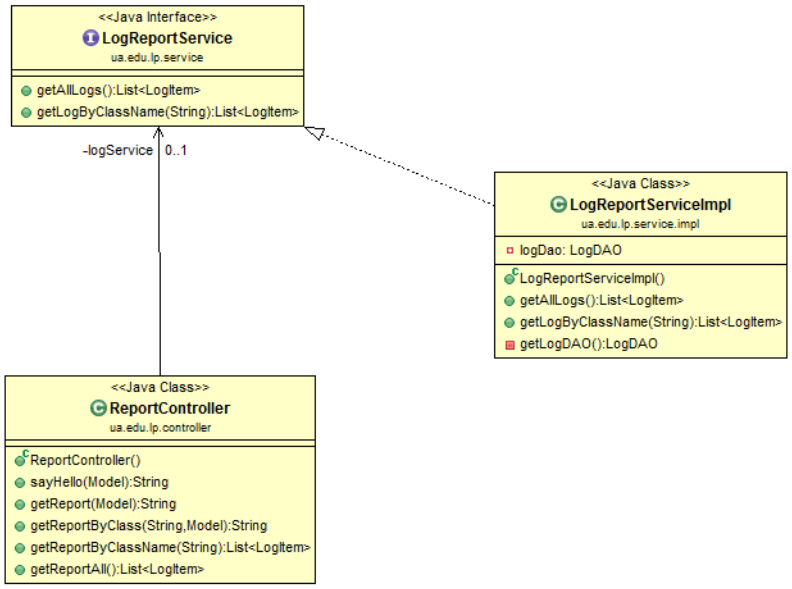


Рис. 5.26. Діаграма класів веб-додатку для візуалізації та аналізу результатів.

5.3. Програмний засіб оцінювання надійності програмних систем "СОН ПЗ"

В цьому підрозділі наведено опис програмного засобу оцінювання надійності ПС на основі архітектурних моделей "СОН ПЗ" [200]. Цей засіб складається з наступних функціональних блоків:

- модуль для отримання моделі функціонування ПС;
- модуль АФСТ на основі моделі функціонування ПС;
- модуль для оцінювання показників надійності ПС з використанням ЛМВП.

В роботі модуля побудови моделі функціонування ПС використовується засіб оцінювання ймовірностей переходів між модулями програмної системи, описаний в підрозділі 5.2, а в роботі модуля оцінювання показників надійності ПС з використанням ЛМВП використовуються результати роботи засобу попереднього опрацювання даних про відмови ПС, описаного в підрозділі 5.1. Інформація про передавання управління між модулями досліджуваної ПС та тривалість їх виконання зберігається в БД.

Призначення модуля створення моделі функціонування ПС – отримання інформації про передавання управління між відповідними модулями для знаходження матриці оцінок ймовірностей переходів та часу виконання модулів ПС, за допомогою засобу реєстрації, а також аналіз коду досліджуваної ПС для визначення змінних кожного модуля та їх правильних і неправильних класів еквівалентності. Така модель функціонування ПС є основою як для автоматизованого формування набору сценаріїв тестування, так і для оцінювання показників надійності ПС.

Модуль АФСТ ПС забезпечує генерацію сценаріїв тестування для досліджуваної ПС за стратегією "чорної" та "білої скриньки" з урахуванням метрики покриття значень параметрів коду.

Функціями модуля оцінювання показників надійності ПС є визначення порядку марковського процесу, який використовується для моделювання

характеру взаємодії модулів ПС, а також оцінювання імовірності безвідмовної роботи та інтенсивності відмов ПС на основі отриманих даних з використанням ЛМВП.

Поведінка програмного засобу оцінювання надійності ПС на основі архітектурних моделей "СОН ПЗ" моделюється наступним чином.

Для створення моделі функціонування ПС здійснюються наступні кроки (рис. 5.27):

1) декомпозиція ПС на модулі в залежності від кількості можливих вузлів у графі, на основі аналізу коду. Для оцінювання надійності ПС, як правило, модулями є пакети у зв’язку із розмірністю матриці ймовірностей передавання управління між модулями, а для АФСТ ПС – методи для більш ефективного формування тестових сценаріїв;
2) використовуючи аналіз коду, на другому кроці визначається множина змінних, що використовуються та змінюється відповідним модулем C_i у програмній системі;
3) для отримання оцінок ймовірностей передачі управління між модулями застосовується реєстрація виконання ПС. Для цього зчитується код ПС та здійснюється пошук початку та кінця кожного модуля, куди додаються повідомлення для побудови файлу протоколу;
4) після додавання необхідних повідомлень ПС компілюється, та починається процедура її використання. Крім цього у файл протоколу записується тривалість виконання кожного модуля;
5) останнім кроком є аналіз файлу протоколу для визначення оцінок ймовірностей передавання управління між модулями, а також для отримання додаткової інформації про відмови, зафіксовані у протоколі виконання ПС.

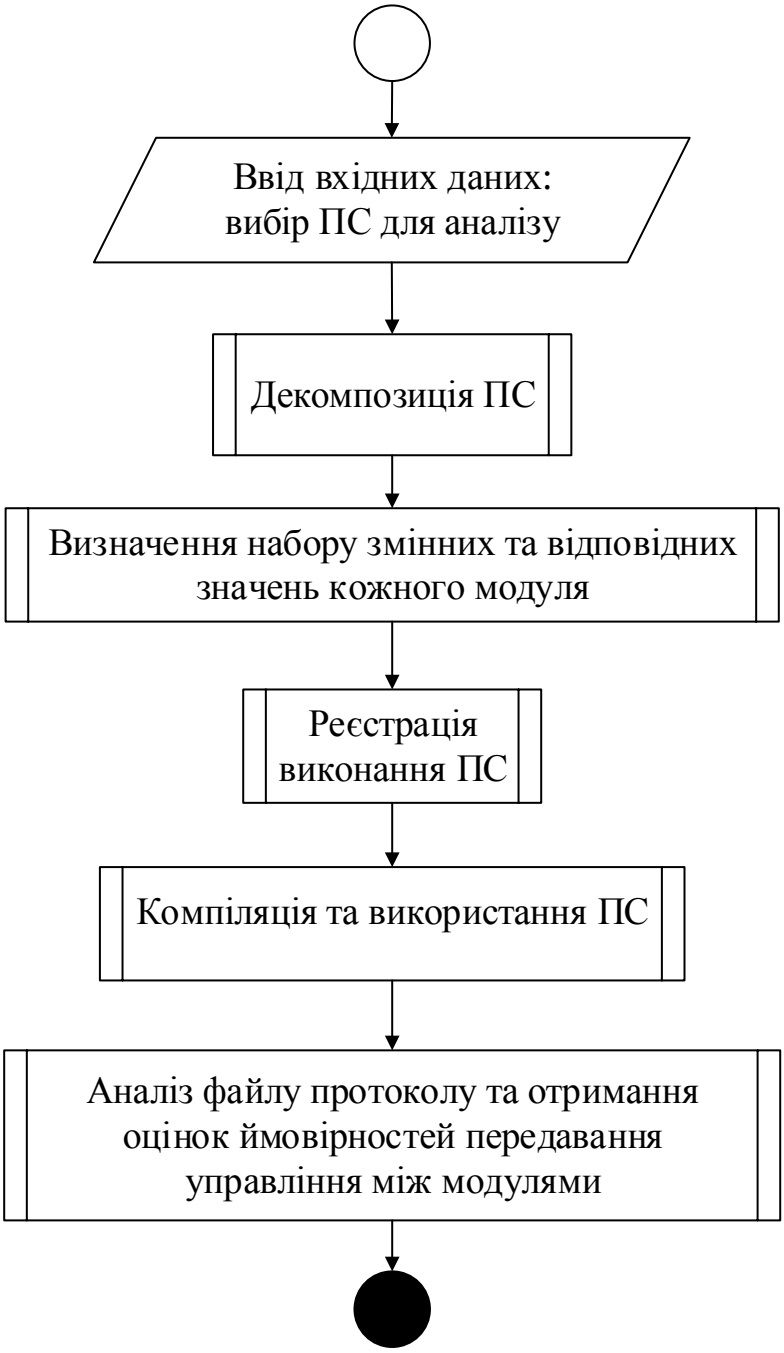


Рис. 5.27. Схема побудови моделі функціонування ПС.

Автоматизоване формування сценаріїв тестування ПС складається з наступних основних етапів:

1) на першому кроці отримують модель функціонування ПС, на основі якої здійснюється АФСТ ПС, а саме визначають систему модулів із відповідним набором змінних та їхні класи еквівалентності,

отримують оцінку ймовірності передавання управління між модулями та початковий ймовірнісний вектор;

2) генерування тестів з використанням стратегії "чорної скриньки" на основі методу, описаного у [172, 173]. Слід зазначити, що таке формування використовується тоді, коли немає інформації про внутрішню структуру ПС, а модель її функціонування створюється на основі аналізу інтерфейсу користувача;
3) генерування тестів із використанням стратегії "білої скриньки" з використанням відповідного методу [172, 173]. Слід зауважити, що таке формування потребує додаткового обчислення метрики параметрів змінних коду;
4) вивід результатів АФСТ у вигляді множини модулів, змінних та класів еквівалентності, які необхідно виконати на етапі тестування досліджуваної ПС.

Для оцінювання показників надійності ПС використано алгоритм аналізу надійності ПС з використанням ЛМВП, який складається з наступних кроків (рис. 5.28):

1) вхідною інформацією для алгоритму є параметри моделі функціонування ПС, а саме: набір модулів ПС, матриця оцінок ймовірностей передавання управління між модулями, початковий ймовірнісний вектор, середня тривалість виконання модуля. На цьому етапі використовують БД, наповнену засобом реєстрації виконання ПС;
2) у залежності від обсягу бази даних, отриманої на попередньому кроці, визначається порядок марковського ланцюга K. Якщо обсяг вибірки даних є невеликим, то для визначення порядку ЛМВП, який точніше описує поведінку ПС, застосовується критерій BIC, у іншому випадку – критерій AIC. Для цього попередньо отримують оцінки елементів матриці ймовірностей передавання управління між модулями для різного порядку ланцюга Маркова, після чого

визначають порядок процесу, якому відповідає мінімальне значення відповідного інформаційного критерію;

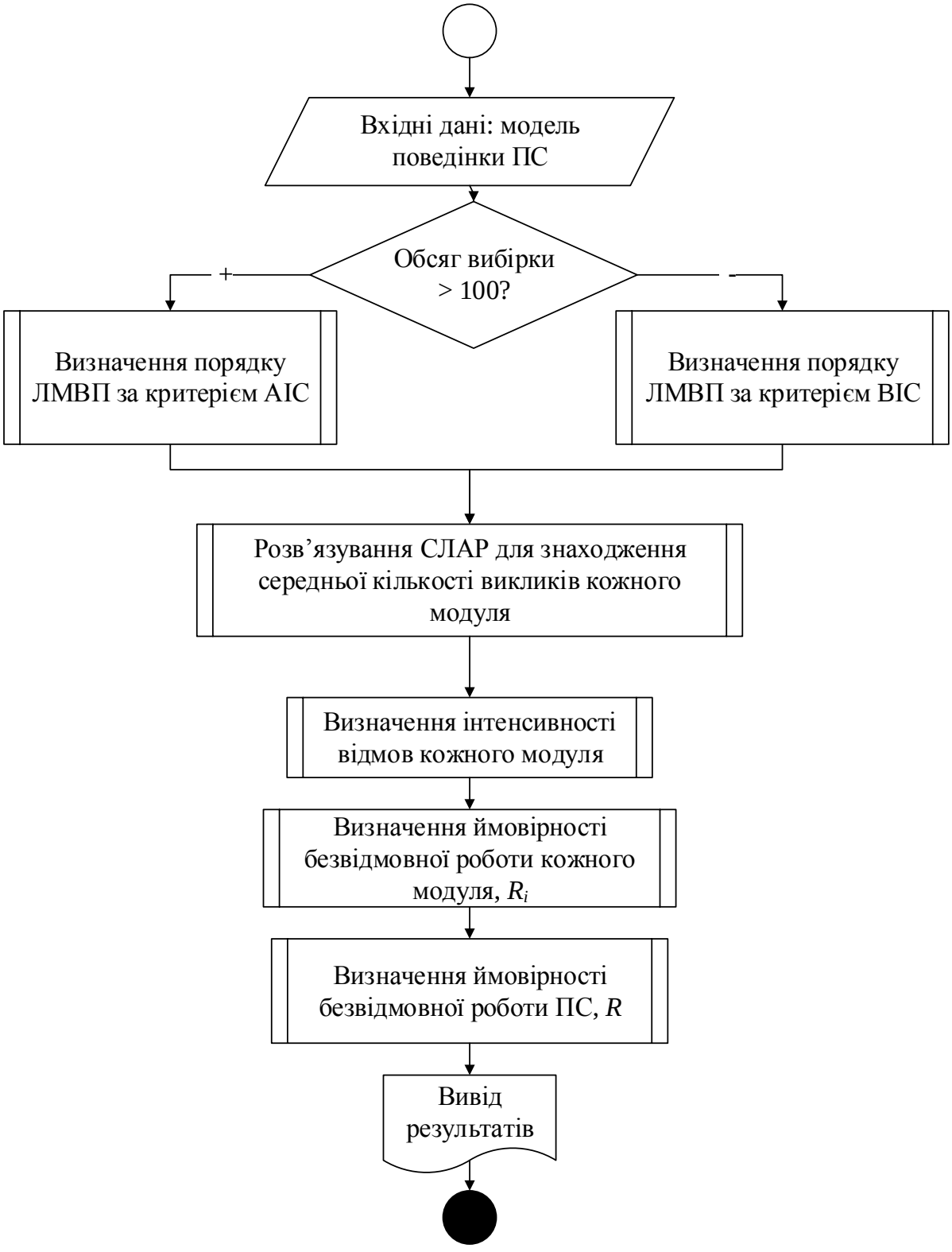


Рис. 5.28. Схема оцінювання показників надійності ПС на основі ЛМВП.

3) на наступному етапі обчислюється середня кількість очікуваного виконання кожного модуля на основі розв’язку системи лінійних алгебраїчних рівнянь (СЛАР). Для знаходження вектору кількості виконань кожного модуля в залежності від перебування в попередніх K модулях СЛАР розв’язується звичайним методом Зейделя;
4) з використанням МНПС, на основі результатів тестування модулів ПС, обчислюється інтенсивність відмов кожного модуля ПС;
5) після отримання усіх необхідних параметрів обчислюється ймовірність безвідмовної роботи кожного модуля згідно (3.2). Для числового інтегрування інтенсивності відмов використовується метод Гауса;
6) на останньому кроці алгоритму визначають ймовірність безвідмовної роботи ПС як добуток ймовірностей безвідмовної роботи її модулів.

Програмний засіб "СОН ПЗ" реалізовано мовою програмування C#, а основні функції винесено у вигляді динамічної бібліотеки DLL (dynamic link library). Таке рішення дає можливість як легкого підключення цієї бібліотеки до існуючих програмних засобів, так і розширення функціональних можливостей програмного засобу за рахунок додавання нових моделей надійності чи засобів отримання параметрів таких моделей. На рис. 5.29 наведено вигляд віконного інтерфейсу системи "СОН ПЗ".

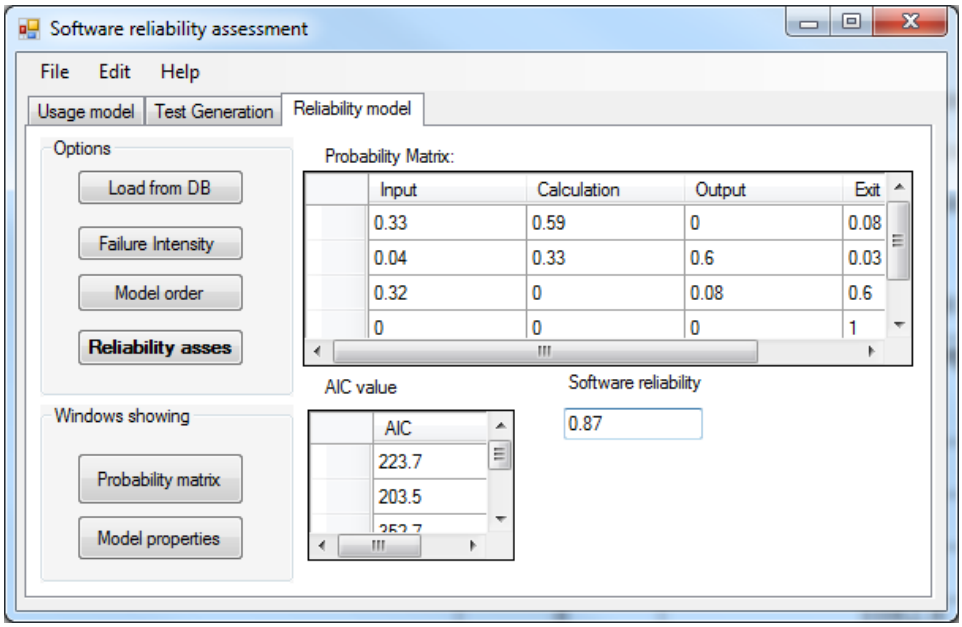


Рис. 5.29. Вигляд віконного інтерфейсу програмної системи "СОН ПЗ".

5.4. Прогнозування відмов програмних систем засобами нейронних мереж

Традиційні параметричні моделі зростання надійності ПС, такі як неоднорідний пуассонів процес успішно використовуються в практиці інженерії надійності ПЗ. Однак, на даний час не існує єдиної параметричної моделі, яка дозволяє точно прогнозувати надійність для усіх класів ПС. Разом з тим складність сучасних програмних систем, та складна взаємодія з середовищем функціонування значно утруднюють побудову аналітичних моделей задовільного ступеня адекватності.

До перспективних методів прогнозування надійності ПС, зокрема кількості відмов (чи інтенсивності відмов) можуть бути методи на основі непараметричних моделей [266, 267]. Такі моделі позбавлені основних недоліків та труднощів аналітичних моделей, оскільки не висувають жодних припущень про механізм відмов ПС. Основою таких методів є опис стохастичного процесу відмов ПС на основі достовірної та репрезентативної статистичної вибірки. Переважна більшість непараметричних моделей зводиться до прогнозування кількості відмов на деякому часовому інтервалі, що є задачею прогнозування часових рядів. До середини 80-х років минулого століття існувало декілька

загальновизнаних методів прогнозування часових рядів: економетричні, регресійні, методи Бокса–Дженкінса (ARIMA, ARMA) [268]. Однак в останні два десятиліття все більшого поширення набувають методи прогнозування часових рядів (в т.ч. надійності ПС) на основі нейронних мереж (НМ) завдяки їх доведеній якості узагальнення та апроксимації практично будь-яких гладких функцій [269]. Так, НМ знайшли широке застосування для передбачення цін на фондовій біржі, прогноз погоди, прогноз споживання електроенергії тощо [269, 270]. До основної переваги таких методів прогнозу можна віднести відсутність апріорних припущень щодо закономірностей стохастичного процесу, в той час як основним недоліком є необхідність отримання статистичної вибірки достатнього обсягу та необхідність навчання НМ для кожної нової вибірки.

Штучні НМ, завдяки їх здатності до самонавчання, можуть бути хорошим апроксиматором експериментальних наборів даних з невідомою функціональною залежністю [271]. Натренована на обмеженій множині даних мережа здатна узагальнювати отриману інформацію і показувати хороші результати на даних, що не використовувалися при її навчанні. При виборі архітектури НМ зазвичай випробовується кілька конфігурацій з різною кількістю елементів. Виходячи з того, що задача прогнозування є випадком завдання регресії, випливає, що вона може бути вирішена наступними типами нейронних мереж [272]: багатошаровим перцептроном Румельхарта (Multilayer Perceptron – MLP), радіально-базисною мережею (Radial Basis Function – RBF), мережею Хопфілда, узагальнено-регресійною мережею (Generalized Regression Neural Network – GRNN), мережею Вольтерра та мережею Елмана.

Для задач прогнозування надійності ПС досліджувались різні нейромережеві моделі [266, 267, 273–277]. Зокрема нейронні мережі Wavelet демонструють ефективне прогнозування на початковій стадії тестування ПС [276]. Рекурентні НМ, які однак потребують тривалого навчання, також показують перспективність використання у задачах прогнозуваннях надійності ПС [277].

Більшість нейронних мереж, що використовуються для моделювання надійності ПС, можуть бути поділені на два класи [273]. Перший – використовує в якості вхідних даних кумулятивний час виконання і відповідну кумулятивну кількість відмов як бажаний вихід. Цей клас використовує для моделювання різні архітектури нейронних мереж, такі як рекурентні НМ, мережі Елмана тощо. Другий клас моделює надійність ПС на основі НМ типу "множинний вхід із затримкою – одиночний вихід". Так, наприклад, в роботі [274] було використано значення останніх 50 часів між відмовами в якості входів із затримкою для прогнозування часу виявлення наступної помилки.

Дослідження використання НМ для прогнозування надійності ПС розвиваються в різних напрямках. Так, було показано, що використання комбінованих НМ дозволяє суттєво покращити прогнозування надійності ПС [267]. Інші дослідження використовують підхід на основі НМ для побудови динамічно зваженої комбінаційної моделі надійності ПЗ, яка дозволяє суттєво покращити результати прогнозування надійності ПЗ [273]. Ще один підхід заснований на використанні самогенерованих НМ з ітераційним навчанням (Iteration learning selfgenerating neural networks – ISGNN). Використання такого типу НМ дозволяє зменшити тривалість навчання мережі до 1/6 від тривалості навчання мережі зворотного поширення похибки, а середня абсолютна похибка, як і середня квадратична похибка такої мережі зменшуються на 3–15% порівняно з мережею зворотного поширення похибки [278].

Разом з тим, одним з нових класів нейронних мереж є мережі на основі радіально-базисних функцій (RBF), які володіють великою швидкістю навчання та успішно використовуються для задач апроксимації невідомих функцій [279, 280]. Мережа RBF не містить рекурсії і характеризується наступними особливостями: має єдиний прихований шар нейронів; тільки нейрони прихованого шару мають нелінійну активаційну функцію; синаптичні ваги всіх нейронів прихованого шару дорівнюють одиниці [281].

Таким чином шляхом оптимального вибору архітектури, структури та параметрів НМ можна досягнути заданої точності апроксимації функції

інтенсивності відмов чи кумулятивної кількості відмов і, таким чином, прогнозування показників надійності ПС (кількість відмов чи інтенсивність відмов).

5.4.1. Програмний засіб прогнозування відмов програмних систем на основі нейронних мереж

В цьому підрозділі описано програмний засіб для прогнозування відмов ПС, представлених у вигляді часових рядів, отриманих з результатів тестування чи повідомлень про відмови на етапі експлуатації ПС, з використанням НМ RBF та Елмана.

Основними функціональними можливостями розробленого програмного засобу є:

- Вибір НМ. Програма підтримує підключення різних архітектур НМ.
- Навчання мережі. Передбачає вибір файлу з даними, з яких НМ буде навчатись. Форматом даних для навчання є формат CSV.
- Завантаження конфігураційного файлу, який буде містити налаштування для мережі.
- Задання параметрів мережі – кількість вхідних нейронів, кількість нейронів прихованого шару, кількість епох навчання, кількість даних для навчання, кількість даних для перевірки ефективності прогнозування.
- Завантаження стану мережі. Навчання мережі виконується для визначення вагових коефіцієнтів, які є станом мережі.
- Збереження конфігураційного файлу.
- Збереження стану мережі (вагових коефіцієнтів, отриманих в результаті навчання НМ).
- Перегляд результатів прогнозування – кількості відмов на наступному часовому інтервалі.
- Перегляд графіку результатів прогнозування відмов.

Для реалізації засобу, було вибрано високорівневу мову програмування C# та .NET Framework 3.5. Програмна реалізація модулів НМ RBF та Елмана була створена за допомогою бібліотеки Encog [265].

Encog – це бібліотека НМ та засобів машинного навчання, яка містить класи для створення найрізноманітніших мереж, а також підтримку класів для нормалізації і обробки даних для цих НМ. Бібліотека Encog доступна для мов програмування Java, .Net і Silverlight. На рис. 5.30 зображено базову діаграму класів бібліотеки Encog.

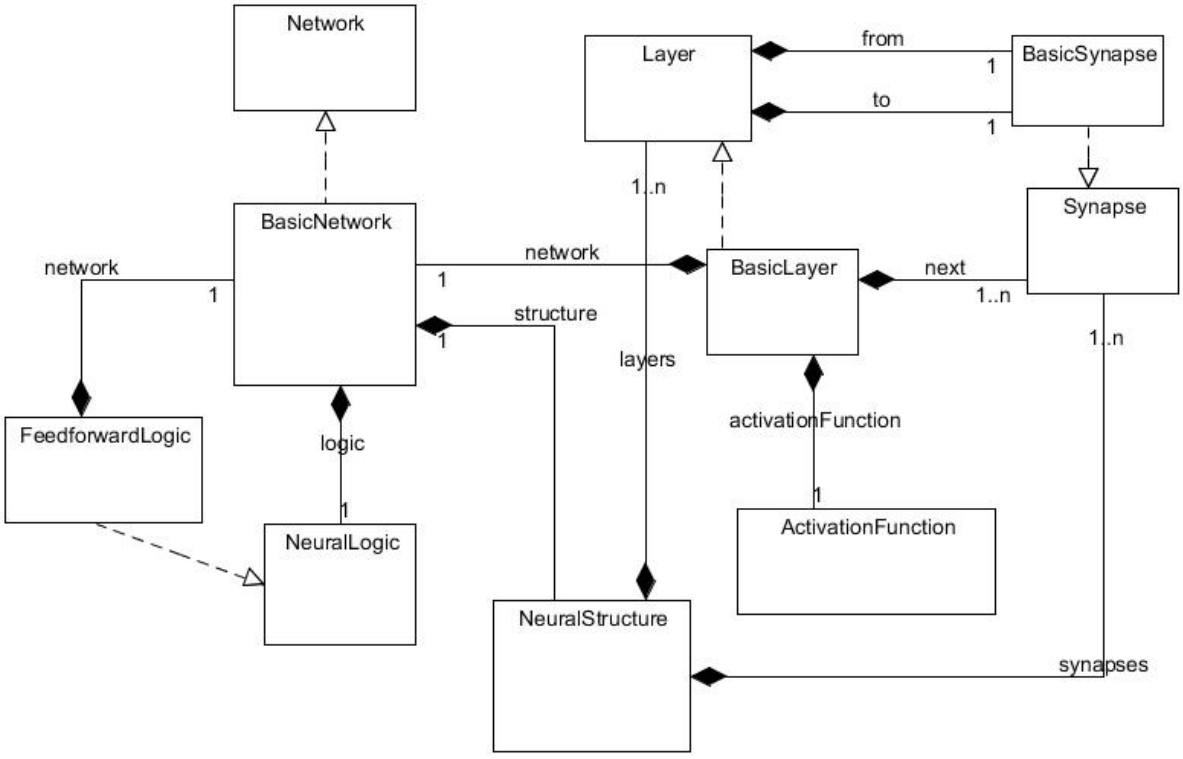


Рис. 5.30. Базова діаграма класів бібліотеки Encog.

Загальна діаграма компонентів програмного засобу для прогнозування відмов ПС на основі НМ зображена на рис. 5.31.

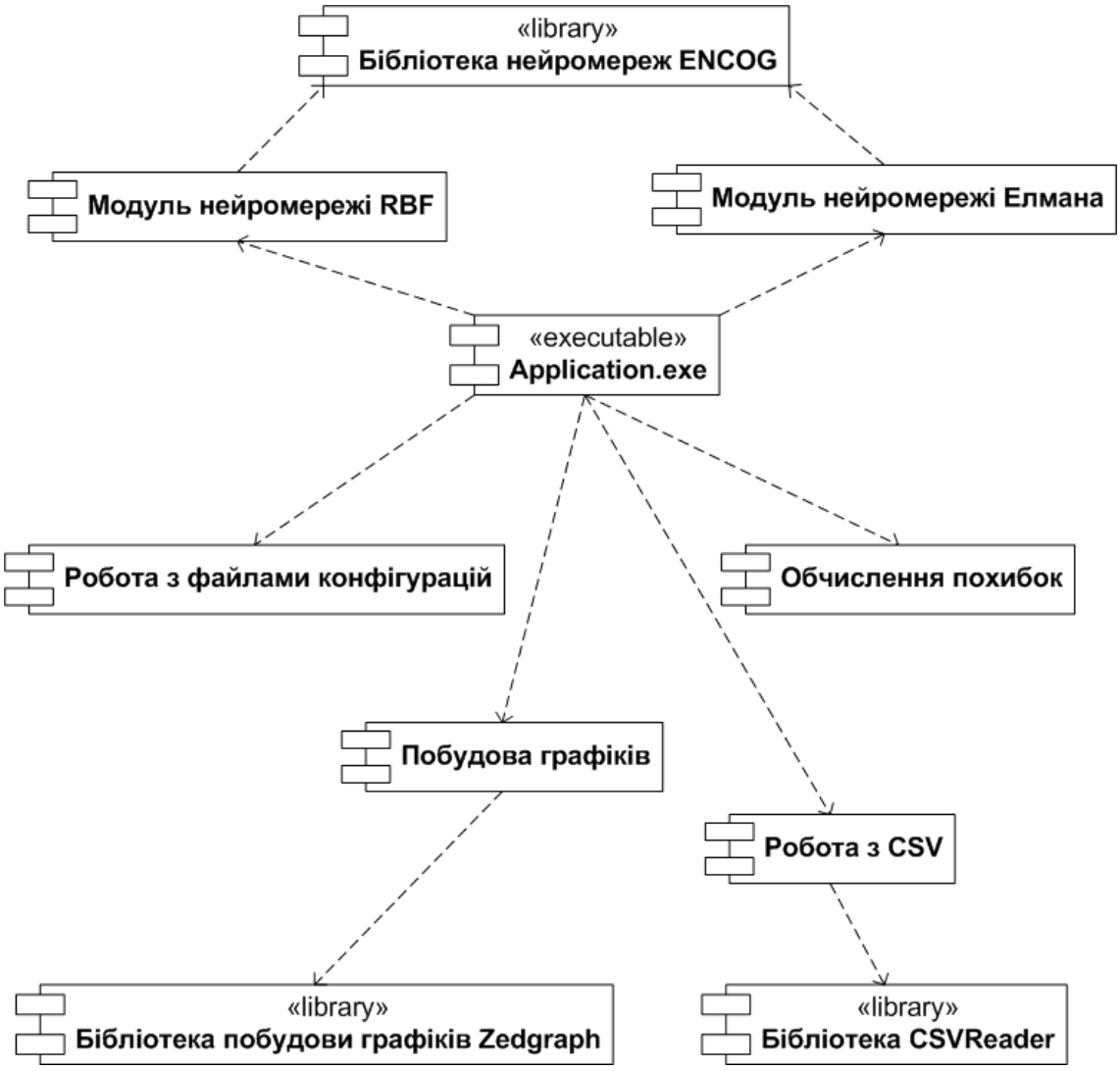


Рис. 5.31. Діаграма компонентів програмного засобу прогнозування відмов ПС.

Для зручності відображення результатів прогнозування у користувача є можливість переглянути результати на графіку. Графік прогнозування будувався за допомогою бібліотеки побудови графіків ZedGraph, яка призначена для створення 2D ліній, лінійних, стовпчастих, кругових діаграм з довільних наборів даних. Класи бібліотеки включають в себе код для вибору відповідних діапазонів масштабу і розмірів кроків на основі діапазону значень даних.

Дані для навчання НМ надаються програмі у вигляді CSV файлів. Тому для роботи з даним форматом було використано відповідний інструментарій, а саме бібліотеку CSVReader. CSVReader це .Net бібліотека класів для розбору даних з практично будь-яким роздільником формату, який визначається бібліотекою

самостійно. Її можна використовувати з C #, VB.Net, ASP.Net, або будь-якою іншою .Net мовою.

Для зручності користувача усі налаштування параметрів НМ можна зберегти у файл для майбутнього відновлення. У файл конфігурації зберігаються наступні дані:

- архітектура НМ;
- функція активації;
- кількість вхідних нейронів;
- кількість нейронів прихованого шару;
- похибка навчання;
- кількість ітерацій для навчання;
- кількість інтервалів для перевірки правильності прогнозу;
- кількість інтервалів прогнозу.

Також після навчання НМ її стан також можна зберегти у файл. Стан мережі – це синаптичні ваги нейронів, які формуються в процесі навчання. Файли конфігурації зберігаються у двійковому форматі.

Для того, щоб програмний засіб підтримував підключення різних архітектур НМ, потрібно реалізувати наступну поведінку:

- програма повинна забезпечувати підтримку різних варіантів алгоритму;
- потрібно змінювати поведінку кожного екземпляра класу;
- необхідно змінювати поведінку об'єктів на стадії виконання;
- введення інтерфейсу дозволяє класам-клієнтам нічого не знати про класи, що реалізують цей інтерфейс та інкапсулюють в собі конкретні алгоритми.

Для вирішення цієї задачі використано шаблон проектування "Стратегія" (рис. 5.32).

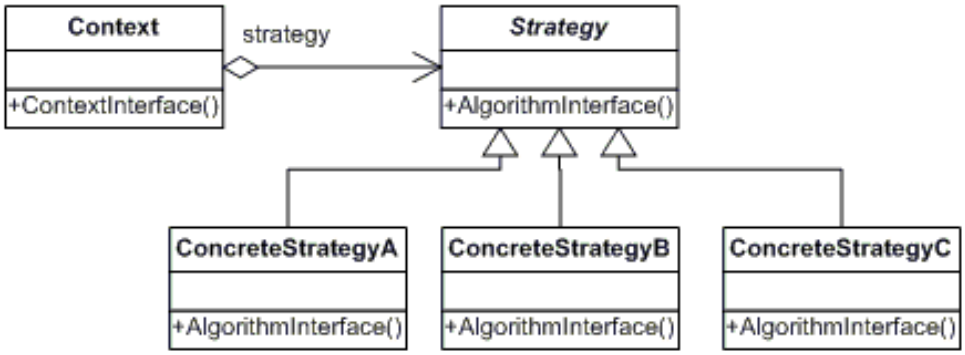


Рис. 5.32. Шаблон проектування "Стратегія".

При реалізації шаблону "Стратегія" використовуються такі класи:

- Клас Strategy визначає, як будуть використовуватися різні алгоритми.
- Конкретні класи ConcreteStrategy реалізують ці різні алгоритми.
- Клас Context використовує конкретні класи ConcreteStrategy допомогою посилання на конкретний тип абстрактного класу Strategy. Класи Strategy і Context взаємодіють з метою реалізації обраного алгоритму (в деяких випадках класу Strategy потрібно посилати запити класу Context). Клас Context пересилає класу Strategy запит, що надійшов від його класу-клієнта.

Також реалізація даного шаблону проектування має наступні переваги:

- Шаблон Strategy визначає множину архітектур НМ.
- Це дозволяє відмовитися від використання перемикачів та / або умовних операторів.
- Виклик всіх алгоритмів здійснюється стандартним чином (всі вони мають однаковий інтерфейс).

Оскільки навчання НМ відбувається не в головному потоці програми потрібно реалізувати поведінку програми, яка повідомить головний потік про завершення навчання і готовність до прогнозування. Для вирішення цієї задачі було використано шаблон проектування "Спостерігач" (рис. 5.33).

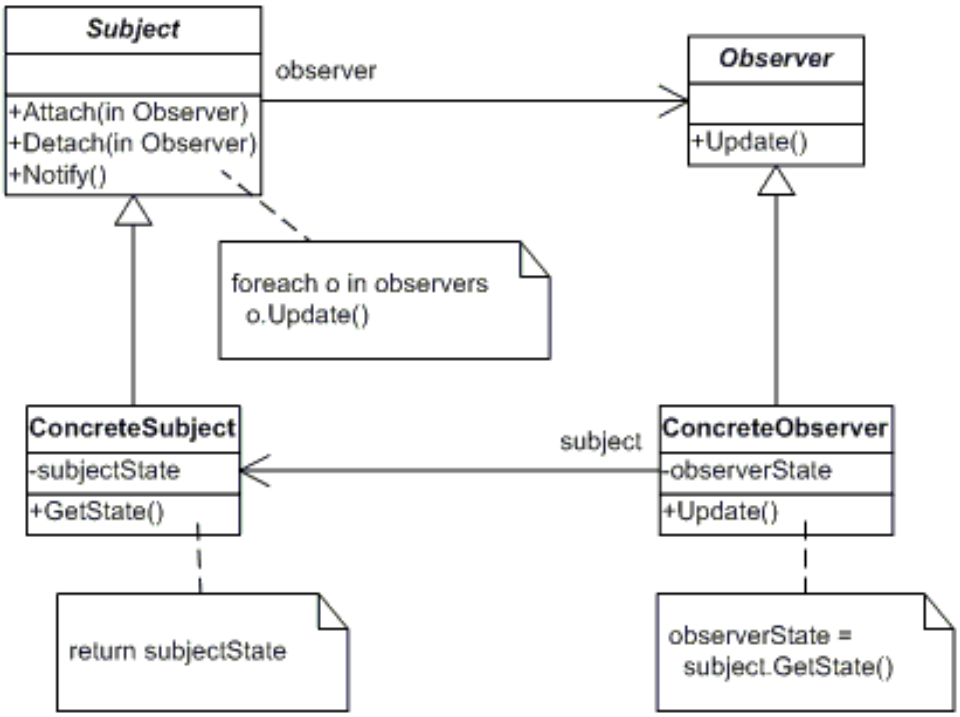


Рис. 5.33. Шаблон проектування "Спостерігач".

При реалізації шаблону "Спостерігач" використано такі класи:

- Observable – інтерфейс, що визначає методи для додавання, видалення та оповіщення спостерігачів.
- Observer – інтерфейс, за допомогою якого спостережуваний об'єкт оповіщає спостерігачів.
- ConcreteObservable – конкретний клас, який реалізує інтерфейс Observable.
- ConcreteObserver – конкретний клас, який реалізує інтерфейс Observer.

При зміні спостережуваного об’єкту (навчання завершилось), сповіщення спостерігачів може бути реалізоване за такими сценаріями:

- Спостережуваний об’єкт надсилає, кожному із зареєстрованих спостерігачів, всю потенційно релевантну інформацію (примусове розповсюдження).
- Спостережуваний об’єкт надсилає, кожному із зареєстрованих спостерігачів, лише повідомлення про те що інформація була змінена, а кожен із спостерігачів, за необхідності, самостійно

здійснює запит необхідної інформації у спостережуваного об’єкта (розповсюдження за запитом).

Інтерфейс користувача був спроектований таким чином щоб програмний засіб для прогнозування відмов ПС міг підтримувати різні модулі реалізацій НМ. На рис. 5.34 зображено інтерфейс користувача програмного засобу.

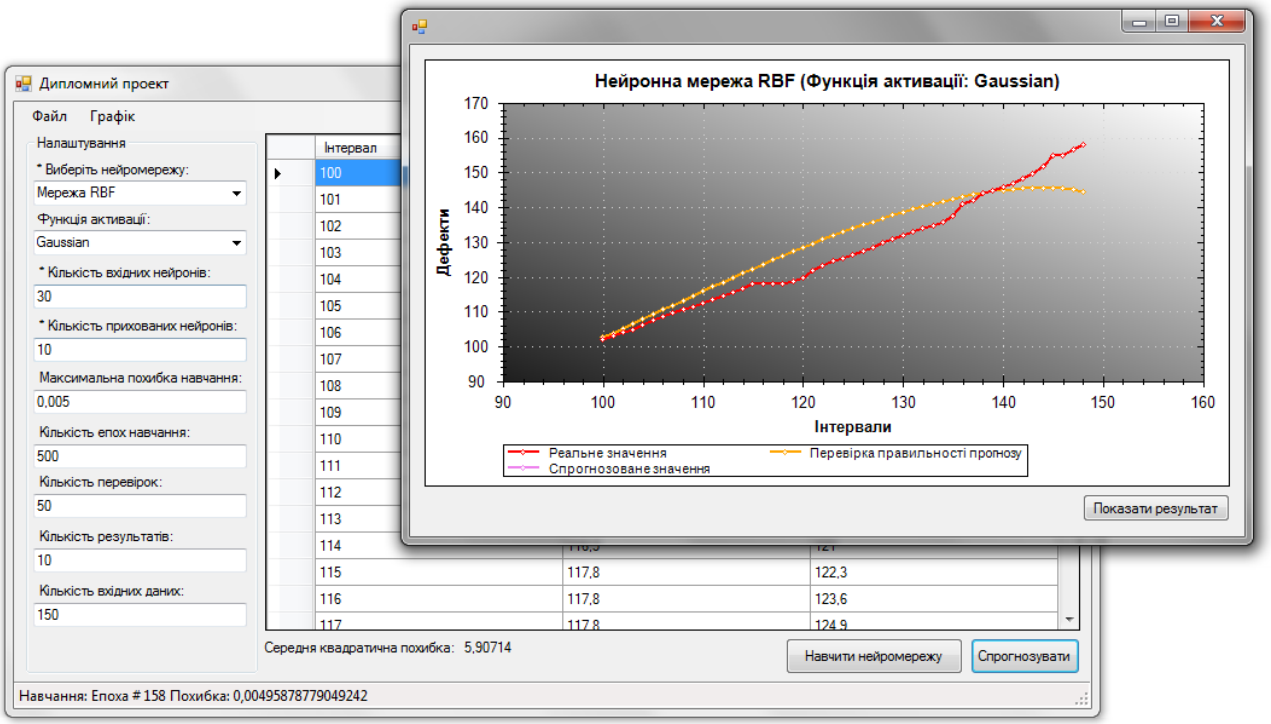


Рис. 5.34. Інтерфейс програмного засобу прогнозування відмов ПС.

5.4.2. Вплив параметрів та архітектури нейронних мереж на ефективність прогнозування відмов програмних систем

Опис проведених експериментів

Вхідними даними для навчання НМ та прогнозування відмов були часові ряди, отримані на основі результатів тестування двох програмних систем з відкритим вихідним кодом: веб-браузера Chromium [282] та операційної системи Chromium-OS [283]. Використання результатів тестування двох промислових ПС покликане підтвердити достовірність результатів і висновків щодо застосовності дослідженої конфігурації нейронної мережі для прогнозування відмов ПС.

Кожна точка часового ряду являла собою значення кількості відмов на часовому інтервалі, на які розбивався весь діапазон тривалості тестування. Такі інтервали вибирались рівномірними, в той час як результати тестування в початковому вигляді представлені без будь-якого нормування. Тому вхідні дані перед навчанням проходили нормалізацію. Оскільки у тестуванні брала участь різна кількість тестувальників, початкові дані переводяться в людино-дні. Проведені попередні експерименти показали, що часовий ряд у вигляді залежності "кількість відмов"–"часовий інтервал" незадовільно прогнозується як НМ Елмана [257], так і RBF [256]. Тому для подальших досліджень було використано часовий ряд у вигляді кумулятивної кількості відмов, виявлених протягом усіх інтервалів часу до поточного. Використання такого представлення результатів тестування ПС дозволило суттєво покращити точність прогнозування відмов ПС за допомогою НМ (як рекурентних, так і RBF) [258, 259]. Після такої попередньої обробки отриманий часовий ряд використовували для навчання НМ. Усі результати тестування в даній роботі були розбиті на 150 однакових часових інтервалів. Перших 100 інтервалів було використано для навчання НМ, а останніх 50 – для перевірки точності прогнозу (прогнозовані НМ результати порівнювались з цими контрольними значеннями).

Для оцінки ефективності прогнозування використовували наступні параметри: кількість епох навчання, яка характеризувала швидкість навчання НМ; коефіцієнт детермінації (R^2), який розглядають як універсальну міру залежності однієї випадкової величини від множини інших, та середня квадратична похибка апроксимації, яка характеризує апроксимацію моделі до статистичних даних в рівномірній метриці та показує близькість прогнозних та експериментальних даних.

Прогнозування відмов ПС за допомогою НМ Елмана

Для дослідження впливу функції активації НМ Елмана на ефективність прогнозування відмов ПС було проведено дві серії експериментів [257, 259], в яких використовувались наступні функції активації: лінійна, гіперболічний тангенс та синус. В першій серії конструювалась НМ з 10 нейронами вхідного

шару та 30 нейронами прихованого шару. В другій серії експериментів НМ містила 30 нейронів у вхідному шарі та 10 – у прихованому. Навчання НМ здійснювалось до досягнення похибки 0,005 або ж до 5000 епох навчання, залежно, що наставало раніше. Навчання та прогнозування відмов здійснювалось на основі результатів тестування веб-браузера з відкритим вихідним кодом Chromium [282], було використано загальнодоступні звіти за близько 870 днів тестування, під час якого було виявлено біля 1000 помилок. Результати статистичного опису ефективності прогнозування відмов ПС за допомогою НМ Елмана під час обох серій експериментів наведено в табл. 5.1.

Таблиця 5.1.

Ефективність прогнозування відмов ПС нейронною мережею Елмана

Функція активації	**Кількість епох навчання**	**Коефіцієнт детермінації**	**Середня квадратична похибка апроксимації**
I експеримент			
Лінійна	631	0,979	922,4
Tanh	6000	0,989	6,6
Sin	6000	0,968	10,1
II експеримент			
Лінійна	3797	0,932	94,9
Tanh	92	0,876	279,5
Sin	2165	0,976	167,8

Як видно з табл. 5.1, у першому експерименті оптимальною функцією активації НМ є гіперболічний тангенс (слід зазначити, що хоча навчання тривало 6000 епох без досягнення заданої точності, похибка навчання становила 0,0053 – що лише трохи більше заданої межі припинення навчання – після чого наставав параліч навчання мережі). Натомість у другому експерименті (з іншою кількістю нейронів у вхідному та прихованому шарах) ця функція активації є найгіршою серед досліджених функцій, незважаючи на швидке навчання (92 епохи). Аналіз

табл. 5.1 показує, що у другому випадку оптимальною є лінійна функція активації – доволі значна тривалість навчання при найкращому значенні середньої квадратичної похибки апроксимації та дещо гіршому коефіцієнті детермінації. Слід також зауважити, що використання лінійної функції активації може бути доцільним для обох проведених експериментів – в обох випадках результати прогнозування є задовільними.

Графік залежності кількості прогнозованих та реальних помилок веб браузера Chromium на інтервалах 100–150 у випадку використання НМ з функцією активації "гіперболічний тангенс" для першого та другого експерименту наведено на рис. 5.35 та 5.36 відповідно. Рис. 5.35 демонструє близькість прогнозованих та реальних точок та загальну якість прогнозу і ефективність використання RBF нейронних мереж для цієї задачі. Натомість рис. 5.36 наочно ілюструє висновок, зроблений з табл. 5.1 про незадовільну точність прогнозування відмов ПЗ НМ Елмана з цією функцією активації у випадку 30 нейронів вхідного шару та 10 – прихованого.

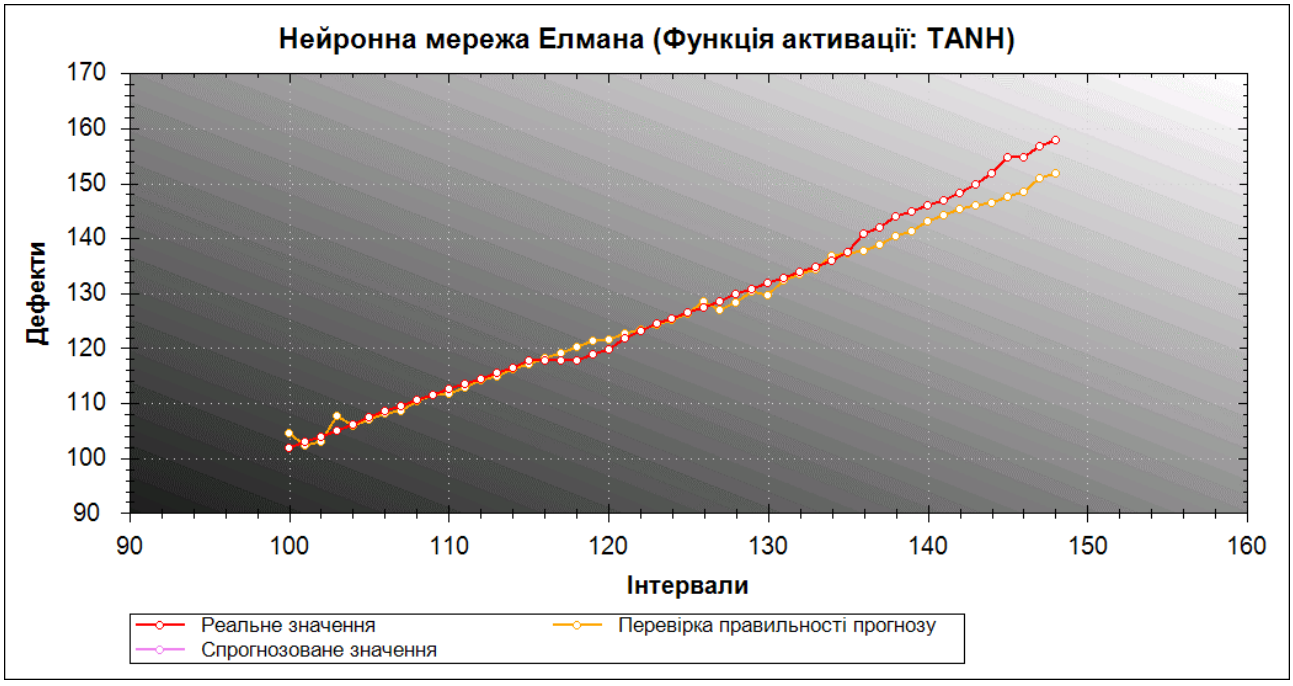


Рис. 5.35. Результати прогнозування відмов браузера Chromium НМ Елмана з функцією активації Tanh (30 нейронів вхідного шару, 30 – прихованого).

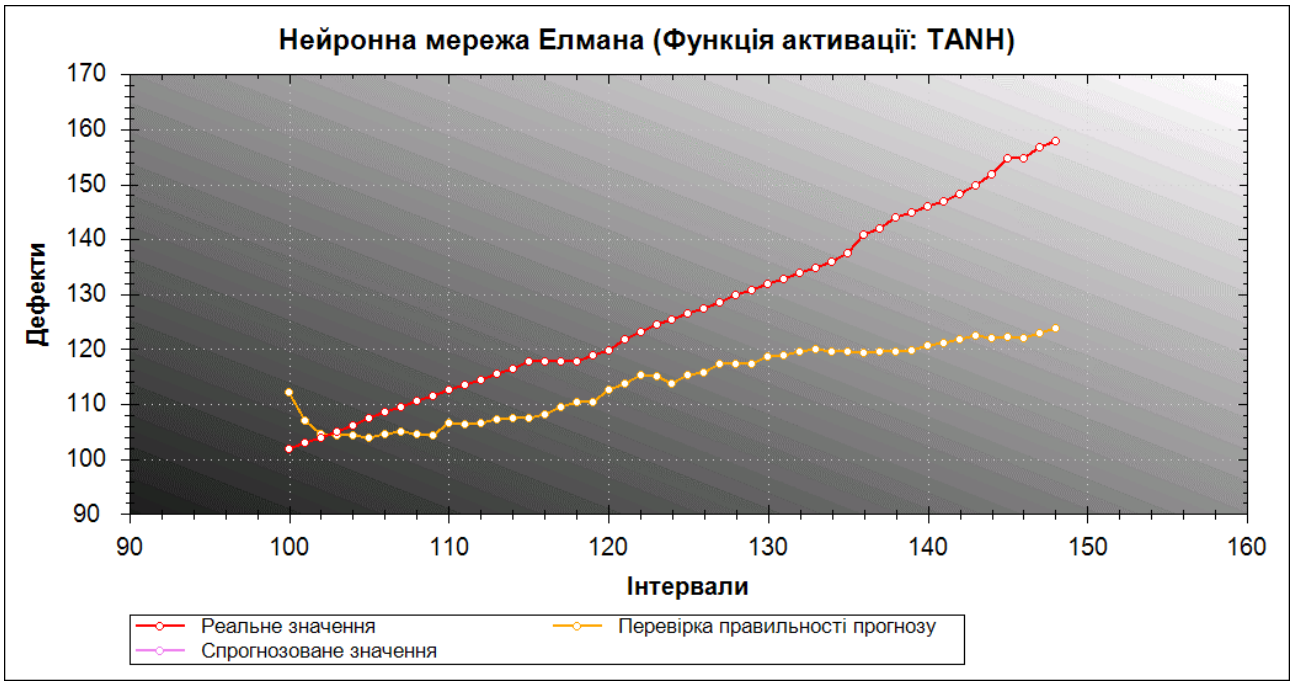


Рис. 5.36. Результати прогнозування відмов браузера Chromium НМ Елмана з функцією активації Tanh (30 нейронів вхідного шару, 10 – прихованого).

Таким чином, рекурентна НМ Елмана при належному підборі параметрів її архітектури (функція активації "гіперболічний тангенс" або лінійна, кількість нейронів прихованого шару не менша, ніж кількість нейронів вхідного) здатна прогнозувати відмови програмних систем значного ступеня складності (що показано на прикладі браузера Chromium) з достатньою точністю (середня квадратична похибка апроксимації не більше 10 при значеннях функції 100–160 відмов).

Прогнозування відмов ПС за допомогою НМ RBF

Вплив функції активації НМ на ефективність прогнозування відмов ПС.

Дослідження впливу функції активації НМ RBF на ефективність прогнозування відмов ПС, представлених у вигляді часового ряду, проводилось у [258, 263]. Для цього дослідження було обрано 4 найбільш поширених функції активації НМ RBF: Gaussian, Multiquadric, Inverse Multiquadric та Mexican Hat [269].

Було проведено дві серії експериментів, кожна з яких включала по одному експерименту з кожною функцією активації. В першій серії конструювалась

нейронна мережа з 10 нейронами вхідного шару та 30 нейронами прихованого шару. В другій серії експериментів нейронна мережа містила 30 нейронів у вхідному шарі та 10 – у прихованому. Навчання нейронної мережі здійснювалось до досягнення похибки 0,005 або ж до 5000 епох навчання, залежно, що наставало раніше. Результати статистичного опису ефективності прогнозування відмов ПС за допомогою НМ RBF під час обох серій експериментів наведено в табл. 5.2.

Таблиця 5.2.

Статистичні характеристики якості прогнозування відмов браузера Chromium НМ типу RBF з різними функціями активації.

Функція активації	**Кількість епох навчання**	**Коефіцієнт детермінації**	**Середня квадратична похибка апроксимації**
I експеримент			
Gaussian	105	0,997	15,4
Multiquadric	432	0,995	22,3
Inverse Multiquadric	293	0,997	14,4
Mexican Hat	130	0,980	146,1
II експеримент			
Gaussian	186	0,986	113,9
Multiquadric	4185	0,988	174,0
Inverse Multiquadric	4149	0,994	63,2
Mexican Hat	3060	0,946	360,9

Як видно з табл. 5.2 найбільш придатною для прогнозування є функція активації Inverse Multiquadric, яка в обох конфігураціях показала найкращі статистичні показники якості прогнозу в поєднанні з не дуже великим часом навчання. Функція активації Gaussian може бути хорошою альтернативою для експрес аналізу надійності, оскільки час її навчання є найменшим, а точність

прогнозу майже не поступається найкращому випадку (за умови оптимально підібраних інших параметрів архітектури мережі). Натомість функція активації Mexican Hat виявилась непридатною для задачі прогнозування надійності ПС у вигляді кумулятивного часового ряду в обох конфігураціях НМ, і продемонструвала найбільшу схильність до "паралічу" при навчанні. Слід також зауважити, що функція активації Gaussian виявила найбільшу стійкість до "паралічу" навчання, який часто виникав у другій серії експериментів, а у випадку функції активації Multiquadric також і в першій серії. Тому функцію Multiquadric, незважаючи на задовільні параметри прогнозу, не слід використовувати для даної задачі через схильність такої НМ до "паралічу" при навчанні.

Графіки залежності кількості прогнозованих та реальних відмов веб-браузера Chromium на інтервалах 100–150 наведено на рис. 5.37 і 5.38 для першого експерименту з використанням функцій активації Inverse Multiquadric та Mexican Hat відповідно. Рис. 5.37 демонструє близькість прогнозованих та реальних точок та загальну якість прогнозу і ефективність використання НМ типу RBF для задачі прогнозування відмов ПС у вигляді часового ряду за умови оптимального вибору параметрів НМ.

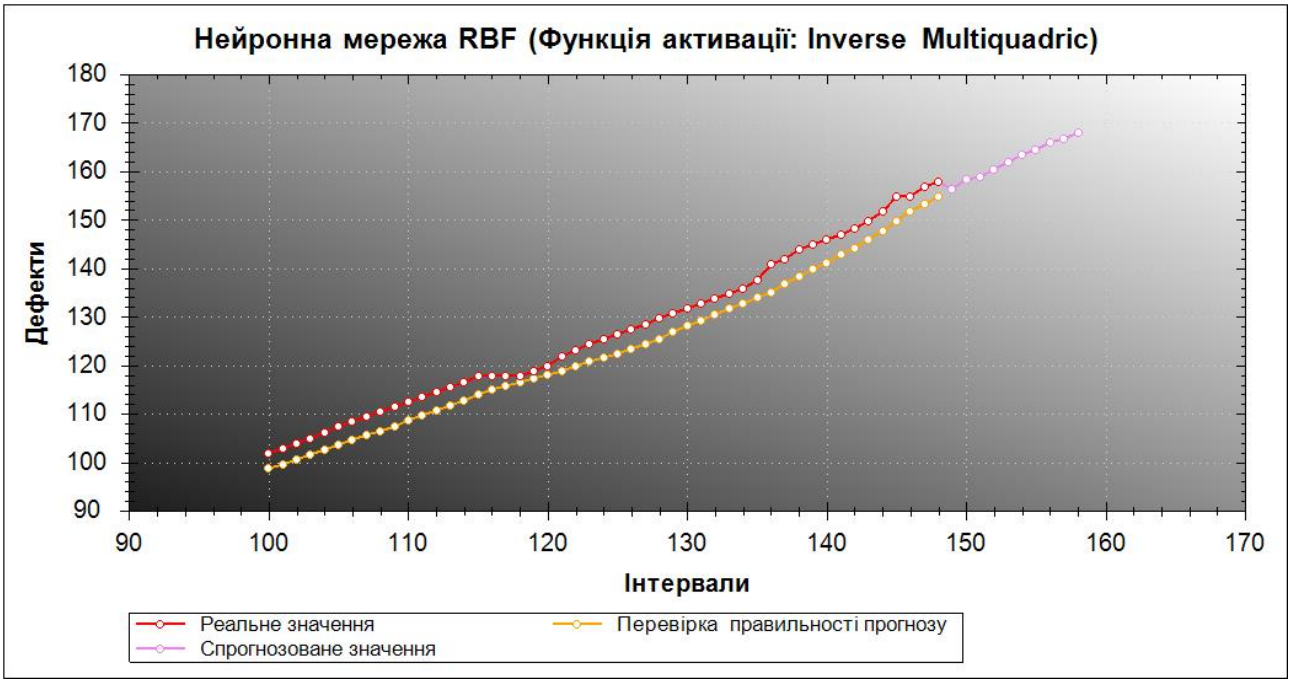


Рис. 5.37. Результати прогнозування відмов браузера Chromium HM RBF (функція активації Inverse Multiquadric, 10 нейронів вхідного та 30 нейронів прихованого шару).

Натомість з рис. 5.38 можна отримати наочне підтвердження даних табл. 5/2 про непридатність функції активації Mexican Hat для прогнозування надійності ПС – починаючи з мінімального відхилення на сотому часовому інтервалі (який ще входив в навчальну вибірку), відхилення зростає і сягає більш ніж 30% на 150-му часовому інтервалі.

Таким чином, можна зробити висновок, що оптимальною з точки зору швидкості навчання HM RBF є функція активації Gaussian, тоді як з точки зору точності прогнозування такою функцією є Inverse Multiquadric. Функція активації Mexican Hat є непридатною для такої постановки задачі прогнозування відмов ПС.

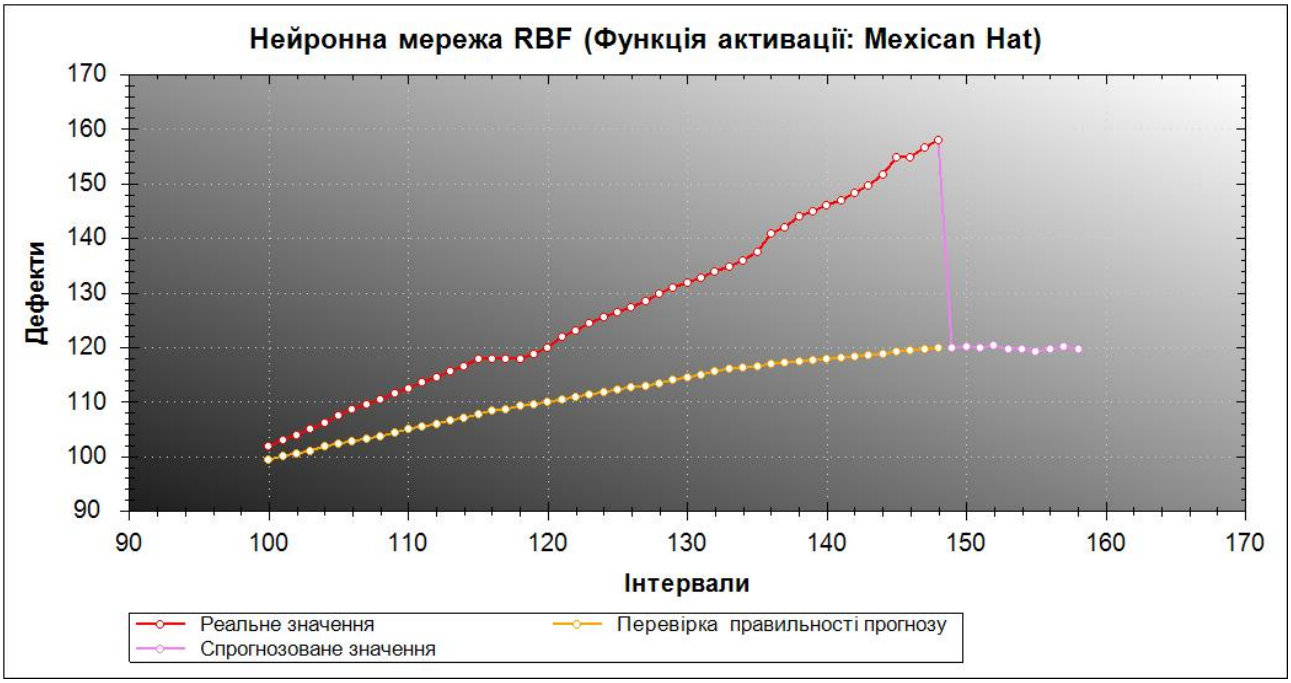


Рис. 5.38. Результати прогнозування відмов браузера Chromium НМ RBF (функція активації Mexican Hat, 10 нейронів вхідного та 30 прихованого шару).

Вплив кількості нейронів вхідного шару НМ на ефективність прогнозування відмов ПС.

В роботі [264] досліджено вплив кількості нейронів вхідного шару НМ RBF з функціями активації Gaussian та Inverse Multiquadric на ефективність прогнозування відмов ПС для встановлення конфігурації НМ при побудові засобів прогнозування відмов ПС.

Було проведено дві серії експериментів для кожної з функцій активації, в яких використовувалась різна кількість нейронів вхідного шару – від 5 до 45 з кроком 5. В першій серії конструювалась НМ з 30 нейронами прихованого шару, в другій – з 10. Навчання НМ здійснювалось до досягнення похибки 0,005 або ж до 5000 епох навчання, залежно, що наставало раніше.

Типові результати статистичного опису ефективності прогнозування відмов ПС наведено в таблиці 5.3 для браузера Chromium та в таблиці 5.4 для системи Chromium-OS (в цих таблицях стовпці, позначені літерою G відповідають функції активації Gaussian, а позначені літерами IM – функції активації Inverse Multiquadric). З цих таблиць можна побачити, що хоча

абсолютні значення тривалості навчання, коефіцієнта детермінації та середньої квадратичної похибки апроксимації є різними, однак тенденції їх зміни в залежності від конфігурації НМ є дуже схожими.

Таблиця 5.3.

Залежність ефективності прогнозу відмов браузера Chromium від кількості вхідних нейронів НМ RBF

Кількість вхідних нейронів	**Кількість прихованих нейронів**	**Тривалість навчання, епохи**		**Коефіцієнт детермінації**		**Середня квадратична похибка, %**	
		G	IM	G	IM	G	IM
10	30	146	478	0,996	0,998	4,9	1,3
15	30	14	171	0,997	0,998	8,0	2,4
45	30	443	27	0,997	0,994	64,7	29,0
5	10	69	1976	0,996	0,998	3,2	3,6
15	10	78	3916	0,997	0,998	2,2	12,4
25	10	68	6991	0,507	0,998	12,1	8,5

Таблиця 5.4.

Залежність ефективності прогнозу відмов Chromium-OS від кількості вхідних нейронів НМ RBF

Кількість вхідних нейронів	**Кількість прихованих нейронів**	**Тривалість навчання, епохи**		**Коефіцієнт детермінації**		**Середня квадратична похибка, %**	
		G	IM	G	IM	G	IM
10	30	212	324	0,882	0,983	5,1	3,2
15	30	59	279	0,969	0,970	4,8	3,3
45	30	113	26	0,988	0,971	24,5	22,4
5	10	364	1618	0,958	0,991	3,2	2,4
15	10	37	1536	0,987	0,979	5,4	3,0
25	10	62	260	0,982	0,967	7,4	3,4

Залежність середньої квадратичної похибки апроксимації від кількості нейронів вхідного шару у випадку прогнозування відмов браузера Chromium наведено на рис. 5.39 для функції активації Gaussian [261], та на рис. 5.40 для випадку функції активації Inverse Multiquadric [262]. З цих рисунків видно, що у обох випадках занадто велика кількість нейронів вхідного шару призводить до значного зростання похибки прогнозування, що в загальному може бути пояснено труднощами апроксимації випадкового процесу без явно вираженого розподілу протягом тривалого проміжку часу (кількість вхідних нейронів визначає передісторію, яка враховується при прогнозуванні наступного значення).

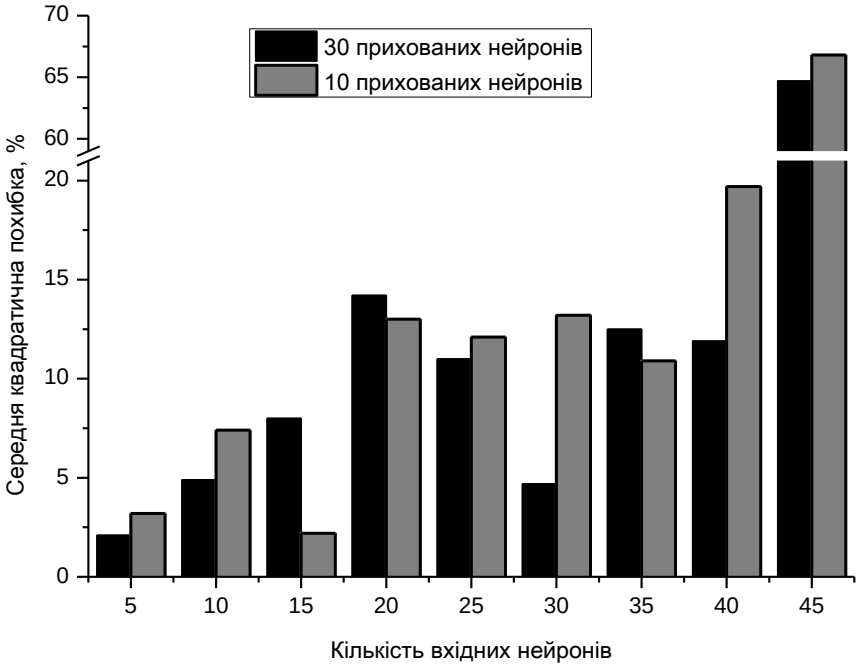


Рис. 5.39. Залежність середньої квадратичної похибки апроксимації від кількості нейронів вхідного шару для функції активації Gaussian.

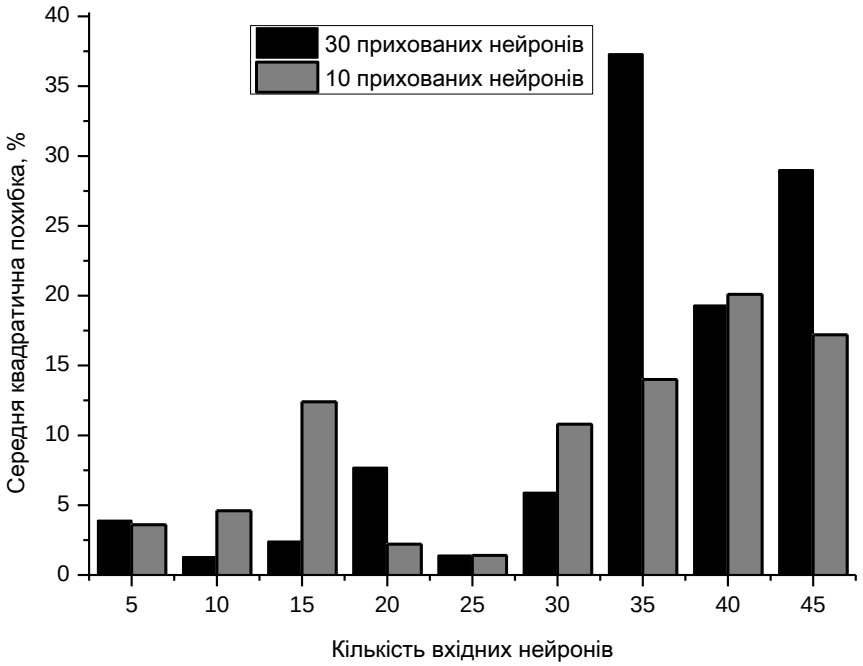


Рис. 5.40. Залежність середньої квадратичної похибки апроксимації від кількості нейронів вхідного шару для функції активації Inverse Multiquadric.

Для порівняння точності прогнозу часових рядів відмов різних програмних систем на рис. 5.41 наведено залежність середньої квадратичної похибки апроксимації від конфігурації НМ (поданої на рисунку у вигляді двох чисел $n-m$, де n – кількість нейронів вхідного шару, m – кількість нейронів прихованого шару) для функції активації Gaussian для обох досліджуваних ПС. Як чітко видно з рис. 5.41, тенденція зміни точності прогнозу в залежності від конфігурації НМ є однаковою для обох ПС – значне зростання похибки зі зростанням кількості нейронів вхідного шару та менші в цілому значення похибки для конфігурації з десятьма нейронами прихованого шару.

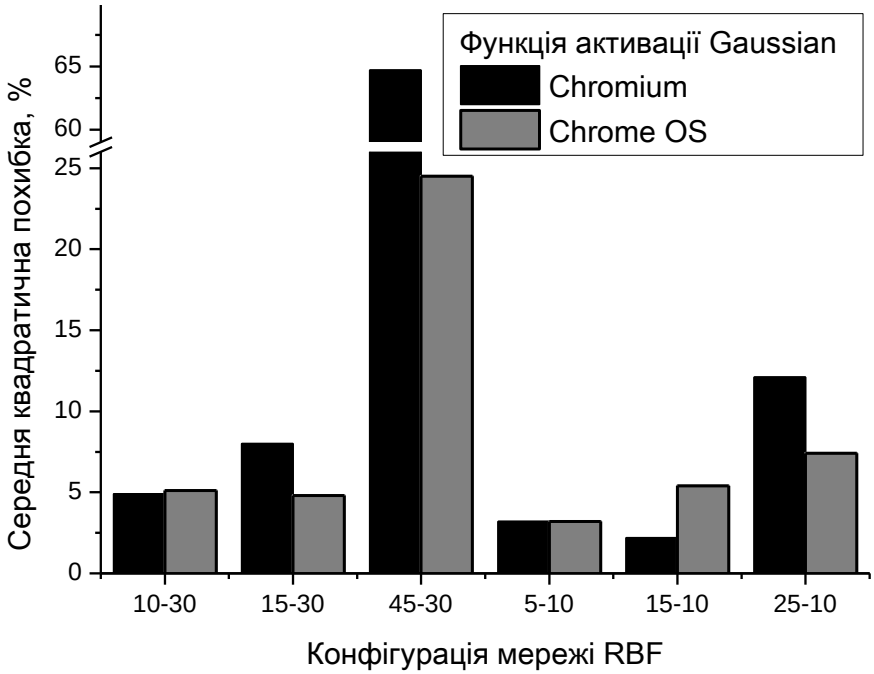


Рис. 5.41. Залежність середньої квадратичної похибки апроксимації від конфігурації НМ для двох ПС у випадку функції активації Gaussian.

Результати дослідження впливу конфігурації НМ на тривалість навчання наведені на рис. 5.42. Тут зображено залежність відносної ефективності навчання (відношення тривалості навчання при використанні функції активації Inverse Multiquadric до тривалості навчання при використанні функції активації Gaussian) від конфігурації НМ RBF для двох досліджуваних ПС. Як видно з рис. 5.42, незважаючи на різні абсолютні значення тривалості навчання (див. табл. 5.3, 5.4) та деякий розкид значень відносної ефективності навчання, загальна тенденція зберігається для обох часових рядів – мінімальна різниця в швидкості навчання спостерігається для конфігурації "45-30", а максимальна – для конфігурацій "15-10" та "25-10". З цього теж можна зробити висновок, що процеси навчання відбуваються приблизно однаково при використанні різних часових рядів зі схожими характеристиками.

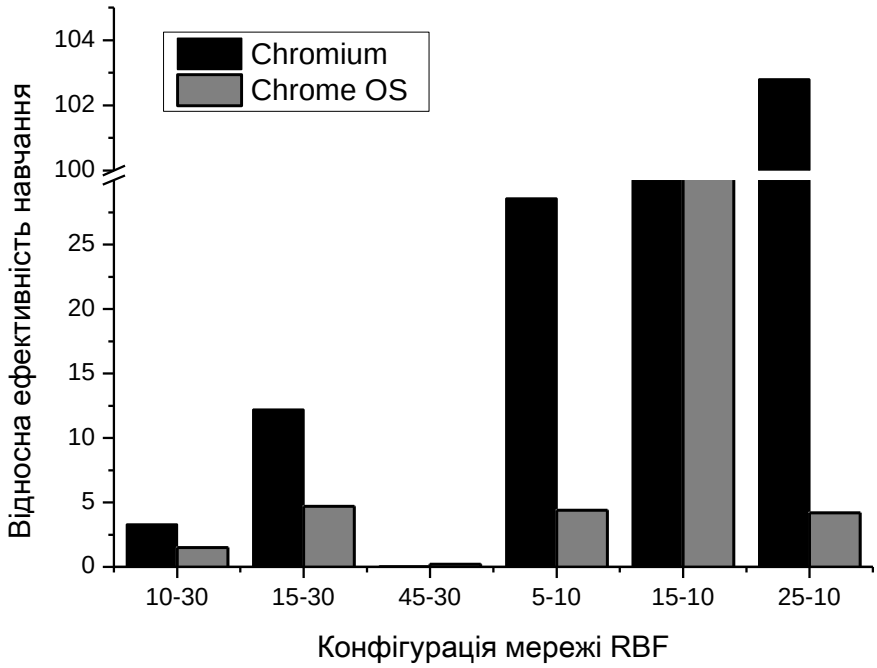


Рис. 5.42. Залежність відносної тривалості навчання для двох функцій активації від конфігурації НМ для двох ПС.

На рис. 5.43 та 5.44 наведено часові залежності прогнозованої та експериментальної кількості відмов браузера Chromium, а на рис. 5.45 – залежності прогнозованої та експериментальної кількості відмов Chromium-OS.

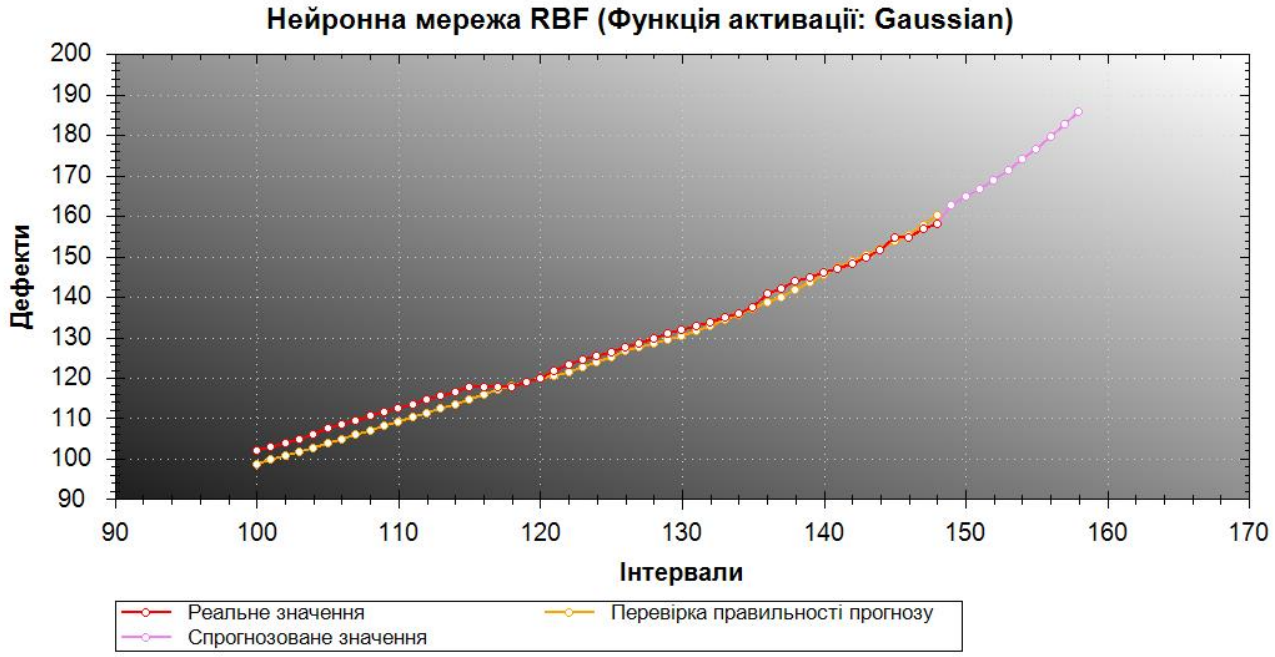


Рис. 5.43. Результати прогнозування відмов браузера Chromium НМ RBF (функція активації Gaussian, 15 нейронів вхідного та 10 – прихованого шару).

Прогнозування здійснювалось з використанням нейронної мережі RBF з такими параметрами: функція активації Gaussian, 15 нейронів вхідного шару, 10 нейронів прихованого шару; тривалість навчання нейронної мережі такої конфігурації становила 78 епох (рис. 5.43); функція активації Inverse Multiquadric, 10 нейронів вхідного шару, 30 нейронів прихованого шару (рис. 5.44, 5.45); тривалість навчання нейронної мережі такої конфігурації становила 478 епох у випадку системи Chromium та 324 епохи у випадку Chromium-OS. Як видно з цих рисунків, підбором параметрів можна досягнути високого ступеня близькості прогнозних та емпіричних даних.

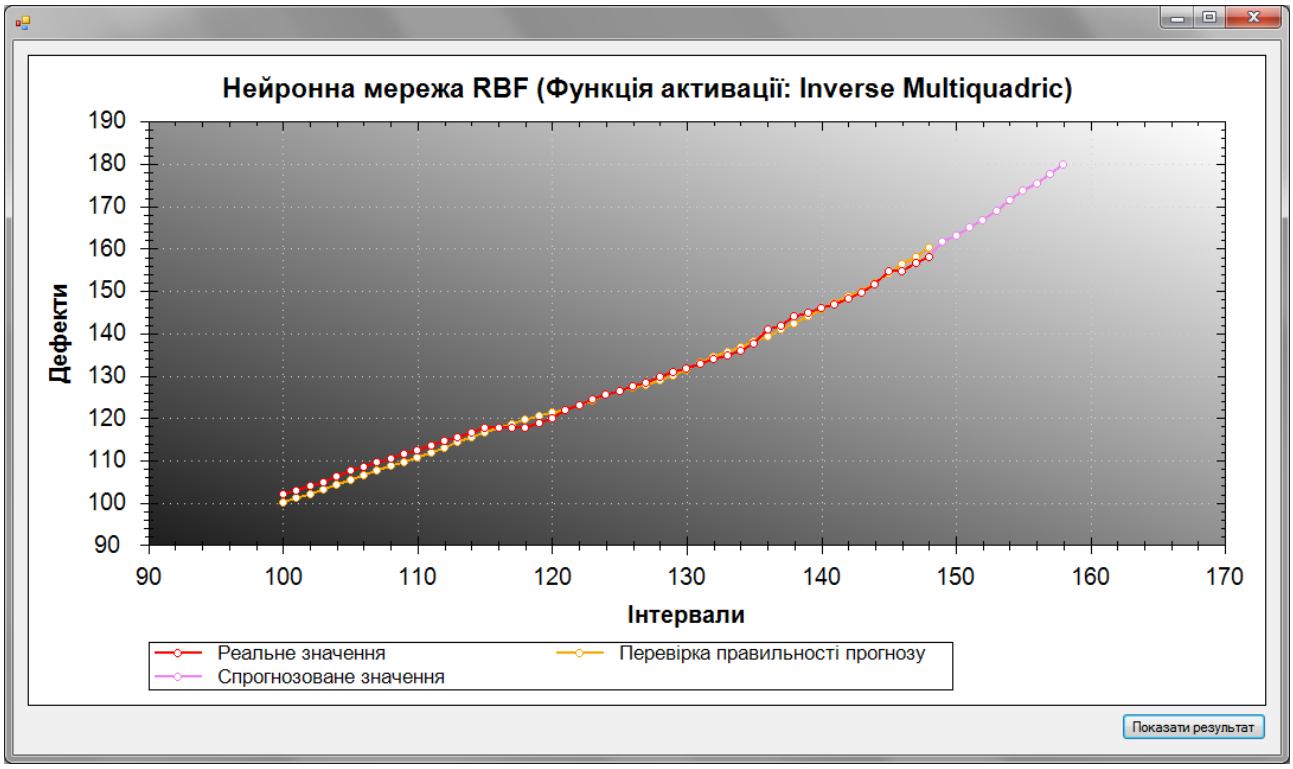


Рис. 5.44. Результати прогнозування відмов браузера Chromium НМ RBF (функція активації Inverse Multiquadric, 10 нейронів вхідного та 30 нейронів прихованого шару).

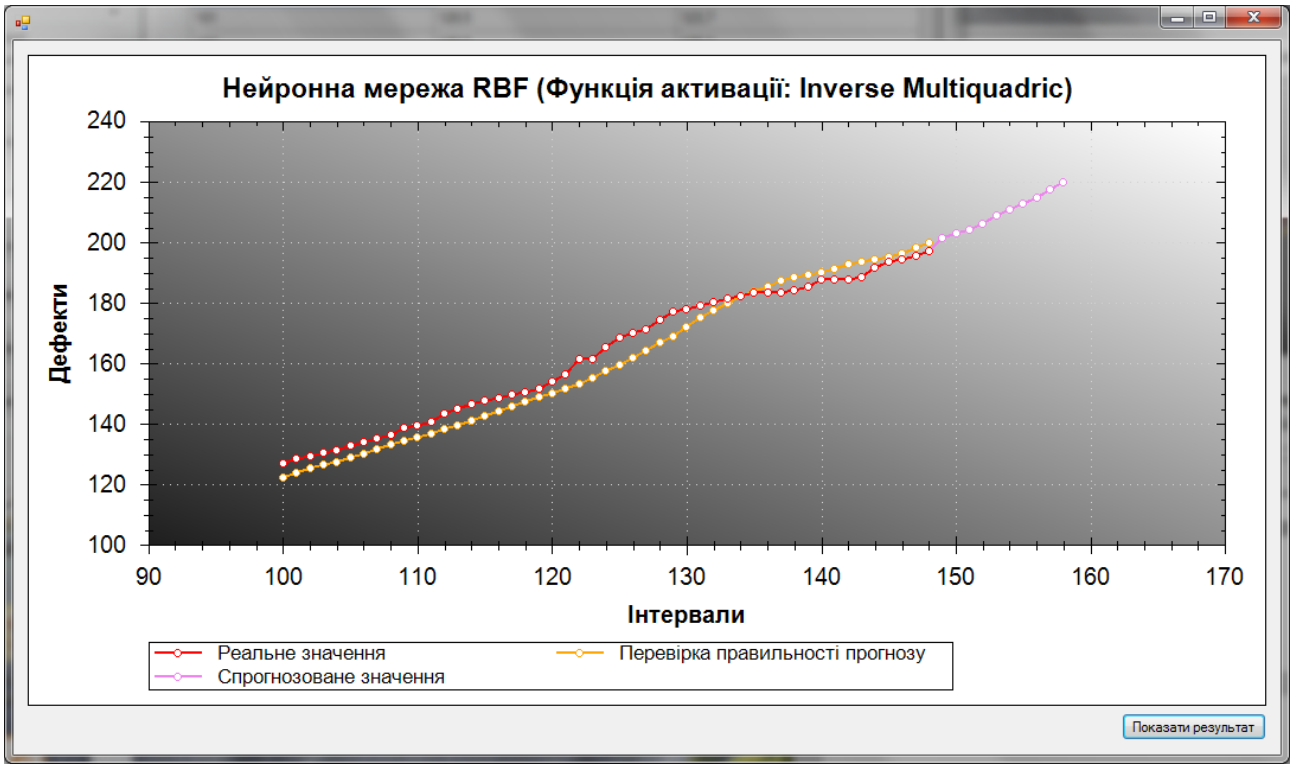


Рис. 5.45. Результати прогнозування відмов Chromium-OS НМ RBF (функція активації Inverse Multiquadric, 10 нейронів вхідного шару та 30 – прихованого).

Як видно з рисунків 5.39, 5.43 у випадку використання в НМ функції активації Gaussian найкращі результати прогнозування отримано при невеликих значеннях кількості вхідних нейронів (до 15), що відповідає врахуванню в прогнозі невеликої передісторії відмов ПС. При кількості вхідних нейронів 40–45 (26,7–30% від навчальної вибірки) результати прогнозування стають неприйнятними, що підтверджується також даними табл. 5.4, отриманими на основі результатів тестування системи Chromium-OS. Разом з тим слід зауважити, що мінімальна тривалість навчання не завжди відповідала максимальній точності прогнозу.

У випадку системи Chromium-OS використання функції активації Gaussian дає найкращу точність прогнозу (середня квадратична похибка 3,2%) в конфігурації "5-10", так само як і у випадку браузера Chromium (див. табл. 5.3, 5.4). При цьому тривалість навчання в цій конфігурації для обох досліджуваних програмних систем є значно меншою у випадку функції активації Gaussian (69

проти 1976 епох у випадку браузера Chromium, та 364 проти 1618 епох у випадку системи Chromium-OS).

У випадку використання функції активації Inverse Multiquadric (рис. 5.40, 5.44, 5.45) можна зробити висновки, що так само як і у випадку функції активації Gaussian, найкращі результати прогнозування отримано при невеликих значеннях кількості вхідних нейронів (до 25), що відповідає врахуванню в прогнозі невеликої передісторії відмов ПС. При кількості вхідних нейронів 40–45 (до 30% від навчальної вибірки) результати прогнозування стають неприйнятними у випадку прогнозування відмов обох досліджуваних в цій роботі ПС. Так само, як і у випадку використання функції активації Gaussian, мінімальна тривалість навчання не завжди відповідала максимальній точності прогнозу. Разом з тим слід зауважити, що використання функції активації Inverse Multiquadric на відміну від Gaussian частіше приводила до паралічу навчання НМ, і в цілому середня тривалість навчання була на порядок вищою. Водночас максимальна точність прогнозу була досягнута при використанні функції активації Inverse Multiquadric (середня квадратична похибка апроксимації становила 1,3% для браузера Chromium та 3,2% для системи Chromium-OS), що підтверджує висновки роботи [258].

З табл. 5.4 видно, що у випадку часового ряду, побудованого на основі тестування ПС Chromium-OS, найкращі результати прогнозування відмов (середня квадратична похибка апроксимації 3,2%, коефіцієнт детермінації 0,983) були досягнуті з використанням функції активації Inverse Multiquadric та конфігурації "10-30" (10 вхідних нейронів та 30 нейронів прихованого шару), що ілюструється рис. 5.45. Так, з табл. 5.4 видно, що дещо меншої похибки апроксимації для цієї ПС можна досягнути з використанням функції активації Inverse Multiquadric та конфігурацій "5-10" чи "15-10" (2,4% та 3,0% відповідно), однак при цьому тривалість навчання становила 1618 та 1536 епох проти 324 епох у випадку конфігурації "10-30". З іншого боку функція активації Gaussian в конфігурації "5-10" дає можливість досягнути схожої похибки апроксимації при співмірній тривалості навчання, однак коефіцієнт детермінації в даному випадку

є дещо гіршим – 0,958 (табл. 5.4). Таким чином висновок про найкращу точність прогнозування часового ряду відмов ПС зазначеною конфігурацією НМ типу RBF підтверджується також для відмов ПС Chromium-OS з іншими метриками та складністю, що дає підстави для висновку про можливість застосування такої конфігурації для прогнозування часових рядів відмов різних ПС.

5.5. Висновки до розділу 5

Розроблено програмний засіб попереднього опрацювання даних про відмови ПС, які є вхідною інформацією для моделей надійності типу "чорної скриньки", зокрема МНПС, а також використовуються для отримання показників надійності модулів програмних систем у випадку використання архітектурних моделей. Програмну реалізацію засобу здійснено на мові Java з використанням середовища розробки Eclipse Kepler. Інтеграцію з системою JIRA здійснено за допомогою JIRA REST API.

Розроблено програмний засіб для оцінювання ймовірностей переходів між модулями програмної системи, який складається з: аспектно-орієнтованого реєстратора, що здійснює модифікацію байт-коду ПС, написаних на мові Java; веб-додатку для перегляду та аналізу результатів. Результати роботи засобу зберігаються в базі даних MySQL, для розроблення веб-додатку було використано Spring MVC Tomcat, jQuery та jQuery UI. Засіб для отримання ймовірностей використання модулів ПС дає можливість отримати статистику виконання класів, часу виконання класів та переходів між класами досліджуваної ПС, зберегти цю статистику в базі даних та візуалізувати результати у вигляді графіків та діаграм за допомогою веб додатку.

Розроблено програмний засіб оцінювання надійності ПС на основі архітектурних моделей "СОН ПЗ". Цей засіб складається з модуля для отримання моделі функціонування ПС; модуля АФСТ на основі моделі функціонування ПС та модуля для оцінювання показників надійності ПС з використанням ЛМВП. Програмний засіб "СОН ПЗ" реалізовано мовою програмування C#, а основні функції винесено у вигляді динамічної бібліотеки DLL, що дає можливість як

легкого підключення цієї бібліотеки до існуючих програмних комплексів, так і розширення функціоналу програмного засобу за рахунок додавання нових моделей надійності ПС.

Розроблено програмний засіб для прогнозування відмов ПС на основі непараметричних моделей, зокрема штучних нейронних мереж, які не вимагають апріорних припущень про структуру чи складність ПС, а також про поведінку його відмов. Засіб розроблено з використанням платформи .NET, мови програмування C# та бібліотеки машинного навчання з відкритим кодом Encog.

Проведено дослідження впливу архітектури та параметрів НМ Елмана та RBF на ефективність прогнозування відмов програмних систем Chromium та Chromium-OS. Показано, що рекурентна НМ Елмана при належному підборі параметрів її архітектури (функція активації "гіперболічний тангенс" або лінійна, кількість нейронів прихованого шару не менша, ніж кількість нейронів вхідного) здатна прогнозувати відмови ПС значного ступеня складності (що показано на прикладі браузера Chromium) з достатньою точністю (середня квадратична похибка апроксимації не більше 10 при значеннях функції 100–160 відмов).

Показано, що при використанні нейронної мережі RBF найкращі результати прогнозування відмов ПС були отримані в конфігурації з функцією активації Inverse Multiquadric, 10 нейронами у вхідному шарі та 30 нейронами у прихованому. У такій конфігурації середня квадратична похибка апроксимації досягає 1,0%, а коефіцієнт детермінації між прогнозованою та контрольною вибірками – 0,9965. Разом з тим, за рахунок незначного погіршення точності прогнозу до 1,7%, можна зменшити тривалість навчання нейронної мережі у 3–6 разів завдяки використанню функції активації Gaussian з 15 вхідними нейронами при десяти нейронах прихованого шару. У такій конфігурації відносна середня квадратична похибка апроксимації становить 1,7%, а коефіцієнт детермінації між прогнозованою та контрольною вибірками – 0,9969. Натомість функція активації Mexican Hat є непридатною для вирішення такої задачі прогнозування відмов ПС засобами НМ RBF. Отримання подібних

результатів для двох різних за характеристиками програмних систем дає підстави для висновку, що отримана конфігурація нейронної мережі RBF може бути успішно використана для прогнозування часових рядів зі схожими характеристиками – однорідний процес відмов, представлений у вигляді кумулятивного часового ряду.

ВИСНОВКИ

У дисертаційній роботі вирішено науково-прикладну проблему створення моделей, методів та засобів аналізу надійності програмних систем підвищеного ступеня адекватності, що враховують процеси життєвого циклу ПЗ, його архітектуру та складність. Отримано такі основні наукові та практичні результати:

1. Розроблено модель надійності ПЗ з показником складності на основі неоднорідного пуассонового процесу, яка, на відміну від наявних, враховує складність ПЗ і належить до узагальнених моделей на основі неоднорідного пуассонового процесу. Зокрема, модель надійності Goel–Okumoto є частковим випадком цієї моделі при $s = 0$, а S-подібна модель надійності – при $s = 1$.
2. Використання моделі надійності ПЗ з показником складності дає можливість підвищити точність оцінки кількості невиявлених помилок у програмі на 3–7 %, а точність оцінки інтенсивності відмов ПЗ до трьох порядків порівняно з найбільш поширеними моделями надійності цього класу. При цьому моделі Goel–Okumoto та S-подібна дають занижені значення інтенсивності відмов ПЗ, що призводить до завищених значень функції та коефіцієнта готовності ПАС, до складу яких входить це ПЗ.
3. Розроблено метод оцінювання показників надійності ПЗ з урахуванням його складності, але без урахування архітектури, який базується на моделі надійності ПЗ з показником складності та критерії достатності процесу тестування, що дає можливість більш достовірно оцінювати показники надійності ПЗ.
4. Розроблено три моделі оцінювання показників надійності складних програмних систем, що використовують ланцюги Маркова вищого порядку – з дискретним часом; з неперервним часом; з неперервним часом та урахуванням відмов інтерфейсів модулів. Ці моделі, на відміну від наявних, враховують вплив потоку управління на інтенсивність відмов програмної

системи, та відмови при передачі управління між модулями, що дає можливість підвищити достовірність оцінювання показників надійності програмних систем на 6–10 % у випадку використання моделі з дискретним часом та на 10–20 % у випадку моделі з неперервним часом.

5. Розроблено метод подання марковського процесу вищого порядку у вигляді еквівалентного процесу першого порядку з віртуальними станами для програмних систем, в яких існує взаємозалежність виконання модулів, що дає можливість звести роботу з ланцюгами Маркова вищого порядку до класичних ланцюгів Маркова. Перевагою цього методу є можливість застосування до марковських процесів будь-якого порядку (в тому числі змінного) та безпосереднє використання отриманої матриці інтенсивностей переходів для побудови системи рівнянь Колмогорова–Чепмена.
6. Розроблено узагальнений метод аналізу надійності програмних систем з урахуванням їх складності, архітектури та етапів ЖЦ. Цей метод, на відміну від інших, базується на побудованих моделях надійності та засобах отримання параметрів цих моделей, та дає можливість оцінювання та прогнозування показників надійності програмних систем на різних етапах ЖЦ.
7. На основі побудованих моделей і методів показано шляхи підвищення ефективності розробки ПЗ з урахуванням вимог до його надійності, які містять: критерій достатності процесу тестування; метод прогнозування кількості невиявлених помилок в ПЗ; модель функціонування програмних систем з урахуванням змінних коду та архітектури; методи автоматизованого формування сценаріїв тестування ПЗ за стратегіями "чорної" та "білої скриньки".
8. Використання критерію достатності процесу тестування виступає як числова метрика, на основі якої можна приймати рішення стосовно процесу тестування ПЗ. Метод прогнозування кількості невиявлених помилок в ПЗ дає можливість покращити точність оцінки кількості невиявлених помилок на 3–5 %, або ж скоротити тривалість тестування на 30–40 % (при збереженні

точності прогнозу кількості невиявлених помилок). Засоби автоматизованого формування сценаріїв тестування забезпечують рівномірне покриття програмного коду тестами та дають можливість зменшити часові, фінансові та людські ресурси на етапі тестування; на основі побудованої моделі функціонування програмних систем розроблено рекомендації щодо вибору стратегії тестування ПЗ.

9. Показано, що рекурентна нейронна мережа Елмана при належному підборі параметрів її архітектури здатна прогнозувати відмови програмних систем значного ступеня складності з достатньою точністю (середня квадратична похибка апроксимації не більша 6–10 %). При використанні нейронної мережі RBF найкращі результати прогнозування відмов програмних систем отримують в конфігурації з функцією активації "Inverse Multiquadric", 10 нейронами у вхідному шарі та 30 нейронами у прихованому. У такій конфігурації середня квадратична похибка апроксимації сягає 1,0%, а коефіцієнт детермінації між прогнозованою та контрольною вибірками – 0,9965. Разом з тим, за рахунок незначного погіршення точності прогнозу до 1,7%, можна зменшити тривалість навчання нейронної мережі у 3–6 разів завдяки використанню функції активації Гауса з 15 вхідними нейронами при 10 нейронах прихованого шару.
10. Розроблені моделі та методи реалізовані у вигляді програмних засобів, які дають можливість здійснювати оцінювання показників надійності програмних систем на основі архітектурного підходу та прогнозувати відмови програмних систем з використанням штучних нейронних мереж Ефективність розроблених програмних засобів підтверджується результатами їх впровадження у виробничі процеси ДП НДІ "Система" та ПП "Лінк Ап Студіо".

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 2844-94. Програмні засоби ЕОМ. Забезпечення якості. Терміни та визначення. – К. : Держстандарт України, 1996. – 19 с.
2. ДСТУ 2850-94. Програмні засоби ЕОМ. Показники і методи оцінювання якості. – К. : Держстандарт України, 1996. – 20 с.
3. Половко А. М. Основы теории надежности / А. М. Половко, С. В. Гуров. – СПб. : БХВ-Петербург, 2006. – 704 с.
4. Муляк О.В. Надійнісні моделі програмно-апаратних засобів радіотехнічних систем з версійно-структурним резервуванням [Текст] : дис. … канд. техн. наук : 05.12.17 : захищена 29.11.14 : затв. 26.02.15 / Муляк Олександр Володимирович ; Нац. ун-т "Львівська політехніка". – Львів, 2014. – 247 с.
5. Гнеденко Б. В. Математические методы в теории надежности / Б. В. Гнеденко, Ю. К. Беляев, А. Д. Соловьев. – М. : Издательство "Наука", 1965. – 524 с.
6. Hecht H. Can software benefit from hardware experience? / H. Hecht // Proceedings of the Annual Reliability and Maintainability Symposium. – 1975. – P. 480–484.
7. Soi I.M. Hardware vs. software reliability – a comparative study / I.M. Soi, K. Gopal // Microelectronics and Reliability. –1980. –Vol. 20 – P. 881–885.
8. Pham H. Software Reliability Models for Critical Applications / H. Pham, M. Pham // EGG—2663 Technical Report. Idaho National Engineering Laboratory, EG&G Idaho Inc. – 1991. – 98 p.
9. Pham H. System software reliability // Springer-Verlag London Limited, 2006. – 440 p.
10. Липаев В. В. Надежность программных средств / В. В. Липаев. – М. : СИНТЕГ, 1998. – 232 с.
11. Волочій Б. Ю. Технологія моделювання алгоритмів поведінки інформаційних систем / Б. Ю. Волочій. – Л. : Вид-во Нац. ун-ту "Львів. політехніка", 2004. – 220 с.
12. Lyu M. R. The handbook of software reliability engineering / Michael R. Lyu. – New York, NY : Computing McGraw-Hill, 1996. – 819 p.

13. Lyu M. R. Software Reliability Engineering: A Roadmap / Michael R. Lyu // Future of Software Engineering (FOSE'07), Minneapolis, USA, May 2007. – P. 153–170.
14. Lai R. A Detailed Study of NHPP Software Reliability Models / R. Lai, M. Garg // Journal of Software. – 2012. – Vol. 7, No 6. – P. 1296–1306.
15. Gokhale S.S. An Analytical Approach to Architecture-Based Software Performance and Reliability Prediction / S.S. Gokhale, W.E. Wong, J.R. Horgan, K.S. Trivedi // Performance Evaluation. – 2004. – Vol. 58, No 4. – P. 391–412.
16. Cheung L. Early prediction of software component reliability / Leslie Cheung, Roshanak Roshandel, Nenad Medvidovic, Leana Golubchik // Proceedings of the 30th international conference on Software engineering. – 2008. – P. 111–120.
17. Mohanta S. An approach for early prediction of software reliability / S. Mohanta, G. Vinod, A.K. Ghosh, R. Mall // ACM SIGSOFT Software Engineering Notes archive. – 2010. – Vol. 35, Issue 6. – P. 1–9.
18. Lyu M.R. Reliability-oriented software engineering: design, testing and evaluation techniques / M.R. Lyu // Software, IEE Proceedings. – 1998. – Vol. 145, No 6. – P. 191–197.
19. Одарущенко О.Н. CASE-оценка критических программных систем. Т. 2. Надежность: монография / О. Н. Одарущенко, В. С. Харченко, Д. А. Маевский, Ю. Л. Поночовный, А. А. Руденко, Е. Б. Одарущенко, С. А. Засуха, В. О. Жадан, С. В. Живило. – Х: ХАИ. – 2012. – 292 с.
20. Соколов Ю.Н. Инструментальное оценивание надёжности программно-технических комплексов при росте интенсивности отказов / Соколов Ю.Н., Харченко В.С., Поночёвный Ю.Л. // Системи обробки інформації. – Харків: ХУПС. – 2014. – Вип. 2 (118). – С. 205–211.
21. Malaiya Y. K. Software Reliability Models: Theoretical Developments, Evaluation and Applications / Malaiya Y.K., Srimani P.K. – Los Angeles : IEEE Computer Society Press, 1990. – 372 p.
22. Тейер Т. Надежность программного обеспечения : пер. с англ. / Т. Тейер, М. Липов, Э. Нельсон. – М. : Мир, 1981. – 323 с.

23. Tariq Hussain Sheakh. A Study of Analytically Improving the Reliability of Software / Tariq Hussain Sheakh, S.M.K. Quadri, and VijayPal Singh // International Journal of Research and Reviews in Computer Science. – 2012. – Vol. 3, No 1. – P. 1404–1406.
24. ДСТУ 3918-99 (ISO/IEC 12207: 1995) Інформаційні технології. Процеси життєвого циклу програмного забезпечення. – К.: Держстандарт України, 2002. – 49 с.
25. Cobra Rahmani. Exploitation of Quantitative Approaches to Software Reliability / Cobra Rahmani, Azad Azadmanesh // University of Nebraska at Omaha, 2008. – 32 p.
26. Goel A.L. Software reliability models: assumptions, limitations, and applicability / A.L. Goel // IEEE Transactions on software engineering. – 1985. – Vol. SE-11, No 12. – P. 1411–1423.
27. Maevsky D. A. A New Approach to Software Reliability / Dmitry A. Maevsky // Lecture Notes in Computer Science : Software Engineering for Resilient Systems.– 2013. – № 8166. – P. 156–168.
28. Маевский Д. А. Основы теории устойчивости программмных систем / Д. А. Маевский // Електротехнічні та комп’ютерні системи. – 2012. – № 5. – С. 221–228.
29. Конорев Б.М. Квалификационные испытания критического программного обеспечения космических систем: целевая технология независимой верификации и прогнозирования скрытых дефектов / Б.М. Конорев, Ю.Г. Алексеев, С.А. Засуха, Л.П. Семенов, В.С. Харченко, Г.Н. Чертков // Космічна наука і технологія. – 2008. – т. 14, № 4.– С. 9–26.
30. Руденко А.А. Модели оценки надежности программных средств с учетом недетрминированного числа вторичных дефектов / А.А. Руденко, О.Н. Одарущенко, В.С. Харченко // Радіоелектронні і комп'ютерні системи. – 2010. – №6 (47). – С. 197–203.

31. Одарущенко О.Н. Метод оценивания надежности программных средств с учетом вторичных дефектов / Одарущенко О.Н., Руденко А.А., Харченко В.С. // Радіоелектронні і комп'ютерні системи. – 2012. – № 7 (59). – С. 294–300.
32. Конорев Б.М. Прогнозирование вероятности скрытых дефектов критического ПО с заданной точностью / Б.М. Конорев, В.В. Сергиенко, В.С. Харченко, Г.М. Жолткевич // Радіоелектронні і комп'ютерні системи. – 2014. – № 5 (69). – С. 50–54.
33. Goševa-Popstojanova K. Architecture-based approach to reliability assessment of software systems / K. Goševa-Popstojanova, K.S. Trivedi // Performance Evaluation. – 2001. – Vol. 45. – P. 179–204.
34. Hamill M. Common Trends in Software Fault and Failure Data / M. Hamill, K. Goševa-Popstojanova // IEEE Transactions on Software Engineering. – 2009. – Vol. 35, No 4. – P. 484–496.
35. Palviainen M. The reliability estimation, prediction and measuring of component-based software / M. Palviainen, A. Evesti, E. Ovaska // The Journal of Systems and Software. – 2011. –Vol. 84. – P. 1054–1070.
36. Reussner R.H. Reliability prediction for component-based software architectures / R.H. Reussner, H.W. Schmidt, I.H. Poernomo // The Journal of Systems and Software. – 2003. – Vol. 66. – P. 241–252.
37. Durand J.B. Software reliability modelling and prediction with hidden Markov chains / J.B. Durand, O. Gaudoin // Statistical Modelling. –2005. – Vol. 5 (1). – P. 75–93.
38. Яковина В.С. Огляд основних підходів до аналізу надійності програмного забезпечення / В. Яковина, В. Смірнов // Вісник Національного університету "Львівська політехніка". Комп'ютерні науки та інформаційні технології. – 2011. – № 719. – С. 278–282.
39. Маевский Д.А. Оценка количества дефектов программного обеспечения на основе метрик сложности / Д.А. Маевский, С.А. Яремчук // Електротехнічні та комп’ютерні системи. – 2012. – № 7. – С. 113–120.

40. Halstead M.H. Elements of Software Science // New York: Elsevier North-Holland Publishing, 1977. – 127 p.
41. McCabe T.J. A Complexity Measure / T.J. McCabe // IEEE transaction on Software Engineering. – 1976. – Vol. SE-2, No 4. – P. 308–320.
42. Маевский Д.А. Верификация моделей надежности программного обеспечения / Д.А. Маевский, Е.Ю. Маевская, О.П. Жеков // Радіоелектронні і комп'ютерні системи. – 2014. – № 5 (69). – С. 55–59.
43. Kharchenko, V.S. The Method of Software Reliability Growth Models Choice Using Assumptions Matrix / V.S. Kharchenko, O.M. Tarasyuk, V.V. Sklyar, V.Yu. Dubnitsky // Computer Software and Applications: Proceedings of the 26th Annual Int. Conf., Oxford, England, 26-29 August 2002. – 2002. – P. 541-546.
44. Основи надійності цифрових систем [Текст] : підручник / В.С. Харченко, В.Я. Жихарев, В.М. Ілюшко, В.А. Краснобаєв, П.М. Куліков, І.В. Лисенко, М.В. Нечипорук, Г.М. Тимонькін. – Харків: Нац. аерокосм. ун-т "Харк. авіац. ін-т", 2004. – 573 с.
45. Mills H.D. On the statistical validation of computer programs // IBM Federal Syst. Div., Gaithersburg, MD, Rep. 72-6015, 1972.
46. Jelinski Z. Software reliability research / Z. Jelinski, P. Moranda // Statistical Computer Performance Evaluation. – 1972. – P. 465–484.
47. Schick G.J. An analysis of competing software reliability models / G.J. Schick, R.W. Wolverton // IEEE Transactions on Software Engineering. – 1978. – Vol. SE-4, No. 2. – P. 104–120.
48. Moranda P.B. An error detection model for application during software development / P.B. Moranda // IEEE Transactions on Reliability. – 1979. – Vol. R-28, No 5. – P. 325–329.
49. Littlewood B. Software reliability model for modular program structure / B. Littlewood // IEEE Transactions on Reliability. – 1979. – Vol. R-28, No. 3. – P. 241–246.

50. Goel A.L. An analysis of recurrent software failures in a real-time control system / A. L. Goel, K. Okumoto // Proc. ACM Annu. Tech. Conf., ACM. – 1978. – P. 496–500.
51. Quadri S.M.K. Software Reliability Growth Modeling with New Modified Weibull Testing-effort and Optimal Release Policy / S.M.K. Quadri, N. Ahmad // International Journal of Computer Applications. – 2010. – Vol. 6, No 12. – P. 1–10.
52. Dhillon B.S. Reliability Engineering in Systems Design and Operation // New York : Van Nostrand Reinhold Co., 1983. – 319 p.
53. Wall J.K. Pragmatic software reliability prediction / J.K. Wall, P.A. Ferguson // Proceedings Annual Reliability & Maintainability Symposium, IEEE Reliability Society. – 1977. – P. 485–488.
54. Wagoner W.L. The final report on a software reliability measurement study / W.L. Wagoner // Report TOR-0074(41221)-l. The Aerospace Corp., El Segundo, CA, 1973.
55. Yamada S. Software reliability growth modeling: Models and applications / S. Yamada, S. Osaki // IEEE Transactions on Software Engineering. – 1985. – Vol. SE-11, No. 12. – P. 1431–1437.
56. Cheung R.C. An user-oriented software reliability model / R.C. Cheung // IEEE Transactions on Software Engineering. – 1980. – Vol. SE-6, No 2. – P. 118–125.
57. Yakovyna V.S. Review of architecture-based software reliability models / V. Yakovyna, O. Nytrebych // VIth International Scientific and Technical Conference "Computer Science and Information Technologies" CSIT 2011, Lviv, November 16–19, 2011 : Proceedings. – P. 106–109.
58. Nelson E. Estimating software reliability from test data / E. Nelson // Microelectron. Rel. – 1978. – Vol. 17. – P. 67–74.
59. Ramamoorthy C.V. Software reliability: Status and perspectives / C.V. Ramamoorthy, F.B. Bastani // IEEE Trans. Software Eng. – 1982. – Vol. SE-8. – P. 359–371.

60. Shooman M.L. Structure models for softwarereliability prediction / M.L. Shooman // Proceedings of the 2nd International Conference on Software Engineering. – 1976. – P. 268–280.
61. Musa J.D. A theory of software reliability and its application / J.D. Musa // IEEE Transactions on Software Engineering. –1975. – Vol. SE-1(3). –P. 312–327.
62. Goel A.L. Time-Dependent Error-Detection Rate Model for Software and other Performance Measures / A.L. Goel, K. Okumoto // IEEE Transactions on Reliability. – 1979. – Vol. R-28, No 3 – P. 206–211.
63. Yamada S. S-shaped reliability growth modeling for software error detection / S. Yamada, M. Ohba, S. Osaki // IEEE Transactions on Reliability. – 1983. – Vol. R-32, No 5. – P. 475–478.
64. Huang X.Z. The hypergeometric distribution model for predicting the reliability of software / X.Z. Huang // Microelectronics and Reliability. – 1984. – Vol. 24, No. 1. – P. 11–20.
65. Тимошенко Ю.О. Узагальнена модель негомогенного пуассонівського процесу для оцінювання надійності програмного забезпечення / Ю. О. Тимошенко, М. В. Дідковська // Проблеми програмування. – 2004. – № 2–3. – С. 480–489.
66. Чабанюк Я.М. Побудова і дослідження моделі надійності програмного забезпечення з індексом величини проекту / Я.М. Чабанюк, В.С. Яковина, Д.В. Федасюк, М.М. Сенів, У.Т. Хімка // Інженерія програмного забезпечення. – 2010. – № 1. – С. 24–29.
67. Kwon Y.-M. A Markov Reward Model for Software Reliability / Y.-M. Kwon, G. Agha // The Next Generation Software (NGS) Workshop at International Parallel and Distributed Processing Symposium (IPDPS). – 2007. – P. 1–6.
68. Whittaker J.A. A Markov chain model for predicting the reliability of multi-build software / J.A. Whittaker // Information and Software Technology. – 2000. – Vol. 42. – P. 889–894.
69. Littlewood B. A reliable model for systems with Markov structure / B. Littlewood // Applied Statistics. – 1975. – Vol. 24, No 2. – P. 172–177.

70. Ohba M. S-shaped software reliability growth models / M. Ohba, S. Yamada // Proceedings of the 4th International Conference on Reliability and Maintainability. – 1984. – P. 430–436.
71. Nishi Kareer. An S-shaped software reliability growth model with two types of errors / Nishi Kareer, P.K. Kapur, P.S. Grover // Microelectronics Reliability. – 1990. – Vol. 30, Issue 6. – P. 1085–1090.
72. Kremer W. Birth-death and bug counting / W. Kremer // IEEE Transactions on Reliability. – 1983. – Vol. R-32. – P. 37–47.
73. Musa J.D. A Logarithmic Poisson Execution Time Model for Software Reliability Measurement / J.D. Musa, K. Okumoto // Proceedings of the 7th International Conference on Software Engineering. – 1984. – P. 230–238.
74. Харченко В.С. Гарантоспособность и гарантоспособные системы: элементы методологии / В.С. Харченко // Радіоелектронні і комп'ютерні системи. – 2006. – №5 (17). – С. 7–19.
75. Харченко В. С. Технологии высокой готовности для программно-технических комплексов космических систем: монографія / В. С. Харченко, О. Н. Одарущенко, Ю. Л. Поночовный, Е. Б. Одарущенко, В. В. Скляр, Б. М. Конорев, Г. Н. Чертков. – Х: ХАИ. – 2010. – 372 с.
76. Brooks W. D. Analysis of discrete software reliability models / W. D. Brooks, R. W. Motley // Techn. Rep. Rome Air Development Center. – 1980. – RADC-TR-80-84.
77. Shooman M. L. Probabilistic models for software reliability prediction / M. L. Shooman // Statist. Computer Performance Evaluation. – 1972. – P. 485–502.
78. Bell D. Elliott. Secure computer systems: mathematical fondations and model / Bell D. Elliott, Leonard J. La Padula // MITRE Techn. Reports. – Bedford, Massachusetts, 1974. – Rep. M74-244 : October.
79. Bell David Elliott. Looking back at the Bell-La Padula model / Bell David Elliott // Proc. of the 21st Annual computer security applications conf. –2005. – P. 337–351.
80. Moranda P. B. Prediction of software reliability during debugging / P. B. Moranda // Proc. of Annu. reliability and maintainability symp. – 1975. – P. 327–332.

81.Schick G. J. Assessment of software reliability / G. J. Schick, R. W. Wolverton // Proc. Oper. Res. – Wirzberg-Wien : Physica-Verl. – 1973. – P. 395–422.

82.Musa J. D. Software reliability: measurement, prediction, application / J. D. Musa, A. Iannino, K. Okumoto. – New York : McGraw-Hill, 1987. – 621 p.

83.Musa J. D. Software reliability engineering / J. D. Musa. – New York : McGraw-Hill, 1999. – 380 p.

84.Schneidewind N. F. Analysis of error process in computer software / N. F. Schneidewind // Sigplan Note. – 1975. – Vol. 10, № 6. – P. 337–346.

85.ДСТУ 2860-94. Надійність Техніки. Терміни та визначення. – К. : Держстандарт України, 1994. – 33 с.

86.American Institute of Aeronautics and Astronautics. Recommended Practice for Software Reliability. // ANSI/AIAA Report 0131992. AIAA, 1992.

87.Ohba M. Software reliability analysis models / M. Ohba // IBM J. Res. Develop. – 1984. – Vol. 28, № 4. – P. 428–443.

88.P.K. Kapur, R.B. Garg "A software reliability growth model for an error-removal phenomenon" // Software Engineering Journal, Vol. 7 (1992), Issue 4, pp. 291–294.

89.H. Pham "Handbook of Reliability Engineering". – Springer-Verlag London limited, 2003. – 663 p.

90.Cai K.-Y. Does software reliability growth behavior follow a non-homogeneous Poisson process? / K.-Y. Cai, D.-B. Hu, C.-G. Bai, H. Hu, T. Jing. // Information and Software Technology. – 2008. – Vol. 50. – P. 1232–1247.

91.Lipow M. Estimation of software package residual errors / M. Lipow // Software Series Rep. – Redondo Beach, CA. – 1972. – TRW-SS-72-09.

92.Brown J. R. Testing for software reliability / J. R. Brown, M. Lipow // Proc. of Int. conf. reliable software, Los Angeles, CA, Apr. 1975. – Los Angeles, 1975. – P. 518–527.

93.Boehm B. W. Quantitative evaluation of software quality / B. W. Boehm, J. R. Brown, M. Lipow // Proc. of 2nd Int. conf. software eng. –1976. – P. 592–605.

94.Basin S. L. Estimation of software error rate via capture-recapture sampling / S. L. Basin. – Palo Alto, CA, 1974.

95. Bastani F. B. An input domain based theory of software reliability and its application : dissertation / F. B. Bastani ; Univ. California. – Berkeley, 1980.

96. Математичні моделі та методи аналізу надійності радіоелектронних, електротехнічних та програмних систем : монографія / Ю.Я. Бобало, Б.Ю. Волочій, О.Ю. Лозинський, Б.А. Мандзій, Л.Д. Озірковський, Д.В. Федасюк, С.В. Щербовських, В.С. Яковина. – Львів : Видавництво Львівської політехніки, 2013. – 300 с.

97. Yubo Yuan. Software reliability modeling with removed errors and compounded-decreased-rate / Yubo Yuan, Houying Zhu, Bo Liu, Feilong Cao // Mathematical and Computer Modelling. – 2012. – Vol. 55. – P. 697–709.

98. Sy-Yen Kuo. Framework for Modeling Software Reliability, Using Various Testing-Efforts and Fault-Detection Rates / Sy-Yen Kuo, Chin-Yu Huang, Michael R. Lyu // IEEE Transactions on Reliability. – 2001. – Vol. 50, No 3. – P. 310–320.

99. Маевский Д. А. Анализ моделей надежности программного обеспечения гарантоспособных информационных систем / Д. А. Маевский, С. А. Яремчук // Електромашинобудування та електрообладнання. – 2010. – № 76. – С. 68 – 79.

100. Szyperski S. B. Object-Oriented Programming. Second edition / S.Szyperski, D.Gruntz, S. Murer. – Great Britan : Addison-Wesley Longman Publishing. – 2010. – 595 p.

101. Xie M. An additive reliability model for the analysis of modular software failure data / M. Xie, C. Wohlin // Proceedings of the Sixth International Symposium on Software Reliability Engineering. – 1995. – P. 188–194.

102. Krishnamurthy S. On the estimation of reliability of a software system using reliabilities of its components / S. Krishnamurthy, A.P. Mathur // Proceedings of the Eighth International Symposium on Software Reliability Engineering . – 1997. – P. 146–155.

103. Gokhale S. Structure-based software reliability prediction / S. Gokhale, K. Trivedi // Proceedings of the Fifth International Conference on Advanced Computing . – 1997. – P. 447–452.

104. Gokhale S. An analytical approach to architecture based software reliability prediction / S. Gokhale, W.E. Wong, K. Trivedi, J.R. Horgan // Proceedings of the Third International Computer Performance and Dependability Symposium. – 1998. – P. 13–22.

105. Horgan J.R. ATAC: a data flow coverage testing tool for C / J.R. Horgan, S. London // Proceedings of the Second Symposium on Assessment of Quality Software Development Tools. – 1992. – P. 2–10.

106. Kubat P. Assessing reliability of modular software / P. Kubat // Oper. Res. Lett. 8. –1989. – P. 35–41.

107. Wen-Li Wang. An Architecture-Based Software Reliability Model / Wang Wen-Li, Wu Ye, Chen Mei-Hwa // Proceedings of the 1999 Pacific Rim International Symposium on Dependable Computing. – 1999. – P. 143-150.

108. Laprie J.C. Dependability evaluation of software systems in operation / J.C. Laprie // IEEE Trans. Software Eng. – 1984. – Vol.10, №6. – P. 701–714.

109. Laprie J.C. X-ware reliability and availability modeling / J.C. Laprie, K. Kanoun // IEEE Trans. Software Eng. – 1992. – Vol.18, №10. – P.130–147.

110. Dolbec J. A Component Based Software Reliability Model / J. Dolbec, T. Shepard // Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research. – 1995. – P. 19.

111. Goševa-Popstojanova K. Architecture-Based Software Reliability: Why Only a Few Parameters Matter? / K. Goševa -Popstojanova, M. Hamill // Computer Software and Applications Conference of 31st Annual International. – 2007. – P. 423–430.

112. Yong Guo. Reliability Evaluation Optimal Selection Model of Component-Based System / Yong Guo, Tian Tian Wan, Pei Jun Ma, Xiao Hong Su // Journal of Software Engineering and Applications. – 2011. – Vol. 4. – P. 433–441.

113. Jehad Al Dallal. The impact of accounting for special methods in the measurement of object-oriented class cohesion on refactoring and fault prediction activities / Jehad Al Dallal // The Journal of Systems and Software. – 2012. – Vol. 85. – P. 1042–1057.

114. Ampatzoglou A. A methodology to assess the impact of design patterns on software quality / A. Ampatzoglou, G. Frantzeskou, I. Stamelos // Information and Software Technology. – 2012. – Vol. 54. – P. 331–346.

115. Абдул-Хади А.М. Разработка базовых марковских моделей для исследования готовности коммерческих веб-сервисов / А.М. Абдул-Хади, Ю.Л. Поночовный, В.С. Харченко // Радіоелектронні і комп'ютерні системи. – Харків: НАКУ «ХАИ». – 2013. – № 5(64). – С. 186-191.

116. Bochmann G. Improved Usage Model for Web Application Reliability Testing / Gregor v. Bochmann, Guy-Vincent Jourdan, and Bo Wan // B. Wolff and F. Zaidi (Eds.): ICTSS 2011, LNCS 7019, pp. 15–31, (2011).

117. Pham H. Software Reliability / H. Pham // Wiley Encyclopedia of Electrical and Electronic Engineering. – 1999. – P.565–578.

118. Zhang X. A software cost model with error removal times and risk costs / X. Zhang, H. Pham // International Journal of Systems Science. – 1998. – Vol. 29, No 4. – P. 435–442.

119. Zhang X. A software cost model with warranty cost, error removal times and risk costs / X. Zhang, H. Pham // IIE Transactions on Quality and Reliability Engineering. – 1998. – Vol. 30, No 12. – P. 1135–1142.

120. Pham H. NHPP software reliability and cost models with testing coverage / H. Pham, X. Zhang // European Journal of Operational Research. – 2003. – Vol. 145. – P. 443–454.

121. Ehrlich W. Determining the cost of a stop-testing decision / W. Ehrlich, B. Prasanna, J. Stampfel, J. Wu // IEEE Software. – 1993. – P. 33–42.

122. Pham H. A software cost model with warranty and risk costs / H. Pham, X. Zhang // IEEE Trans on Computers. – 1999. – Vol. 48, No. 1. – P.71–75.

123. Pham H. A software cost model with imperfect debugging, random life cycle and penalty cost / H. Pham // International Journal of Systems Science. – 1996. – Vol. 27, No. 5. – P. 455–463.

124. Teng X. A software cost model for quantifying the gain with considerations of random field environments / X. Teng, H. Pham // IEEE Transactions on Computers. – 2004. – P. 380–384.

125. Яковина В.С. Моделі, методи та засоби аналізу надійності програмних систем : монографія / Яковина В.С., Федасюк Д.В., Сенів М.М., Нитребич О.О. – Львів : Видавництво Львівської політехніки, 2015. – 220 с.

126. Gokhale S. Analysis of software fault removal policies using a non homogeneous continuous time Markov chain / S. Gokhale, M.R. Lyu, K.S. Trivedi // Software Quality Journal. – 2004. – Vol. 12, No 3. – P. 211–230.

127. Jeske D. R. Some Successful Approaches to Software Reliability Modeling in Industry / D. R. Jeske, X. Zhang // Journal of Systems and Software. – 2005. – Vol. 74. – P. 85–99.

128. Яковина В.С. Критерій достатності процесу тестування програмного забезпечення / Яковина В.С., Сенів М.М., Чабанюк Я.М., Федасюк Д.В., Хімка У.Т. // Вісник Національного університету "Львівська політехніка". Комп'ютерні науки та інформаційні технології. – 2010. – № 672. – С. 346–358.

129. Сенів М.М. Аналіз використання моделі надійності програмного забезпечення з динамічним показником складності проекту протягом життєвого циклу / Сенів М.М., Федасюк Д.В., Чабанюк Я.М., Яковина В.С. // Комп’ютерні технології друкарства. – 2010. – № 24. – С. 107–122.

130. Сенів М.М. Метод оцінювання та прогнозування надійності програмного забезпечення на основі моделі з динамічним показником величини проекту / М. Сенів, В. Яковина, Я. Чабанюк, Д. Федасюк // Комп'ютинг. – 2011. – Том 10, Випуск 2. – С. 97–107.

131. Яковина В.С. Оцінювання та прогнозування надійності програмного забезпечення на основі моделі з індексом складності проекту / В.С. Яковина, Я.М. Чабанюк, М.М. Сенів, У.Т. Хімка // Вісник Хмельницького національного університету. Серія : Технічні науки. – 2011. – № 2 (174). – С. 149–157.

132. Яковина В.С. До питання про визначення діапазонів значень індексу складності програмного засобу / В.С. Яковина, В.І. Коцун // ІІ всеукраїнська науково-практична конференція "Сучасні інформаційні технології в економіці, менеджменті та освіті" СІТЕМ-2011, Львів, 18 листопада 2011 : матеріали. – С. 140–143.

133. Яковина В.С. Моделювання параметра потоку відмов програмного забезпечення та визначення діапазонів показника його складності / В.С. Яковина // Вісник Національного університету "Львівська політехніка". Комп'ютерні системи та мережі. – 2014. – № 806. – С. 296–302.

134. Volochiy B.Yu. Reliability estimation of embedded systems witch software is presented by piecewise linear function / B. Volochiy, L. Ozirkovskyi, M. Miskiv, V. Yakovyna, O. Mulyak // XIIth International Conference Modern Problems of Radio Engineering, Telecommunications and Computer Science TCSET'2014, Lviv–Slavske, February 25 – March 1, 2014 : Proceedings. – P. 255–257.

135. Муляк О.В. Вплив моделей надійності програмних засобів на показники надійності програмно-апаратних систем / О.В. Муляк, В.С. Яковина, Б.Ю. Волочій // Східно-Європейський журнал передових технологій. – 2015. – Том 4, № 9(76). – С. 53–57.

136. Сенів М.М. Засоби прогнозування надійності програмного проекту із врахуванням показника його складності [Текст] : дис. … канд. техн. наук : 01.05.03 : захищена 09.06.2011 : затв. 17.02.2012 / Сенів Максим Михайлович ; Нац. ун-т "Львівська політехніка". – Львів, 2011. – 173 с.

137. Анцыпов А.В. Оценка влияния качества документации на надежность программных средств / А.В. Анцыпов // Технологии информатизации и управления. – 2009. – С. 332–337.

138. Goel A.L. A guidebook for software reliability assessment / A. L. Goel. – Rep. RADCTR-83-176, Aug. – 1983.

139. Viktorov O. Reconfigurable Multiprocessor System Reliability Esimation / Oleg Viktorov // Asian Jounal of Information Technology. – 2007. – No 6 (9). – P. 958–960.

140. Белый Ю.А. Модели отказов и оценка надежности мультидиверсных систем / Ю.А. Белый // Радіоелектронні і комп'ютерні системи. – 2008. – № 5 (32). – С. 62–66.

141. Можаев, А. С. Общий логико-вероятностный метод анализа надежности сложных систем [Текст] / А. С. Можаев. – Л.: ВМА, 1988. – 68 с.

142. Рябинин, И. А. Надежность и безопасность сложных систем [Текст] / И. А. Рябинин. – СПб.: Политехника, 2000. – 248 с.

143. Томашевський, В. М. Моделювання систем [Текст] / В. М. Томашевський. – К.: Видавнича група BHV, 2005. – 352 с.

144. Советов, Б. Я. Моделирование систем [Текст] / Б. Я. Советов, С. А. Яковлев. – М.: Высшая школа, 2001. – 343 с.

145. Volochiy B.Yu. Automated Development of Markovian Chains for Fault-Tolerant Computer-Based Systems with Version-Structure Redundancy [Електронний ресурс] / Bogdan Volochiy, Oleksandr Mulyak, Vyacheslav Kharchenko // 11th International Conference on ICT in Education, Research and Industrial Applications: Integration, Harmonization and Knowledge Transfer (ICTERI 2015) : Proceedings. – Lviv, 14–16 May 2015. – P. 462–475. – Режим доступу : http://ceur-ws.org/Vol-1356/paper_82.pdf

146. Волочий Б.Ю. Надежностная модель отказоустойчивой многопроцессорной системы с учетом двукратного обновления программного обеспечение / Б.Ю. Волочий, М.В. Миськив, А.В. Муляк, Л.Д. Озирковский // Надежность и качество. Труды международного симпозиума, Т. 1 / Под ред. Н.К. Юркова. – Россия, Пенза: Изд-во Пенз. гос. ун-та, 2013. – С. 128–133.

147. Волочій Б.Ю. Надійнісна модель відмовостійкої багатопроцесорної систем з відновленням працездатності програмного забезпечення / Б.Ю. Волочій, Л.Д. Озірковський, О.В. Муляк, М.М. Змисний, Т.І. Панський // Вісник НТУУ «КПІ». Серія Радіотехніка. Радіоапаратобудування. – 2013. – № 54. – С. 33 – 43.

148. Волочій Б.Ю. Визначення впливу оновлення програмного забезпечення на показники надійності відмовостійкої багатопроцесорної системи / Б.Ю.

Волочій, М. В. Міських, О.В. Муляк, Л.Д. Озірковський // Східноєвропейський журнал передових технологій. – 2013. – №9 (63). – С. 55–59.

149. Мандзій Б.А. Дослідження впливу двократного оновлення програмного забезпечення на показники надійності відмовостійкої програмно-апаратної системи / Б.А. Мандзій, Б.Ю. Волочій, М.В. Міських, О.В. Муляк // ІІІ міжнародна науково-практична конференція "Фізико-технологічні проблеми радіотехнічних пристроїв, засобів телекомунікацій, нано- та мікроелектроніки" : Матеріали. – Чернівці, 2013. – С. 72–73.

150. Яковина В.С. Використання інформаційного критерію Акаїке в задачах моделювання надійності програмного забезпечення / В. Яковина, О. Нитребич, Д. Федасюк // Вісник Національного університету "Львівська політехніка". Комп'ютерні науки та інформаційні технології. – 2012. – № 732. – С. 190–192.

151. Yakovyna V.S. On using AIC in software reliability modeling / V. Yakovyna, D. Fedasyuk, O. Nytrebych // VIIIth International Conference on Perspective Technologies and Methods in MEMS Design MEMSTECH'2012, Polyana, April 18–21, 2012 : Proceedings. – P. 93–94.

152. Yakovyna V.S. Modified high-order software reliability Gokhale model / V. Yakovyna, O. Nytrebych, D. Fedasyuk // VIIth International Scientific and Technical Conference "Computer Science and Information Technologies" CSIT'2012, Lviv, November 20–24, 2012 : Proceedings. – P. 194–195.

153. Yakovyna V.S. Evaluation matrix transition probabilities between software components based on UML use case diagram / V. Yakovyna, Iu. Parfeniuk // VIIth International Scientific and Technical Conference "Computer Science and Information Technologies" CSIT'2012, Lviv, November 20–24, 2012 : Proceedings. – P. 196–197.

154. Яковина В.С. Використання марковських ланцюгів вищого порядку в задачах моделювання надійності програмного забезпечення / Яковина В., Сердюк П., Нитребич О., Федасюк Д. // Вісник Національного університету

"Львівська політехніка". Комп'ютерні науки та інформаційні технології. – 2013. – № 771. – С. 209–213.

155. Яковина В.С. Використання засобів UML для прогнозування надійності програмного забезпечення на етапі його проектування / Яковина В.С., Парфенюк Ю.І. // Вісник Національного університету "Львівська політехніка". Комп'ютерні системи та мережі. – 2013. – № 773. – С. 151–156.

156. Yakovyna V.S. Determination of transition probabilities between software components, written in Java, based on monitoring of its execution / V. Yakovyna, Iu. Parfeniuk // 12th International Conference "The Experience of Designing and Application of CAD Systems in Microelectronics" CADSM 2013, Lviv–Polyana, February 19–23, 2013 : Proceedings. – P. 382–383.

157. Yakovyna V.S. Markov chain dynamic representation model for reliability testing / Yakovyna V., Serdyuk P., Nytrebych O. // 12th International Conference "The Experience of Designing and Application of CAD Systems in Microelectronics" CADSM 2013, Lviv–Polyana, February 19–23, 2013 : Proceedings. – P. 384–385.

158. Yakovyna V.S. Determination of transition probabilities between software components on the design stage / V. Yakovyna, Iu. Parfeniuk // IXth International Conference on Perspective Technologies and Methods in MEMS Design MEMSTECH 2013, Polyana, April 16–20, 2013 : Proceedings. – P. 156–157.

159. Yakovyna V.S. The representation of high order Markov process through equivalent first order process [Електронний ресурс] / V. Yakovyna, O. Nytrebych, D. Fedasyuk // 6th International Academic Conference of Young Scientists "Computer Science and Engineering 2013" (CSE-2013), Lviv, 21–23 November, 2013 : Proceedings. – P. 216–217. – CD-ROM. – Заг. з етикетки диска.

160. Яковина В.С. Аналіз використання інформаційних критеріїв у моделях оцінки надійності програмного забезпечення / В.С. Яковина, Д.В. Федасюк, О.О. Нитребич // Вісник Національного технічного університету "ХПІ". Серія: Нові рішення в сучасних технологіях. – 2014. – № 26 (1069). – С. 108–115.

161. Yakovyna V.S. Software reliability assessment using high-order Markov chains / V. Yakovyna, D. Fedasyuk, O. Nytrebych, Iu. Parfenyuk, V. Matselyukh // International Journal of Engineering Science Invention. – 2014. – Vol. 3, Issue 7. – P. 1–6.

162. Yakovyna V.S. Introduction to reliability development lifecycle / V. Yakovyna // Modern Problems of Radio Engineering, Telecommunications and Computer Science : XIIth International Conference TCSET'2014, Lviv–Slavske, February 25 – March 1, 2014 : Proceedings. – P. 192–194.

163. Yakovyna V.S. The use of modern software development tools to analyze its reliability / V. Yakovyna, Iu. Parfeniuk // Xth International Conference on Perspective Technologies and Methods in MEMS Design MEMSTECH 2014, Lviv, June 22–24, 2014 : Proceedings. – P. 112–114.

164. Yakovyna V.S. The model for software reliability estimation using high-order continuous-time Markov chains / V. Yakovyna, V. Masyukevych // IXth International Scientific and Technical Conference "Computer Science and Information Technologies" CSIT'2014, Lviv, November 18–22, 2014 : Proceedings. – P. 83–86.

165. Яковина В.С. Компонентні моделі надійності програмного забезпечення вищого порядку / В.С. Яковина // Електротехнічні та комп'ютерні системи. – 2015. – № 19 (95). – С. 252–255.

166. Яковина В.С. Метод аналізу надійності програмних засобів з урахуванням їх складності / В.С. Яковина // Радіоелектронні і комп'ютерні системи. – 2015. – № 2 (72). – С. 127–133.

167. Yakovyna V.S. Discrete and continuous time high-order Markov models for software reliability assessment [Електронний ресурс] / V. Yakovyna, O. Nytrebych // 11th International Conference on ICT in Education, Research and Industrial Applications: Integration, Harmonization and Knowledge Transfer (ICTERI 2015), Lviv, 14–16 May 2015 : Proceedings. – P. 419–431. – Режим доступу : http://ceur-ws.org/Vol-1356/paper_62.pdf

168. Burkhart W. Testing Software and Systems / W. Burkhart, Z. Fatiha // 23rd Ifip Wg 6.1 International Conference. – 2011. – P. 236.

169. Takagi T. Accurate Usage Model Construction Using High-Order Markov Chains / T. Takagi, Z. Furukawa, T. Yamasaki // Supplementary Proceedings of 17th International Symposium on Software Reliability Engineering. – 2006. – P. 1-2.

170. Нитребич О.О. Моделі та методи оцінювання надійності програмного забезпечення з урахуванням його архітектури [Текст] : дис. … канд. техн. наук : 01.05.03 : захищена 19.03.15 : затв. 30.06.15 / Нитребич Оксана Олександрівна ; Нац. ун-т "Львівська політехніка". – Львів, 2015. – 153 с.

171. Wood A. Predicting software reliability / A. Wood // Computer. – 1996. – Vol. 29 (11). – P. 69–77.

172. Федасюк Д.В. Метод побудови сценаріїв тестування програмного забезпечення на основі аналізу його змінних / Федасюк Д.В., Яковина В.С., Сердюк П.В., Нитребич О.О. // Інформаційні технології та комп'ютерна інженерія. – 2014. – № 2 (30). – С. 50–58.

173. Fedasyuk D.V. Algorithm for automated test scenario construction / D. Fedasyuk, V. Yakovyna, P. Serdyuk, O. Nytrebych // Xth International Conference on Perspective Technologies and Methods in MEMS Design MEMSTECH 2014, Lviv, June 22–24, 2014 : Proceedings. – P. 59–62.

174. Tobon-Mejia D. Hidden Markov models for failure diagnostic and prognostic / Diego Tobon-Mejia, Kamal Medjaher, Noureddine Zerhouni, G´erard Tripot // Prognostics and System Health Management Conference. – 2011. – P. 1– 8.

175. Yi Xie. Modeling Oscillation Behavior of Network Traffic by Nested Hidden Markov Model with Variable State-Duration / Yi Xie, Jiankun Hu, Yang Xiang, Shui Yu, Shensheng Tang, Yu Wang // Parallel and Distributed Systems, IEEE Transactions. – 2012. – Vol.24, Issue 9. – P. 1807–1817

176. Gales M. The Application of Hidden Markov Models in Speech Recognition / Mark Gales, Steve Young // Foundations and Trends in Signal Processing. – 2008. – Vol. 1, No 3. – P. 195–304

177. Schliep A. Using hidden Markov models to analyze gene expression time course data / A. Schliep, A. Schonhuth, C. Steinhoff // Bioinformatics. – 2003. – Vol. 19. – P. i255-i263.

178. Event-based Failure Prediction: An Extended Hidden Markov Model Approach. [Електронний ресурс] : Режим доступу https://informatik.hu-berlin.de/Members/salfner/publications/salfner08eventbased.pdf.

179. Derivation of Baum-Welch Algorithm for Hidden Markov Models. [Електронний ресурс] : Режим доступу http://people.csail.mit.edu/stephentu/writeups/hmm-baum-welch-derivation.pdf.

180. Липаев В. В. Документирование и управление конфигурацией программных средств / В. В. Липаев. – М. : СИНТЕГ, 1998. – 372 с.

181. Острейковский В. А. Теория надежности / В. А. Острейковский. – М. : Высш. школа, 2003. – 363 с.

182. ДСТУ 30004-95. Надійність техніки. Методи оцінки показників надійності за експериментальними даними. – К. : Держстандарт України, 1995. – 42 с.

183. Markov Models, Hidden and Otherwise. [Електронний ресурс] : Режим доступу http://kochanski.org/gpk/teaching/0401Oxford/HMM.pdf

184. Berchtold A. The mixture transition distribution model for high-order Markov chains and non-Gaussian time series / André Berchtold, Adrian E. Raftery // Statistical Science. – 2002. – Vol. 17, No 3. – P. 328–356.

185. Shamshad A. First and second order Markov chain models for synthetic generation of wind speed time series / A. Shamshad, M.A. Bawadi, W.M.A. Wan Hussin, T.A. Majid // S.A.M. Sanusi Energy. – 2005. – Vol. 30, Issue 5. – P. 693–708.

186. Волочій Б. Ю. Формалізація побудови моделей дискретно-неперервних стохастичних систем з використанням методу фаз Ерланга / Б. Ю. Волочій, Л. Д. Озірковський, І. В. Кулик // Відбір і оброб. інформації : міжвід. зб. наук. пр. – 2012. – Вип. 36. – С. 39–47.

187. Stepashko V. Methods and criteria for solving structural identification / V. Stepashko, Y. Kocherha // Automatics. – 1985. – vol. 5. – P. 29-37.

188. Burnham P. Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach / P. Burnham, D. Anderson – Springer, 2002. – 488 p.

189. Mallows C. Some comments on Cp / C. Mallows // Technometrics. – 1973. – vol. 15. – P. 661-667.

190. Liu T. Application of Markov Chains to Analyze and Predict the Time Series / T. Liu // Modern Applied Science. – 2010. – P. 162–166.

191. Aho K. Model selection for ecologists: the worldviews of AIC and BIC / K. Aho, D. Derryberry, T. Peterson // Ecology, Mar;95(3):631-6, 2014.

192. Akaike H. A new look at the statistical model identification / H. Akaike // IEEE Trans. Auto. Control. – 1974. – P. 716-723.

193. Liew K. The Performance of AICc as an Order Selection Criterion in ARMA Time Series Models / K. Liew, S. Mahendran // Pertanika J. Sci. & Technol. – 2002. – P. 25–33.

194. Burnham K.P. Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach Hardcover / Kenneth P. Burnham, David R. Anderson. – Springer Science & Business Media. – 2002. – P. 488.

195. Schwarz G. Estimating the Dimension of a Model / G. Schwarz // Annals of Statistics. – 1978. – P. 461–64.

196. Cortellessa V. Early reliability assessment of UML based software models / V. Cortellessa, H. Singh, B. Cukic // In Proceedings of the 3rd international workshop on Software and performance. – 2002. – P. 302–309.

197. Катренко А. В. Системний аналіз об’єктів та процесів комп’ютеризації / А. В. Катренко. – Львів: Новий світ, 2000. – 424 с.

198. Fedasyuk D.V. Variable state-based software usage-model based on its variables / D. Fedasyuk, V. Yakovyna, P. Serdyuk, O. Nytrebych // Econtechmod. – 2014. – Vol. 3, No 2. – P. 15–20.

199. Яковина В.С. Удосконалена процедура визначення кількості дефектів програмного продукту на ранніх етапах тестування / Яковина В.С., Сенів М.М., Гаранджа І.Я. // Міжнародна наукова конференція

"Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту" ISDMCI'2012, Євпаторія, 27–31 травня 2012 : збірка наукових праць. – С. 238.

200. Свідоцтво про реєстрацію авторського права на твір. Україна. Система оцінювання надійності програмного забезпечення / В.С. Яковина, О.О. Нитребич, П.В. Сердюк. – № 58985 ; заявл. 26.01.15 ; зареєстр. 24.03.15.

201. Яковина В.С. Дослідження характеристик криптостійкості алгоритму симетричного шифрування DES / Яковина В.С., Федасюк Д.В., Салій С.І., Сенів М.М. // Вісник Національного університету "Львівська політехніка". Комп'ютерні системи проектування. Теорія і практика. – 2008. – № 626. – С. 55–62.

202. Яковина В.С. Дослідження основних характеристик алгоритму симетричного шифрування RC5 для побудови модуля захисту розподіленої системи теплового проектування / Яковина В.С., Одуха О.В., Сенів М.М., Білас О.Є. // Вісник Національного університету "Львівська політехніка". Комп'ютерні науки та інформаційні технології. – 2008. – № 616. – С. 143–150.

203. Яковина В.С. Аспектна декомпозиція компонентів захисту розподіленої системи теплового проектування / В. Яковина, Н. Мамроха, М. Сенів // Шоста міжнародна конференція "Інтернет – Освіта – Наука – 2008"ІОН-2008, Вінниця, 7–11 жовтня 2008 : збірник матеріалів. – Т. 2. – С. 407–410.

204. Яковина В.С. Порівняння швидкодії програмної реалізації алгоритмів симетричного (DES) та асиметричного (RSA) шифрування / Яковина В., Федасюк Д., Сенів М., Білас О. // Вісник Національного університету "Львівська політехніка". Комп'ютерні науки та інформаційні технології. – 2007. – № 598. – С. 181–185.

205. Яковина В.С. Аналіз використання аспектно-орієнтованого програмування як засобу підвищення надійності програмного забезпечення / В.С. Яковина, Д.В. Федасюк, Н.М. Мамроха // Інженерія програмного забезпечення. – 2010. – № 2. – С. 24–29.

206. Сенів М.М. Розробка методу оцінювання і прогнозування надійності програмного забезпечення з врахуванням критерію достатності процесу тестування / Сенів М.М., Федасюк Д.В., Яковина В.С. // Комп'ютерні технології друкарства. – 2011. – № 25. – С. 72–83.

207. Яковина В.С. Удосконалена процедура прогнозування відмов програмних засобів на основі моделі надійності з індексом складності / Яковина В.С., Федасюк Д.В. // Інженерія програмного забезпечення. – 2015. – № 2 (22). – С. 5–13.

208. Broy M. Model Based Testing of Reactive Systems / M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, A. Pretschner. – LNCS 3472, Springer. – 2005. – 659 p.

209. Blackburn M. Why Model-Based Test Automation is Different and What You Should Know to Get Started / M. Blackburn, R. Busser, A. Nauman // In International Conference on Practical Software Quality. – 2004. – P. 87-90

210. Legeard B. Controlling Test Case Explosion in Test Generation from B Formal Models / B. Legeard, F. Peureux, M. Utting // The Journal of Software Testing, Verification and Reliability. – 2004. – № 14(2). – P. 81–103.

211. Shamsoddin-Motlagh E. A Review of Automatic Test Cases Generation / E. Shamsoddin-Motlagh // International Journal of Computer Applications. – 2012. – № 57(13). – P. 25–29.

212. Jerry Gao. A Review of Model-Based Software Testing and Automation Tools [Електронний ресурс] : Режим доступу http://www.asq-silicon-valley.org/document-for-members/doc_download/117-3-jerry-gao-model-based-software-testing.

213. Belli F. A Holistic Approach to Testing of Interactive Systems using Statecharts / F. Belli, C.J. Budnik, A. Hollman // Proceedings of 2nd South-East European Workshop on Formal Methods (SEEFM 05), South-Eastern European Research Center SEERC. – 2005. – P. 1–15.

214. Hong H. Automatic Test Generation from Statecharts Using Model Checking / H. Hong, Insup Lee, O. Sokolsky, Sung Deok Cha // In Proceedings of FATES'01, Workshop on Formal Approaches to Testing of Software. – 2001. – P.15–30.

215. Hu Y.T. Automatic Black-Box Method-Level Test Case Generation Based on Constraint Logic Programming. / Y.T. Hu, N.W. Lin // Computer Symposium (ICS). – 2010. – P. 977–982.

216. Samuel P. Automatic test case generation using unified modeling language (UML) state diagrams / P. Samuel, R. Mall, A. Bothra // The Institution of Engineering and Technology, IET Softw. – 2008. – № 2. – P. 79–93.

217. Sarma M. Automatic Test Case Generation from UML Models / M. Sarma, R. Mall // 10th International Conference on Information Technology. – 2007. – P. 196–201.

218. Sarma M. Automatic Test Case Generation from UML Sequence Diagrams / M. Sarma, D. Kundu, R. Mall // 15th International Conference on Advanced Computing and Communications. – 2007. – P. 60–65.

219. Santiago V. An Environment for Automated Test Case Generation from Statechart-based and Finite State Machine-based Behavioral Models / V. Santiago, N. Vijaykumar, D. Guimaraes // IEEE International Conference on Software Testing Verification and Validation Workshop (ICSTW'08). – 2008. – P.63–72.

220. Susumu Fujiwara. Test Selection Based on Finite State Models / Susumu Fujiwara, Gregor Bochmann, Ferhat Khendek, Mokhtar Amalou, Abderrazak Ghedamsi // IEEE Transactions on Software Engineering. – 1991. – № 6(17). – P. 591–603.

221. Apfelbaum L. Model-based testing / L. Apfelbaum, J. Doyle. // Proceedings of the 10th International Software Quality Week. – 1997.

222. Clarke J. Automated test generation from a behavioral model / J. Clarke // Proceedings of the 11th International Software Quality Week . – 1998.

223. Robinson H. Finite state model-based testing on a shoestring / H. Robinson // Proceedings of the 1999 International Conference on Software Testing Analysis and Review. – 1999.

224. Whittaker J. A. Markov Chain Model for Statistical Software Testing / J. A. Whittaker, M. G. Thomason // Software Engineering, IEEE Transactions. – 1994. – P. 812–824.

225. Winfried Dulz. Statistical Usage Testing by Annotated Sequence Diagrams, Markov Chains and TTCN-3 / Winfried Dulz, Fenhua Zhen // Third International Conference On Quality Software. – 2003. – P. 336–342.

226. Siegl S. Model-Driven Testing based on Markov Chain Usage Models in the Automotive Domain / S. Siegl, D. Winfried, G. Reinhard, K. Gerhard // Proc. of the 12th European Workshop on Dependable Computing. – 2009.

227. Guen H. Model-Based Testing Automatic Generation of Test Cases Using the Markov Chain Model / L. H. Guen, F. Vallée, A. Faucogney. – Jean-Louis Boulanger, Consultant. – 2012. – 320 p.

228. Pretschner A. One Evaluation of Model-Based Testing and Its Automation / A. Pretschner, W. Prenninger, S. Wagner, C. Kühnel, M. Baumgartner, B. Sostawa, R. Zölch, T. Stauner // Proceedings of the 27th International Conference on Software Engineering. – 2005. – P. 392–401.

229. Blackburn M. Model-Based Approach to Security Test Automation / M. Blackburn, R. Busser, A. Nauman, R. Chandramouli // Proceedings of Quality Week. – 2001.

230. El-Far I. K. Model-based software testing / I. K. El-Far, J.A. Whittaker // Encyclopedia of Software Engineering. – 2002. – № 1. – P. 825–837.

231. Hartma A. The AGEDIS tools for model based testing / A. Hartma, K. Nagin // Proceedings of the 2004 ACM SIGSOFT International Symposium on Software Testing and Analysis. – 2004. – P. 129-132.

232. Jungmayr S., Stumpe J. Another motivation for usage models: generation of user documentation / S. Jungmayr, J. Stumpe // Proceedings of CONQUEST'98, Nüremberg, Germany. – 1998.

233. Herbold S. A Model for Usage-based Testing of Event-driven Software / S. Herbold, J. Grabowski, S. Waack // 3rd International Workshop on Model-Based Verification & Validation From Research to Practice , IEEE Computer Society. – 2011. – P. 172–178.

234. Heiko Koziolek. Operational Profiles for Software Reliability / Heiko Koziolek // Dependability Engineering, volume 2 of Trustworthy Software Systems. – 2006. – P. 119–142.

235. Bieman J. M. Cohesion and Reuse in an Object-Oriented System / J. M. Bieman, B-K. Kang // Proc. ACM Symposium on Software Reusability (SSR'95). – 1995. – P. 259–262.

236. Ott L. Developing Measures of Class Cohesion for Object-Oriented Software / L. Ott, J. M. Bieman, B-K. Kang, B. Mehra // Proc. Annual Oregon Workshop on Software Merics. – 1995. – P. 11.

237. Изосимов А.В. Метрическая оценка качества программ / А.В. Изосимов, А.Л. Рыжко. – М.: Изд-во МАИ. – 1989. – 96 с.

238. Chapin N. An Entropy Metric For Software Maintainability System Sciences / N. Chapin // Proceedings of the Twenty-Second Annual Hawaii International Conference, Volume II: Software Track. – 1989. – P. 522–523.

239. Холстед М.Х. Начала науки о программах / М.Х. Холстед. – М.: Финансы и статистика, 1981. – 128 с., ил.

240. McConnell S. Software quality at top speed / S. McConnell // Software Development. – 1996. – Vol. 4(8). – P. 38–42.

241. Barry Boehm. Equity Keynote Address. – 2007.

242. IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology. – 1990. – 84 P.

243. Broy M. What Characterizes a (Software) Component? / Manfred Broy, Anton Deimel, Juergen Henn, Kai Koskimies, Frantisek Plasil, Gustav Pomberger, Wolfgang Pree, Michael Stal, Clemens Szyperski // Journal of Software – Concepts and Tools. – 1998. – Vol. 19, No 1. – P. 49–56.

244. Brown A. W. The Current State of CBSE / Alan W. Brown, Kurt C. Wallnau // IEEE Software. – 1998. – Vol. 15, No 5. – P. 37–46.

245. Crnkovic I. Building Reliable Component-Based Software Systems / Ivica Crnkovic, Magnus Larsson. – Artech House Inc. – 2002. – 399 p.

246. Jerry Zeyu Gao. Testing and Quality Assurance for Component-Based Software / Jerry Zeyu Gao, H.-S. Jacob Tsao, Ye Wu. – Artech House Inc. – 2003. – 439 p.

247. Heineman G. T. Component-Based Software Engineering: putting the pieces together / G. T. Heineman, W. T. Councill. – Addison-Wesley. – 2001. – 818 p.

248. Clemens Szyperski. Component Software: Beyond Object-Oriented Programming / Clemens Szyperski. – ACM Press/Addison-Wesley. – 2002. – 531 p.

249. Mohd Ehmer Khan. Different Forms of Software Testing Techniques for Finding Errors / Mohd Ehmer Khan // International Journal of Computer Science Issues. – 2010. – № 7(3). – P. 11-16.

250. Shatnawi O. Discrete Time NHPP Models for Software reliability growth phenomenon / O. Shatnawi // The International Arab Journal of Information technology. – 2010. – № 6 (2). – P. 124-131.

251. Emam K. A methodology for validating software product metrics / El K. Emam // Tech. rep. NCR/ERC-1076, National Research Council of Canada, Ottawa, Ontario, Canada. – 2000.

252. Fedasyuk D.V. Information model of data representation for software reliability estimation framework / Fedasyuk D.V., Seniv M.M., Yakovyna V.S., Mamrokha N.M. // Xth International Conference "The Experience of Designing and Application of CAD Systems in Microelectronics" CADSM 2009, Lviv–Polyana, February 24–28, 2009 : Proceedings. – P. 287–291.

253. Seniv M.M. Architecture of software system for the software reliability evaluation and prediction / M. Seniv, D. Fedasyuk, Yu. Parfenyuk, V. Yakovyna, Ya. Chabanyuk // Vth International Scientific and Technical Conference "Computer Science and Information Technologies" CSIT 2010, Lviv, October 14–16, 2010 : Proceedings. – P. 164–167.

254. Сенів М.М. Система оцінювання та прогнозування надійності програмного забезпечення / Сенів М.М., Федасюк Д.В., Парфенюк Ю.І., Яковина В.С., Чабанюк Я.М. // Відбір і обробка інформації. – 2010. – № 33 (109). – С. 123–129.

255. Seniv M.M. Dataware and software of client-server system for software reliability indexes evaluation / Seniv M.M., Yakovyna V.S., Parfenyuk Y.I., Lobutska O.I., Noha Y.M. // XIth International Conference "The Experience of Designing and Application of CAD Systems in Microelectronics" CADSM 2011, Lviv–Polyana, February 23–25, 2011 : Proceedings. – P. 324–326.

256. Yakovyna V.S. On the possibility of software reliability prediction using RBF neural network / V. Yakovyna, O. Synytska, T. Kremen, V. Smirnov // VIth International Scientific and Technical Conference "Computer Science and Information Technologies" CSIT 2011, Lviv, November 16–19, 2011 : Proceedings. – P. 39–42.

257. Кремень Т.І. Використання НМ Елмана для прогнозування надійності програмного забезпечення / Т. Кремень, В. Яковина, О. Синицька // V міжнародна конференція молодих вчених "Комп'ютерні науки та інженерія" CSE-2011, Львів, 24–26 листопада 2011 : матеріали. – С. 88–89.

258. Yakovyna V.S. Influence of the activation function of RBF neural network on software reliability time series prediction / V. Yakovyna, V. Ketsman // VIIIth International Conference on Perspective Technologies and Methods in MEMS Design MEMSTECH'2012, Polyana, April 18–21, 2012 : Proceedings. – P. 91–92.

259. Яковина В.С. Про ефективність НМ Елмана для прогнозування відмов програмного забезпечення / В.С. Яковина, В.Д. Кецман // III всеукраїнська науково-практична конференція "Сучасні інформаційні технології в економіці, менеджменті та освіті" CITEM-2012, Львів, 21 листопада 2012 : матеріали. – С. 223–227.

260. Яковина В.С. Вдосконалений протокол взаємної аутентифікації на основі смарт-карт для побудови модуля захисту розподіленої системи теплового проектування / Яковина В.С., Одуха О.В. // Вісник Національного університету "Львівська політехніка". Комп'ютерні науки та інформаційні технології. – 2009. – № 650. – С. 214–221.

261. Яковина В.С. Вплив кількості вхідних нейронів мережі RBF з функцією активації Гаусса на ефективність прогнозування відмов програмного

забезпечення / Яковина В.С., Гаранджа І.Я. // Міжнародна наукова конференція "Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту" ISDMCI'2013, Євпаторія, 20–24 травня 2013 : збірка наукових праць. – С. 520–522.

262. Yakovyna V.S. Influence of RBF neural network input layer parameters on software reliability prediction / V. Yakovyna // 4-th International Conference on Inductive Modelling ICIM'2013, Kyiv, 16–20 September 2013 : Proceedings. – P. 344–347.

263. Яковина В.С. Прогнозування відмов програмного забезпечення з використанням НМ на основі радіально-базисних функцій / В.С. Яковина // Вісник Національного університету "Львівська політехніка". Інформаційні системи та мережі. – 2014. – № 805. – С. 230–236.

264. Yakovyna V.S. Software failures prediction using RBF neural network / V.S. Yakovyna // Праці Одеського політехнічного університету. – 2015. – вип. 2(46). – С. 111–118.

265. Encog Machine Learning Framework [Електронний ресурс]. – Режим доступу: http://www.heatonresearch.com/encog. – Назва з екрану.

266. Khoshgoftaar, T.M., Predicting software quality, during testing, using neural network models: A comparative study / T.M. Khoshgoftaar, R.M. Szabo // International Journal of Reliability, Quality and Safety Engineering. – 1994. – Vol. 1. – P. 303–319.

267. Zheng, J. Predicting software reliability with neural network ensembles / J. Zheng // Expert Systems with Applications. – 2009. – Vol. 36. – P. 2116-2122.

268. НейроПроект. Аналитические технологи для прогнозирования и анализа данных [Електронний ресурс] :- книга / Компания «НейроПроект» – Електорон. дан. (1 файл). – 2005. – 1с. – Режим доступу: http://www.neuroproject.ru/forecasting_tutorial.php. – Назва з екрана.

269. Paliwal, M., Neural networks and statistical techniques: A review of applications / M. Paliwal, U.A. Kumar // Expert Systems with Applications. – 2009. – Vol. 36. – P. 2–17.

270. Mirmomeni M. Fuzzy descriptor systems and spectral analysis for chaotic time series prediction / M. Mirmomeni, C. Lucas, M. Shafiee, B.N. Araabi, E. Kamaliha // Neural Computing and Applications. – 2009. – Vol. 18. – P. 991–1004.

271. Осовский С. Нейронные сети для обработки информации / С. Осовский. – М.: Финансы и статистика, 2002. – 344 с.

272. Уоссермен Ф. Нейрокомпьютерная техника: теория и практика / Ф. Уоссермен. – М.: Мир, 1992. – 386 с.

273. Su Y.-S. Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models / Y.-S. Su, C.-Y. Huang // The Journal of Systems and Software. – 2007. – Vol. 80. – P. 606–615.

274. Cai K.Y. On the neural network approach in software reliability modeling / K.Y. Cai, L. Cai, W.D. Wang, Z.Y. Yu, D. Zhang // The Journal of Systems and Software. – 2001. – Vol. 58, Issue 1. – P. 47–62.

275. Dohi T. Optimal software release scheduling based on artificial neural networks / Dohi T., Nishio Y., Osaki S. // Annals of Software Engineering. – 1999. – Vol. 8. – P. 167–185.

276. Mei D. Early Software Reliability Prediction with Wavelet Networks Models / Denghua Mei // The 2007 International Conference on Intelligent Systems and Knowledge Engineering ISKE-2007 : Proceedings. – October 15–16, 2007. – P. 1–4.

277. Karunanithi N. Using neural networks in reliability prediction / Karunanithi N., Whitley D., Malaiya Y.K. // IEEE Software. – 1992. – Vol. 9, Issue 4. – P. 53–59.

278. Li A. Predicting Time between Software Failures Using ISGNN / A. Li, D. Qiu, Z.H. Li // International Journal of Computer Science and Network Security. – 2006. – Vol. 6, No 6. – P. 116–118.

279. Salmeron M. Parallel Computation of an Adaptive Optimal RBF Network Predictor / M. Salmeron, J. Ortega, C.G. Puntonet, M. Damas // LNCS. – 2003. – Vol. 2687. – P. 425–432.

280. Sing J.K. High-speed face recognition using self-adaptive radial basis function neural networks / J.K. Sing, S. Thakur, D.K. Basu, M. Nasipuri, M. Kundu // Neural Computing and Applications. – 2009. – Vol. 18. – P. 979–990.

281. Вороновский Г.К. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности / Г.К. Вороновский, К.В. Махотило, С.Н. Петрашев, С.А. Сергеев. – Харків : Основа, 1997. – 112 с.

282. Chromium Issues [Електронний ресурс]. – режим доступу: https://code.google.com/p/chromium/issues/list. – Назва з екрану.

283. Chromium-OS issue tracker [Електронний ресурс]. – режим доступу: https://code.google.com/p/chromium-os/issues/list. – Назва з екрану.

ДОДАТОК А. РЕЗУЛЬТАТИ ТЕСТУВАННЯ ПРОГРАМНИХ ЗАСОБІВ SPACE ТА SESD

Таблиця А.1

Результати 40 профілів тестування програмного засобу Space [90]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
10	9	9	10	10	10	9	9	9	9	10	8	9	9	9	10	9	10	9	10	10	9	9	9	9	10	9	9	10	10	10	10	9	9	9	10	10	10	9	9	9
20	15	15	14	17	16	15	17	16	14	19	16	18	14	13	15	14	16	17	16	17	16	13	16	15	15	17	15	15	18	14	18	15	17	17	18	17	17	14	16	15
30	18	20	17	19	20	22	21	21	18	23	21	21	18	19	19	17	23	19	19	20	18	16	21	19	18	23	19	21	21	18	22	20	21	20	20	22	21	19	18	18
40	22	23	20	21	22	26	22	23	21	25	24	23	21	22	21	18	24	23	21	24	21	19	24	20	21	25	24	24	23	22	25	20	22	23	22	23	24	22	19	20
50	22	23	21	22	23	28	24	24	23	25	27	24	23	24	21	19	26	26	23	25	26	20	24	21	23	27	24	24	23	23	27	23	24	23	27	24	25	24	20	21
60	25	24	22	23	24	28	25	24	26	29	27	24	24	25	23	22	27	27	25	25	26	22	24	21	24	28	24	24	24	24	27	24	27	24	28	25	26	24	22	25
70	26	25	25	24	26	28	26	25	27	29	27	26	24	26	25	23	28	27	26	26	26	22	26	23	25	28	24	27	26	24	28	26	28	27	28	26	27	26	24	25
80	26	27	27	24	26	29	26	26	29	30	28	28	25	26	28	23	28	28	26	27	27	23	27	25	26	29	24	29	26	25	30	26	30	27	29	27	28	26	25	26
90	26	28	30	26	28	32	27	26	32	30	28	29	26	26	28	24	28	28	27	28	28	23	28	25	26	29	25	31	27	25	31	27	33	28	29	27	28	27	28	26
100	27	29	31	26	28	33	27	27	33	30	29	29	26	26	28	25	29	29	28	29	30	23	28	25	26	29	26	31	29	25	31	27	33	28	33	27	28	28	30	27
110	28	29	31	27	29	33	29	27	33	31	30	30	26	29	28	26	29	30	28	30	31	23	28	27	27	29	26	31	29	26	31	28	33	29	34	27	30	28	30	27
120	28	29	32	28	29	34	30	27	34	31	31	32	27	29	28	26	29	30	29	31	32	24	29	28	27	29	26	31	29	27	31	29	33	29	34	28	30	28	30	28
130	28	29	32	29	29	34	30	27	34	31	31	32	27	29	28	27	30	30	29	32	32	24	29	29	27	29	28	32	30	28	31	31	33	31	34	29	31	28	32	28
140	28	29	32	30	29	34	31	28	34	31	32	32	27	29	30	28	31	31	29	32	34	24	29	29	27	29	28	32	30	28	31	31	34	32	34	32	31	28	32	28
150	28	29	32	31	29	34	31	28	34	31	33	32	27	29	30	29	31	31	29	32	34	24	29	29	27	29	29	32	30	28	32	31	35	32	34	34	31	30	32	29
160	28	30	33	31	30	34	31	29	34	32	33	32	27	29	31	30	31	32	31	32	34	24	30	29	27	29	29	32	31	29	32	31	35	32	34	34	32	30	32	29
170	28	30	34	31	30	34	32	29	34	32	33	32	27	29	31	30	31	32	32	33	34	26	31	30	27	29	29	32	31	29	32	32	35	33	34	34	32	31	32	29
180	29	30	34	31	31	34	32	29	34	32	33	32	28	29	31	30	31	32	32	33	34	26	32	31	28	30	29	34	32	29	32	32	35	33	34	34	33	31	32	29
190	29	31	34	32	31	34	32	29	34	32	33	32	28	29	31	31	31	32	32	33	34	26	32	31	29	30	29	34	32	30	32	32	35	33	34	34	33	32	32	29
200	29	31	34	33	31	34	32	30	34	32	33	32	28	29	31	31	31	32	33	33	34	27	34	31	29	30	30	34	32	30	32	32	35	33	34	34	33	32	32	29
210	29	31	34	33	32	34	32	30	34	32	33	32	28	29	31	31	31	32	33	33	34	27	35	32	29	30	30	34	32	30	32	32	35	34	34	34	33	32	32	29
220	29	31	34	34	32	34	33	30	34	32	33	33	29	29	31	31	31	32	33	33	34	27	35	32	29	30	30	34	32	30	33	32	35	34	34	34	33	32	32	30
230	29	31	34	34	32	34	33	30	34	32	33	33	29	29	32	31	31	32	33	33	34	27	35	32	29	30	30	34	33	30	33	32	35	34	34	34	33	33	32	31
240	31	31	34	34	32	34	33	30	34	32	33	33	29	30	32	31	31	32	33	33	34	27	35	32	30	30	30	34	33	30	33	33	35	34	34	34	33	33	32	32
250	31	31	35	34	32	34	33	31	34	32	33	33	29	31	32	32	32	32	33	33	34	28	35	32	30	30	30	34	33	30	33	33	35	34	34	34	33	33	32	32
260	31	31	35	34	32	34	33	31	34	32	33	34	29	31	32	33	32	32	33	33	34	28	35	32	31	31	32	34	33	31	33	33	35	34	34	34	33	33	32	32
270	32	31	35	34	32	34	33	31	34	32	33	34	29	31	32	33	32	32	33	33	34	28	35	33	31	31	32	34	34	31	33	33	35	34	34	34	33	33	32	32
280	32	31	35	34	32	34	33	31	34	32	33	34	29	31	32	33	32	32	33	33	34	28	35	33	32	31	32	34	34	32	33	33	35	34	34	34	33	33	33	32
290	32	31	35	34	32	34	33	31	34	32	33	34	29	31	32	33	32	32	33	33	34	30	35	33	32	31	32	34	34	32	33	33	35	34	34	34	33	33	33	32
300	32	32	35	34	32	34	33	32	35	32	33	34	30	31	32	33	32	32	33	33	34	30	35	33	33	31	33	34	34	32	33	33	35	34	34	35	33	33	33	32
310	33	32	35	34	32	34	33	32	35	32	33	34	30	31	33	33	32	32	33	34	34	30	35	33	33	31	33	34	34	32	33	33	35	36	34	35	33	33	33	32
320	33	32	35	34	32	34	33	32	35	32	33	35	30	31	33	34	33	32	33	34	34	30	35	33	33	31	34	34	34	33	33	33	35	36	34	35	33	33	33	32
330	33	32	35	34	32	34	33	32	35	32	33	35	31	31	33	34	33	32	33	34	34	30	35	33	33	32	35	34	34	33	33	33	35	36	34	35	33	33	33	32
340	33	32	35	34	32	34	33	32	35	32	33	35	32	31	33	34	33	32	33	34	34	30	35	33	33	32	35	34	34	33	33	33	35	36	34	35	33	33	33	32
350	33	32	35	34	32	34	33	32	35	32	33	36	32	31	33	34	33	32	33	34	34	30	35	33	33	32	35	34	34	33	33	33	35	36	34	35	33	33	33	32
360	33	32	35	34	32	34	33	32	35	32	33	36	32	31	34	34	33	32	33	34	34	31	35	33	33	32	35	34	34	33	33	33	35	36	34	35	33	34	34	32
370	33	32	35	34	32	34	33	32	35	32	33	36	32	31	34	34	33	32	33	34	34	31	35	33	33	32	35	34	34	33	33	33	35	36	35	35	33	34	34	32
380	33	32	35	34	32	34	33	32	35	32	33	36	32	31	34	34	33	32	33	34	34	31	35	33	33	32	35	34	34	33	33	33	35	36	35	35	33	35	34	33

390	33	32	35	34	32	34	33	32	35	32	33	36	32	31	34	34	33	33	33	34	34	31	35	33	33	32	35	34	34	33	33	34	35	36	35	35	33	35	34	33
400	33	32	35	34	32	34	34	32	35	32	33	36	32	31	34	34	33	33	33	34	34	31	35	34	33	32	35	34	34	33	33	34	35	36	35	35	33	35	34	33
410	33	32	35	34	32	34	34	32	35	32	33	36	32	31	34	34	33	33	33	34	34	31	35	34	33	32	35	34	34	33	33	34	35	36	35	35	33	35	34	33
420	33	32	35	34	32	34	34	32	35	32	33	36	32	31	34	34	33	33	33	34	34	31	35	34	33	32	35	34	34	33	33	34	35	36	35	35	33	35	34	33
430	33	32	35	34	32	34	34	32	35	32	33	36	32	31	34	34	33	33	33	34	34	31	35	34	33	32	35	34	34	33	33	34	35	36	35	35	33	35	34	33
440	33	32	35	34	32	34	34	32	35	32	33	36	32	31	34	34	33	33	34	34	34	31	35	34	33	32	35	34	34	33	33	34	35	36	35	35	33	35	34	33
450	33	32	35	34	32	34	34	32	35	32	33	36	32	32	34	34	33	33	34	34	34	33	35	34	33	32	35	34	34	33	33	34	35	36	35	35	33	35	34	33
460	33	32	35	34	32	34	34	32	35	32	33	36	32	32	35	34	33	33	34	34	34	33	35	34	33	32	35	34	34	33	33	34	35	36	35	35	33	35	34	33
470	33	32	35	34	32	34	34	32	35	32	33	36	32	32	35	34	33	33	34	34	34	33	35	34	33	32	35	34	34	33	33	34	35	36	35	35	33	35	34	33
480	33	33	35	34	32	34	34	32	35	32	33	36	32	32	35	34	33	33	35	34	34	33	35	34	33	32	35	34	34	33	33	34	35	36	35	35	33	35	34	33
490	33	33	35	34	32	35	34	32	35	32	33	36	32	32	35	34	33	33	35	34	34	34	35	34	33	32	35	34	34	33	33	34	35	36	35	35	33	35	34	33
500	34	33	35	34	32	35	34	32	35	32	33	36	32	32	35	34	33	33	35	34	34	34	35	34	33	32	35	34	34	33	33	34	35	36	35	35	33	35	34	33
510	34	33	35	34	32	35	34	33	35	32	33	36	32	32	35	34	33	33	35	34	34	34	35	34	33	33	35	34	34	33	33	34	35	36	35	35	33	35	34	33
520	34	34	35	34	32	35	34	33	35	32	33	36	32	32	35	34	33	33	35	34	34	34	35	34	34	33	35	34	34	33	33	35	35	36	35	35	33	35	34	33
530	34	34	35	34	33	35	34	33	35	32	33	36	32	32	35	34	33	33	35	34	34	34	35	34	34	33	35	34	34	33	33	35	35	36	35	35	33	35	34	33
540	34	34	35	34	33	35	34	33	35	32	33	36	32	32	35	34	33	33	37	34	34	34	35	34	34	33	35	34	34	33	33	35	35	36	35	35	33	35	34	33
550	34	34	35	34	33	35	34	33	35	32	33	36	32	32	35	34	33	33	37	34	34	34	35	34	34	33	35	34	34	33	33	35	35	36	35	35	33	35	34	33
560	34	34	35	34	33	35	34	33	35	32	33	36	32	32	35	34	33	33	37	34	34	34	35	34	34	33	35	34	34	33	33	35	35	36	35	35	33	35	34	33
570	34	34	35	34	33	35	34	33	35	32	33	36	32	32	35	34	33	33	37	34	34	34	35	34	34	33	35	34	34	33	33	35	35	36	35	35	33	35	34	33
580	34	34	35	34	33	35	34	34	35	32	33	36	32	32	35	35	33	33	37	34	34	34	35	34	34	33	35	34	34	33	33	35	35	36	35	35	33	35	34	33
590	34	34	35	34	33	35	34	34	35	32	33	36	32	32	35	35	34	33	37	34	34	34	35	34	34	33	35	34	34	33	33	35	35	36	35	35	33	35	34	33
600	34	34	35	34	33	35	34	34	35	32	33	36	32	32	35	35	34	33	37	34	34	34	35	34	34	33	35	34	34	33	33	35	35	36	35	35	33	35	34	33
610	34	34	35	34	33	35	34	34	35	32	33	36	32	32	35	35	34	33	37	34	34	34	35	34	34	33	35	34	34	33	33	35	35	36	35	35	33	35	34	33
620	34	34	35	34	33	35	34	34	35	33	33	36	32	32	35	35	34	34	37	34	34	34	35	34	34	33	35	34	34	33	33	35	35	36	35	35	33	35	34	33
630	34	34	35	34	33	35	34	34	35	33	33	36	32	32	35	35	34	34	37	34	34	34	35	34	34	33	35	34	34	33	33	35	35	36	35	35	33	35	34	33
640	34	34	35	34	33	35	34	35	35	33	33	36	32	32	35	35	34	35	37	34	34	34	35	34	34	34	35	34	34	33	33	35	35	36	35	35	33	35	34	33
650	34	34	35	34	33	35	34	36	35	33	33	36	32	32	35	35	34	35	37	34	34	34	35	34	34	34	35	34	34	33	33	35	35	36	35	35	33	35	34	33
660	34	34	35	34	33	35	34	36	35	33	33	36	32	32	35	35	34	35	37	34	34	34	35	34	34	34	36	34	34	33	33	35	35	36	35	35	33	35	34	33
670	34	34	35	34	33	35	34	36	35	33	33	36	33	32	35	35	34	35	37	34	34	34	35	34	34	34	36	34	34	33	33	35	35	36	35	35	33	35	34	33
680	34	34	35	34	33	35	34	36	35	34	33	36	33	32	35	35	34	35	37	34	34	34	35	34	34	34	36	34	34	33	33	35	35	36	35	35	33	35	34	33
690	34	34	35	34	33	35	34	36	35	34	33	36	33	32	35	35	34	36	37	34	34	34	35	34	34	34	36	34	34	33	33	35	35	36	35	35	33	35	34	33
700	34	34	35	34	33	35	34	36	35	34	33	36	33	32	35	35	34	36	37	34	34	34	35	34	34	34	36	34	34	33	33	35	35	36	35	35	33	35	34	33
710	34	34	35	34	33	35	34	36	35	34	34	36	33	32	35	36	34	36	37	34	34	34	35	34	34	34	36	34	34	33	33	35	35	36	35	35	33	35	34	33
720	34	34	35	34	33	35	34	36	35	34	34	36	33	32	35	36	34	36	37	34	34	34	35	34	34	34	36	34	34	33	33	35	35	36	35	35	33	35	34	33
730	34	34	35	34	33	35	34	36	35	34	34	36	33	32	35	36	34	36	37	34	34	35	35	34	34	34	36	34	34	33	33	35	35	36	35	35	33	35	34	33
740	34	34	35	34	33	35	34	36	35	34	34	36	33	32	35	36	34	36	37	34	34	35	35	34	34	34	36	34	34	33	33	35	35	36	35	35	33	35	34	34
750	34	34	35	34	33	35	34	36	35	34	34	36	33	32	35	36	34	36	37	34	34	35	35	34	34	34	36	34	34	33	33	35	35	36	35	35	33	35	34	34
760	34	34	35	34	33	35	34	36	35	34	34	36	33	32	35	36	34	36	37	34	34	35	35	34	34	34	36	34	34	33	33	35	35	36	35	35	33	35	34	34
770	34	34	35	35	33	35	34	36	35	34	34	36	33	32	35	36	34	36	37	34	34	35	35	34	34	34	36	34	34	33	33	35	35	36	35	35	33	35	34	34
780	34	34	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	34	34	35	35	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
790	34	34	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	34	34	35	35	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
800	34	34	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	34	34	35	35	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
810	34	34	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	34	34	35	35	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
820	34	34	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	34	34	35	35	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
830	34	34	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	34	34	35	35	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
840	34	34	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	34	34	35	36	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34

850	34	34	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	34	34	35	36	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
860	34	34	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	34	34	35	36	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
870	34	34	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	34	34	35	36	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
880	34	34	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	34	34	35	36	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
890	34	34	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	35	34	35	36	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
900	34	34	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	35	34	35	36	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
910	34	34	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	35	34	35	36	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
920	34	34	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	35	34	35	36	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
930	34	35	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	35	34	35	36	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
940	34	35	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	35	34	35	36	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
950	34	35	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	35	34	35	36	34	34	34	36	34	34	33	33	35	35	36	35	35	34	35	34	34
960	34	35	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	35	34	35	36	34	34	34	36	34	35	33	33	35	35	36	35	35	34	35	34	34
970	34	35	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	35	34	35	36	34	34	34	36	34	35	33	33	35	35	36	35	35	34	35	34	34
980	34	35	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	35	34	35	36	34	34	34	36	34	36	33	33	35	35	36	35	35	34	35	34	34
990	34	35	35	35	33	35	34	36	35	34	34	36	33	33	35	36	34	36	37	35	34	35	36	34	34	34	36	34	36	33	33	35	35	36	35	35	34	35	34	34
1000	34	35	35	35	33	35	34	36	36	34	34	36	33	33	35	36	34	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1010	34	35	35	35	33	35	34	36	36	34	34	36	33	33	35	36	34	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1020	34	35	35	35	33	35	34	36	36	34	34	36	33	33	35	36	34	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1030	34	35	35	35	33	35	34	36	36	34	34	36	33	33	35	36	34	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1040	34	35	35	35	33	35	34	36	36	34	34	36	33	33	35	36	34	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1050	34	35	35	35	33	35	34	36	36	34	34	36	33	33	35	36	35	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1060	34	35	35	35	33	35	34	36	36	34	34	36	33	33	35	36	35	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1070	34	35	35	35	33	35	34	36	36	34	34	36	33	33	35	36	35	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1080	34	35	35	35	33	35	34	36	36	34	34	36	33	33	35	36	35	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1090	34	35	35	35	33	35	34	36	36	35	34	36	33	33	35	36	35	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1100	34	35	35	35	33	35	34	36	36	35	34	36	33	33	35	36	35	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1110	34	35	35	35	33	35	34	36	36	35	34	36	33	33	35	36	35	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1120	34	35	35	35	33	35	34	36	36	35	34	36	33	33	35	36	35	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1130	34	35	35	35	33	35	34	36	36	35	34	36	33	33	35	36	35	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1140	34	35	35	35	33	35	34	36	36	35	35	36	33	33	35	36	35	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1150	34	35	35	35	33	35	34	36	36	35	35	36	33	34	35	36	35	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1160	34	35	35	35	33	35	34	36	36	35	35	36	33	34	35	36	35	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1170	34	35	35	35	33	35	34	36	36	35	35	36	33	34	35	36	35	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1180	34	35	35	35	33	35	34	36	36	35	35	36	33	34	35	36	35	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1190	34	35	35	35	33	35	34	36	36	35	35	36	34	34	35	36	35	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34
1200	34	35	35	35	33	35	35	36	36	35	35	36	34	34	35	36	35	36	37	35	34	35	36	34	34	34	36	34	36	33	33	36	35	36	35	35	34	35	34	34

Примітка: стовпці таблиці відповідають профілям тестування, рядки – ітераціям. Дані таблиці – кумулятивна кількість відмов, виявлених протягом відповідної кількості ітерацій.

Таблиця А.2

Результати 40 профілів тестування програмного засобу SESD [90]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
10	7	6	8	8	9	10	8	8	9	8	8	9	10	7	9	3	8	10	8	7	7	8	7	8	7	9	9	9	9	9	9	9	7	7	6	7	10	8	10	8
20	16	13	14	16	18	17	13	14	16	15	14	16	16	14	16	10	16	16	13	15	14	15	16	16	14	16	17	14	16	15	18	15	15	15	14	12	16	18	15	17
30	21	18	18	18	18	22	17	17	22	18	17	17	18	19	18	16	21	17	18	20	18	16	20	17	21	20	19	18	18	17	19	19	17	18	19	18	21	20	17	21
40	22	21	19	18	20	22	20	18	23	18	20	17	19	20	19	19	21	17	20	20	19	17	21	18	22	22	20	18	21	18	19	20	18	18	20	20	22	21	18	22
50	22	22	19	18	22	22	22	19	23	19	21	19	19	21	20	20	21	19	22	20	19	18	21	21	22	22	21	19	21	19	20	21	20	20	21	21	23	22	19	22
60	22	22	19	19	22	23	22	19	23	19	21	19	19	23	20	21	21	20	22	20	21	18	21	21	23	22	22	19	22	20	20	21	20	20	21	21	24	22	20	23
70	22	22	21	20	24	23	22	19	23	19	21	20	20	23	20	21	21	21	23	21	21	19	22	21	24	22	23	21	22	20	21	22	21	20	21	21	24	22	22	23
80	22	22	22	20	24	23	22	19	23	19	21	22	21	23	20	21	21	21	23	22	22	20	23	21	24	23	23	22	22	21	21	25	21	21	21	21	24	23	22	23
90	22	22	23	22	25	23	22	20	24	20	22	23	21	24	21	21	22	21	25	22	22	21	23	21	24	23	23	22	23	21	23	25	22	22	21	21	24	23	22	23
100	22	22	23	22	25	23	22	21	24	21	22	24	21	24	22	21	22	21	25	22	23	21	23	21	24	23	23	22	23	21	23	25	22	22	21	21	24	23	22	23
110	22	22	24	22	25	24	22	21	24	22	22	24	21	24	22	21	22	21	25	22	24	21	23	21	24	23	24	22	24	21	23	25	22	22	23	21	24	24	22	23
120	22	22	24	22	25	25	22	22	24	22	23	24	22	24	22	23	22	21	25	23	24	22	23	21	24	23	24	22	24	23	23	25	22	22	23	21	24	24	22	24
130	23	22	25	22	25	25	22	22	24	22	23	24	22	24	22	23	22	21	25	23	24	22	24	22	24	23	24	22	24	23	23	25	23	22	24	21	24	25	22	24
140	23	23	25	23	25	25	23	23	24	23	24	24	22	24	22	23	22	22	25	23	24	22	24	23	24	23	24	22	24	23	24	25	23	22	25	21	24	25	22	24
150	23	24	25	23	25	25	24	23	24	23	24	24	22	24	22	23	22	22	25	23	24	22	24	23	24	24	25	22	24	24	24	25	23	23	25	21	24	25	22	24
160	23	24	25	23	25	25	25	23	24	23	24	24	22	24	22	23	22	22	25	24	24	23	24	23	24	25	25	22	24	24	24	25	23	23	25	22	24	25	22	24
170	23	25	25	24	25	25	25	23	24	23	25	24	23	24	22	23	22	22	25	24	24	23	25	23	24	25	25	22	24	24	24	25	23	23	25	22	24	25	22	24
180	24	25	25	24	25	25	25	23	24	23	25	24	23	24	22	24	22	22	25	24	24	23	25	23	24	25	25	22	24	24	24	25	23	23	25	22	24	25	22	24
190	24	25	25	24	25	25	25	24	24	23	25	24	24	24	22	24	23	22	25	24	24	23	25	23	24	25	25	22	24	24	24	25	23	23	25	22	24	25	22	24
200	24	25	25	24	25	25	25	24	24	23	25	24	24	25	22	24	23	22	25	24	24	23	25	23	24	25	25	22	24	24	24	25	23	23	25	22	24	25	22	24
210	24	25	25	24	25	25	25	24	24	23	25	24	24	25	22	24	23	22	25	24	25	23	25	23	24	25	25	22	24	24	24	25	23	23	25	22	24	25	22	24
220	24	25	25	24	25	25	25	24	24	23	25	24	24	25	22	24	23	22	25	24	25	24	25	23	24	25	25	22	24	24	24	25	23	23	25	22	24	25	22	24
230	24	25	25	24	25	25	25	24	24	23	25	24	24	25	23	24	23	22	25	24	25	24	25	23	24	25	25	22	24	24	24	25	23	23	25	22	24	25	22	24
240	24	25	25	25	25	25	25	24	24	23	25	24	24	25	23	24	23	23	25	24	25	24	25	24	24	25	25	22	24	24	24	25	23	23	25	22	24	25	22	24
250	24	25	25	25	25	25	25	24	24	23	25	24	24	25	23	25	23	23	25	24	25	24	25	24	24	25	25	22	24	24	24	25	23	23	25	22	24	25	22	24
260	24	25	25	25	25	25	25	24	24	23	25	24	24	25	23	25	23	23	25	24	25	24	25	24	24	25	25	22	24	24	24	25	23	23	25	22	24	25	22	24
270	24	25	25	25	25	25	25	24	24	23	25	24	24	25	23	25	23	23	25	24	25	24	25	24	24	25	25	22	24	24	24	25	23	23	25	22	24	25	22	25
280	24	25	25	25	25	25	25	24	24	23	25	24	24	25	23	25	23	23	25	24	25	25	25	24	24	25	25	22	24	24	24	25	23	23	25	22	24	25	22	25
290	24	25	25	25	25	25	25	24	24	23	25	24	24	25	23	25	23	23	25	24	25	25	25	24	24	25	25	22	24	24	24	25	23	23	25	22	24	25	22	25
300	24	25	25	25	25	25	25	24	24	24	25	24	24	25	23	25	23	23	25	24	25	25	25	24	24	25	25	22	24	24	24	25	23	23	25	22	24	25	22	25
310	24	25	25	25	25	25	25	24	24	25	25	24	24	25	24	25	23	24	25	24	25	25	25	24	24	25	25	22	24	24	24	25	23	23	25	22	24	25	23	25
320	24	25	25	25	25	25	25	24	24	25	25	24	24	25	24	25	23	24	25	24	25	25	25	24	24	25	25	22	24	24	24	25	23	23	25	22	24	25	23	25
330	24	25	25	25	25	25	25	24	24	25	25	24	24	25	24	25	23	24	25	24	25	25	25	24	24	25	25	22	24	24	24	25	23	23	25	22	24	25	23	25
340	24	25	25	25	25	25	25	24	24	25	25	24	24	25	24	25	23	24	25	24	25	25	25	24	24	25	25	22	24	24	24	25	23	23	25	23	24	25	24	25
350	24	25	25	25	25	25	25	24	24	25	25	24	24	25	24	25	23	24	25	24	25	25	25	24	24	25	25	23	24	24	24	25	24	23	25	23	24	25	24	25
360	24	25	25	25	25	25	25	24	24	25	25	24	24	25	24	25	23	24	25	24	25	25	25	24	24	25	25	23	24	24	24	25	24	23	25	23	24	25	24	25
370	24	25	25	25	25	25	25	24	24	25	25	24	24	25	24	25	23	24	25	24	25	25	25	24	24	25	25	23	24	25	24	25	24	23	25	23	25	25	24	25
380	24	25	25	25	25	25	25	24	25	25	25	24	24	25	24	25	23	24	25	24	25	25	25	24	25	25	25	23	24	25	24	25	24	23	25	23	25	25	24	25
390	24	25	25	25	25	25	25	24	25	25	25	24	24	25	24	25	23	24	25	24	25	25	25	24	25	25	25	23	24	25	24	25	24	23	25	23	25	25	24	25
400	24	25	25	25	25	25	25	24	25	25	25	24	24	25	24	25	23	24	25	24	25	25	25	24	25	25	25	23	24	25	24	25	24	23	25	23	25	25	24	25
410	24	25	25	25	25	25	25	24	25	25	25	24	24	25	24	25	23	24	25	24	25	25	25	24	25	25	25	23	24	25	24	25	24	25	25	23	25	25	24	25
420	24	25	25	25	25	25	25	24	25	25	25	24	24	25	24	25	24	24	25	24	25	25	25	24	25	25	25	23	24	25	24	25	24	25	25	23	25	25	24	25

430	24	25	25	25	25	25	25	24	25	25	25	24	24	25	24	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	24	25	25	23	25	25	24	25
440	24	25	25	25	25	25	25	24	25	25	25	24	24	25	24	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	24	25	25	23	25	25	25	25
450	24	25	25	25	25	25	25	24	25	25	25	24	24	25	24	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	24	25	25	23	25	25	25	25
460	24	25	25	25	25	25	25	24	25	25	25	24	24	25	24	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	24	25	25	24	25	25	25	25
470	24	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	24	25	25	24	25	25	25	25
480	24	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	24	25	25	24	25	25	25	25
490	24	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
500	24	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
510	24	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
520	24	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
530	24	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
540	24	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
550	24	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
560	24	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
570	24	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
580	24	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
590	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	24	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
600	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
610	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
620	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
630	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
640	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
650	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
660	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
670	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
680	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
690	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
700	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
710	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
720	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
730	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
740	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
750	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
760	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
770	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
780	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
790	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
800	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	24	25	25	25	25	24	25	25	25	25
810	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	24	25	25	25	25
820	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25	25	25
830	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25	25	25
840	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25	25	25
850	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25	25	25
860	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25	25	25
870	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25	25	25
880	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25	25	25

890	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
900	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	24	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
910	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
920	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
930	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
940	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
950	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
960	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
970	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
980	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
990	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
1000	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
1010	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
1020	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
1030	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
1040	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
1050	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
1060	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
1070	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
1080	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
1090	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
1100	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
1110	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
1120	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	23	25	25	25	25	25	25	25	25	25	25
1130	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	24	25	25	25	25	25	25	25	25	25	25
1140	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	24	25	25	25	25	25	25	25	25	25	25
1150	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	24	25	25	25	25	25	25	25	25	25	25
1160	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	24	25	25	25	25	25	25	25	25	25	25
1170	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	24	25	25	25	25	25	25	25	25	25	25
1180	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	24	25	25	25	25	25	25	25	25	25	25
1190	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	24	25	25	25	25	25	25	25	25	25	25
1200	25	25	25	25	25	25	25	25	25	25	25	25	24	25	25	25	25	24	25	25	25	25	25	24	25	25	25	24	25	25	25	25	25	25	25	25	25	25

Примітка: стовпці таблиці відповідають профілям тестування, рядки – ітераціям. Дані таблиці – кумулятивна кількість відмов, виявлених протягом відповідної кількості ітерацій.

ДОДАТОК Б. СВІДОЦТВО ПРО РЕЄСТРАЦІЮ АВТОРСЬКОГО ПРАВА

УКРАЇНА

ДЕРЖАВНА СЛУЖБА ІНТЕЛЕКТУАЛЬНОЇ ВЛАСНОСТІ УКРАЇНИ

СВІДОЦТВО

про реєстрацію авторського права на твір

№ 58985

Комп'ютерна програма "Система оцінювання надійності програмного забезпечення"

(вид, назва службового твору)

Автор(и) **Яковина Віталій Степанович, Нитребич Оксана Олександрівна, Сердюк Павло Віталійович**

(повне ім'я, псевдонім (за наявності))

Авторські майнові права належать **Національний університет "Львівська політехніка", вул. Ст. Бандери, 12, м. Львів, 79013**

(повне ім'я фізичної та/або повне офіційне найменування юридичної особи, адреса)

24.03.2015

Дата реєстрації

Державна служба інтелектуальної власності України. Ідентифікаційний код 37552556. М.П.

Голова Державної служби інтелектуальної власності України
А.Г.Жарінова

ДОДАТОК В. АКТИ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЇ У ВИРОБНИЦТВО

МІНІСТЕРСТВО ЕКОНОМІЧНОГО РОЗВИТКУ І ТОРГІВЛІ УКРАЇНИ

ДЕРЖАВНЕ ПІДПРИЄМСТВО
«НАУКОВО-ДОСЛІДНИЙ ІНСТИТУТ МЕТРОЛОГІЇ
ВИМІРЮВАЛЬНИХ І УПРАВЛЯЮЧИХ СИСТЕМ»
(ДП НДІ «СИСТЕМА»)

вул. М.Кривоноса, 6, м.Львів, 79008, Україна,
тел. (032) 239-92-00, 255-49-39, факс (032) 235-84-49
Web: http: // www.dndi-systema.lviv.ua
E-mail: office@dndi-systema.lviv.ua

р/р 26009053805720 у відділенні банку «Приватбанк»
у м.Львові, МФО 325321, код ЄДРПОУ 04728690

_____№ _____
На №_____ від _____

ЗАТВЕРДЖУЮ
Директор Державного підприємства
"Науково-дослідний інститут метрології
вимірювальних і управляючих систем",
к.т.н., доц.
_____ В.В. Паракуда
28 07 2015 р.

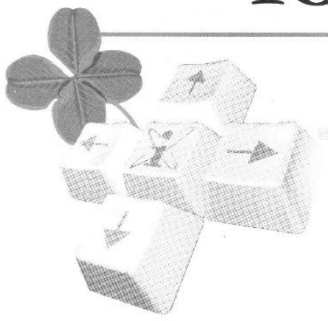
АКТ
про впровадження результатів
дисертаційної роботи завідувача кафедри програмного забезпечення
Національного університету «Львівська політехніка»
Яковини Віталія Степановича

Даний акт складений про те, що програмний засіб прогнозування відмов програмного забезпечення на основі нейронних мереж, розроблений під час виконання дисертаційної роботи Яковини Віталія Степановича на тему «*Методи та засоби аналізу надійності функціонування програмного забезпечення з урахуванням етапів життєвого циклу*» використовується у Державному підприємстві "Науково-дослідний інститут метрології вимірювальних і управляючих систем" (ДП НДІ "Система") для прогнозування відмов програмно-технічних систем спеціального призначення.

Під час етапу експлуатації програмно-технічних систем, у порівнянні з традиційними засобами аналізу та прогнозування відмов програмного забезпечення, використання розробленого дисертантом програмного засобу дало можливість підвищити точність прогнозу відмов на 7–12 % при одночасному зменшенні трудомісткості та тривалості процесу прогнозування, що загалом дає можливість підвищити якість програмно-технічних систем.

Даний акт не є підставою для фінансових розрахунків.

Заступник директора з наукової роботи
та якості, начальник НДВ-11, к.т.н., с.н.с. О.М. Кричевець



lóhika

making IT logical

«Затверджую»
Директор ТзОВ «Логіка ЛТД»
____ _____
__ _____ 2015 р.

АКТ
про дослідне випробування результатів
дисертаційної роботи завідувача кафедри програмного забезпечення
Національного університету «Львівська політехніка»
Яковини Віталія Степановича

Даний акт складений про те, що модель надійності програмного забезпечення у вигляді ланцюга Маркова вищого порядку та метод представлення такого процесу у вигляді еквівалентного процесу першого порядку, розроблений під час виконання дисертаційної роботи Яковини Віталія Степановича на тему «*Методи та засоби аналізу надійності функціонування програмного забезпечення з урахуванням етапів життєвого циклу*» пройшла дослідне випробування на ІП «Логіка» для оцінювання якості розроблюваних програмних систем, зокрема показників їх надійності.

Використання розробленої дисертантом моделі надійності на етапі інтеграційного тестування та супроводу програмного забезпечення, у порівнянні з традиційними засобами оцінювання показників надійності дало можливість підвищити достовірність оцінювання інтенсивності відмов та середнього часу між відмовами комерційних програмних систем на 11–17 %.

Даний акт не є підставою для взаємних фінансових розрахунків.

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ
«САЙПРЕСС СЕМІКОНДАКТОР УКРАЇНА»
ІДЕНТИФІКАЦІЙНИЙ КОД 39163300
ЮР. АДРЕСА: УКРАЇНА, 79034, М. ЛЬВІВ, СИХІВСЬКИЙ Р-Н,
ВУЛ. НАВРОЦЬКОГО, БУД. 10, КІМНАТА 311

«Затверджую»
Директор ТзОВ «Сайпресс Семікондактор Україна»
_____
_____ 2015 р.

АКТ
про впровадження результатів
дисертаційної роботи завідувача кафедри програмного забезпечення
Національного університету «Львівська політехніка»
Яковини Віталія Степановича

Даний акт складений про те, що на ТзОВ «Сайпресс Семікондактор Україна» (м. Львів) впроваджено результати дисертаційного дослідження Яковини Віталія Степановича на тему «*Методи та засоби аналізу надійності функціонування програмного забезпечення з урахуванням етапів життєвого циклу*», а саме удосконалену процедуру прогнозування кількості залишкових помилок в програмному забезпеченні.

Удосконалена процедура прогнозування кількості залишкових помилок програмного забезпечення передбачає отримання прогнозу кількості помилок з використанням результатів тестування та моделі надійності у вигляді неоднорідного пуассонового процесу послідовно в процесі тестування програмного продукту, після чого на основі регресійного аналізу отриманих значень, отримують уточнене значення прогнозованої кількості помилок в програмному забезпеченні.

Використання зазначеної процедури в процесі тестування програмних засобів, що розробляються ТзОВ «Сайпресс Семікондактор Україна» порівняно з іншими методами оцінювання та прогнозування кількості помилок в програмному засобі забезпечило зменшення тривалості тестування, необхідного для отримання прийнятної точності прогнозу на 17–23 %, що дало можливість підвищити ефективність процесу тестування програмного забезпечення.

Даний акт не є підставою для фінансових розрахунків.

PromTechnoService

Товариство з обмеженою
відповідальністю науково-виробнича
фірма «Промтехносервіс Україна»,
03150, Україна, м. Київ,
вул. Червоноармійська, буд. 114,
тел.: (044) 5031527
info@prom-technoservice.com

«Затверджую»
Директор ТОВ
«НВФ «Промтехносервіс Україна»
_____ Муляк О.В.
25 серпня 2015 року

АКТ
про впровадження результатів
дисертаційної роботи завідувача кафедри програмного забезпечення
Національного університету «Львівська політехніка»

Яковини Віталія Степановича

Даний акт складений про те, що на ТОВ «НВФ «Промтехносервіс Україна» (м. Київ) впроваджено такі результати дисертаційного дослідження Яковини Віталія Степановича на тему «*Методи та засоби аналізу надійності функціонування програмного забезпечення з урахуванням етапів життєвого циклу*»:

- модель надійності програмного забезпечення з показником складності та метод аналізу надійності програмно-апаратних систем на її основі.

Розроблена дисертантом модель надійності програмного забезпечення у вигляді неоднорідного пуассонового процесу з показником складності дає вхідні дані стосовно показників надійності програмного забезпечення розробникам програмно-апаратних систем, до складу яких воно входить, що дає можливість здійснювати системотехнічне проектування сучасних програмно-апаратних систем в надійнісному аспекті з урахуванням поведінки надійності програмного забезпечення. Використання цієї моделі під час проектування програмно-апаратних технічних систем дало можливість підвищити точність та достовірність оцінювання показників їх надійності порівняно з найбільш поширеними моделями надійності програмного забезпечення. Зокрема точність та достовірність оцінювання інтенсивності відмов програмного забезпечення зростає на 50–150 %, а функції готовності програмно-апаратної системи – на 3–17 % залежно від алгоритму функціонування системи.

Даний акт не є підставою для фінансових розрахунків.

«Затверджую»
Директор ПП «Лінк Ап Студіо»
Здебська Г. М.
26 серпня 2015 р.

АКТ
про впровадження результатів
дисертаційної роботи завідувача кафедри програмного забезпечення
Національного університету «Львівська політехніка»
Яковини Віталія Степановича

Даний акт складений про те, що програмний засіб оцінювання надійності програмного забезпечення (СОН ПЗ), розроблений під час виконання дисертаційного дослідження Яковини Віталія Степановича на тему «*Методи та засоби аналізу надійності функціонування програмного забезпечення з урахуванням етапів життєвого циклу*» впроваджено на ПП «Лінк Ап Студіо» (м. Львів).

Програмний засіб "СОН ПЗ" дає можливість отримати значення параметрів моделі надійності програмного забезпечення вищого порядку, зокрема матрицю інтенсивностей переходів між модулями програмної системи на основі моніторингу її виконання, та отримати оцінку показників надійності складних програмних систем на різних етапах життєвого циклу.

Під час етапу конструювання та тестування програмного забезпечення на ПП «Лінк Ап Студіо» використання програмного засобу "СОН ПЗ" дало можливість підвищити точність оцінювання показників надійності програмного забезпечення, зокрема імовірності безвідмовної роботи на 7–12 %, а інтенсивності відмов – на 14–18 %. Отримання достовірного прогнозу показників надійності програмного забезпечення на етапі конструювання дало можливість зменшити собівартість розроблених ПП «Лінк Ап Студіо» програмних продуктів на 5–7 %.

Даний акт не є підставою для взаємних фінансових розрахунків.

ТОВ "СЕА Електротехніка"
ЄДРПОУ 38013959
вул. Краківська, буд.13-Б, офіс 128,
м. Київ, 02094
тел./факс : (044) 291-00-20
Р/р № 26004416341400
в АТ "УКРСИББАНК"
МФО 351005

http://www.sea.com.ua
e-mail : info@sea.com.ua

Ефективне
енергозбереження

№ 1529 від 20.08.2015

На № _____

АКТ
про впровадження результатів
дисертаційної роботи завідувача кафедри програмного забезпечення
Національного університету «Львівська політехніка»
Яковини Віталія Степановича

Даний акт складений про те, що модель надійності програмного забезпечення у вигляді ланцюга Маркова вищого порядку та спосіб моделювання вимог до програмного забезпечення з використанням методу аналізу ієрархій, розроблені в рамках дисертаційного дослідження Яковини Віталія Степановича на тему «*Методи та засоби аналізу надійності функціонування програмного забезпечення з урахуванням етапів життєвого циклу*» впроваджено в ТОВ «СЕА Електротехніка» в автоматизованих системах моніторигу та управління для підприємств ЖКГ.

Використання розробленої Яковиною В.С. моделі надійності програмних систем на етапі тестування, у порівнянні з традиційними засобами оцінювання показників надійності дало можливість підвищити достовірність оцінювання імовірності відмови та середнього часу між відмовами розроблюваних програмних систем на 9–13 %.

Використання способу моделювання вимог до програмного забезпечення з використанням методу аналізу ієрархій дало можливість отримати оцінки імовірностей виконання різних сценаріїв використання, що, за рахунок наближення тестового профілю до експлуатаційного, дало можливість підвищити якість програмних продуктів, зокрема зменшити кількість відмов на етапі експлуатації на 5–7 %.

Даний акт не є підставою для взаємних фінансових розрахунків.

Директор І.В. Ленда

ДОДАТОК Г. АКТИ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЇ В НАВЧАЛЬНИЙ ПРОЦЕС

«Затверджую»
Проректор з науково-педагогічної роботи
Національного університету
«Львівська політехніка»
_____ Давидчак О.Р.
__ _____ 2015 р.

АКТ

Про впровадження результатів докторської дисертаційної роботи Яковини Віталія Степановича «*Методи та засоби аналізу надійності функціонування програмного забезпечення з урахуванням етапів життєвого циклу*» у навчальний процес на кафедрі програмного забезпечення Національного університету «Львівська політехніка».

Зокрема, у навчальному процесі (дисципліна «Якість програмного забезпечення та тестування» для студентів 4-го курсу освітньо-кваліфікаційного рівня «бакалавр» для базового напряму 6.050103 «Програмна інженерія») використовувались розроблені Яковиною В.С.:

- метод прогнозування кількості відмов ПЗ з використанням регресійного аналізу та моделі надійності з показником складності (Тема 12 "Документування і оцінка індустріального тестування. Документування і життєвий цикл дефекту");
- непараметричні моделі прогнозування надійності ПЗ у вигляді нейронних мереж RBF та Елмана (Тема 12 "Документування і оцінка індустріального тестування. Документування і життєвий цикл дефекту").

Розроблені моделі та метод прогнозування відмов дають змогу підвищити ефективність процесу тестування програмного забезпечення та прогнозувати надійність програмного забезпечення.

Лектор курсу
к.т.н., доц. Білас О.Є.

Професор кафедри програмного забезпечення,
д.т.н., проф. Федасюк Д.В.

Директор інституту комп'ютерних наук
та інформаційних технологій,
д.т.н., проф. Медиковський М.О.

«Затверджую»
Проректор з науково-педагогічної роботи
Національного університету
«Львівська політехніка»
_____ Давидчак О.Р.
__ _____ 2015 р.

АКТ

Про впровадження результатів докторської дисертаційної роботи Яковини Віталія Степановича «*Методи та засоби аналізу надійності функціонування програмного забезпечення з урахуванням етапів життєвого циклу*» у навчальний процес на кафедрі програмного забезпечення Національного університету «Львівська політехніка».

Зокрема, у навчальному процесі (дисципліна «Аналіз вимог до програмного забезпечення» для студентів 3-го курсу освітньо-кваліфікаційного рівня «бакалавр» для базового напряму 6.050103 «Програмна інженерія») використовувався розроблений Яковиною В.С.:

- спосіб моделювання вимог до програмного забезпечення з використанням методу аналізу ієрархій (Тема 7 "Основні принципи створення вимог. Моделювання вимог").

Розроблений спосіб моделювання вимог до програмного забезпечення дає змогу підвищити ефективність процесу аналізу вимог до програмного забезпечення та отримати оцінки параметрів моделей надійності програмного забезпечення на ранніх етапах його життєвого циклу.

Лектор курсу
к.т.н., доц. Білас О.Є.

Професор кафедри програмного забезпечення,
д.т.н., проф. Федасюк Д.В.

Директор інституту комп'ютерних наук
та інформаційних технологій,
д.т.н., проф. Медиковський М.О.

«Затверджую»
Проректор з науково-педагогічної роботи
Національного університету
«Львівська політехніка»
_____ Давидчак О.Р.
__ _____ 2015 р.

АКТ

Про впровадження результатів докторської дисертаційної роботи Яковини Віталія Степановича «*Методи та засоби аналізу надійності функціонування програмного забезпечення з урахуванням етапів життєвого циклу*» у навчальний процес на кафедрі програмного забезпечення Національного університету «Львівська політехніка».

Зокрема, у навчальному процесі (дисципліна «Основи теорії надійності програмних систем» для студентів 1-го курсу освітньо-кваліфікаційного рівня «магістр» для спеціальностей 8.05010301 «Програмне забезпечення систем» та 8.05010302 «Інженерія програмного забезпечення») використовувались розроблені Яковиною В.С.:

- модель надійності програмних систем у вигляді ланцюгів Маркова вищого порядку з неперервним часом (Тема 8 "Аналіз надійності програмного забезпечення");
- модель надійності з урахуванням недосконалої інтеграції модулів програмних систем (Тема 8 "Аналіз надійності програмного забезпечення");
- спосіб представлення марковського процесу вищого порядку у вигляді еквівалентного процесу першого порядку з віртуальними станами (Тема 6 "Методи аналізу надійності технічних систем");
- узагальнений метод аналізу надійності програмних систем з урахуванням їх складності, архітектури та етапів життєвого циклу (Тема 6 "Методи аналізу надійності технічних систем").

Розроблені моделі та методи дають змогу підвищити достовірність оцінювання показників надійності програмних систем.

Лектор курсу
к.ф.-м.н., доц. Яковина В.С.

Професор кафедри програмного забезпечення,
д.т.н., проф. Федасюк Д.В.

Директор Інституту комп'ютерних наук
та інформаційних технологій,
д.т.н., проф. Медиковський М.О.