УДК 004.052

В. Яковина, В. Мацелюх
Національний університет “Львівська політехніка”,
кафедра програмного забезпечення

ОГЛЯД І АНАЛІЗ МОДЕЛЕЙ НАДІЙНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Оглянуто та проаналізовано моделі надійності програмного забезпечення. Наведено класифікацію моделей за різними критеріями. Особливу увагу зосереджено на моделях надійності, що враховують явище недосконалого відлагодження, зокрема на моделях на основі неоднорідного пуассонового процесу з використанням функцій розподілу зусиль тестування. Розглянуто основні функції розподілу зусиль тестування та проаналізовано їх інтеграцію з моделями надійності програмного забезпечення.

Ключові слова: програмне забезпечення, якість, надійність, модель, недосконале відлагодження, розподіл зусиль тестування.

Вступ

В останні роки вартість розроблення програмного забезпечення (ПЗ) та витрати на відмови програмних систем стали одними із основних затрат на розроблення складних технічних систем. Відмова ПЗ може призвести до неочікуваного стану системи або потоку непередбачених дій. Внаслідок відмови ПЗ можливі пошкодження чи руйнування майна, поранення або загибель людей, втрата грошових коштів. Загальновизнаним в індустрії програмного забезпечення є той факт, що висококваліфіковані програмісти допускають в середньому 6 помилок в програмному забезпеченні при написанні 1000 рядків коду. При такому показнику звичайне комерційне ПЗ, яке складається з 350000 рядків коду, містить близько 2000 помилок, зокрема помилки, пов’язані з витоком пам’яті, пов’язані з текстом програми, помилки використання сторонніх бібліотек, стандартні бібліотечні помилки та інші. Тому питання оцінювання та забезпечення надійності ПЗ відіграють все більшу роль у розробленні складних технічних систем, а побудова все адекватніших моделей надійності ПЗ є важливою науково-технічною проблемою.

Загальна характеристика та класифікація моделей надійності програмного забезпечення

Дослідження в галузі надійності ПЗ беруть початок з 1970-х років. Незважаючи на зусилля провідних світових розробників ПЗ, завдання зниження кількості помилок у програмних системах не отримало ефективного практичного вирішення [1]. Об’єктивно це зумовлено надзвичайно високою структурною складністю програмних систем, динамічністю версій і технологій. Одним із шляхів підвищення рівня надійності ПЗ є використання на етапах тестування і випробувань ПЗ моделей, що дають змогу отримати гарантовані оцінки показників безпеки ПЗ і ефективності технології його розроблення. Більшість таких моделей запозичено з теорії надійності технічних систем, тому в літературі їх часто називають моделями надійності ПЗ [2, 3].

Від 70-х років минулого століття було розроблено значну кількість моделей надійності ПЗ [4–7]. Як основні критерії класифікації таких моделей вибрано два доволі прості предмети дослідження:

1) дослідження кількості відмов за певний період часу (вимір часу “в режимі настінного годинника” або вимір часу відносно виконання процесів пристроями комп’ютера);

2) дослідження часових проміжків між помилками.

Вважали, що моделі, класифіковані цим способом, взаємно не перетинаються і можуть містити часткові випадки в кожному дослідженні [8]. Однією з перших складних моделей надійності ПЗ вважають модель Муси і Окумото [9]. Для побудови цієї моделі використано набір атрибутів, зокрема:

- часовий проміжок;
- загальну кількість відмов, які можливо виявити за нескінченний або скінченний проміжок часу;
- розподіл кількості відмов, які відбулися за час t (розподіл Пуассона або біноміальний розподіл);
- клас відмови, або функціональна форма активності відмов за певний час (застосовується тільки для випробувань із скінченними часовими проміжками);
- тип відмови, або вигляд функції інтенсивності відмов протягом нескінченного проміжку часу (застосовується тільки для випробувань із нескінченними часовими проміжками) [9].

З розвитком технологій розроблення і практичного застосування ПЗ схема класифікації моделей надійності ПЗ значно розширилася. Не тільки розроблено нові критерії класифікації, але й ускладнено саму структуру класифікації внаслідок об'єднання і перетину декількох критеріїв у різних моделях [10].

До найбільш вживаних методик і факторів надійності ПЗ належать: стадія життєвого циклу розроблення ПЗ (класифікація моделей залежить від етапу, на якому розраховується надійність ПЗ), можливість раннього прогнозування помилок, орієнтованість на інформацію або архітектуру (класифікацію роблять на підставі перевірки правильності вхідних / вихідних даних, або перевірки функціонального наповнення ПЗ), зростання надійності ПЗ в процесі виявлення та виправлення помилок тощо [11]. Класифікація за такими факторами [12] є найповнішою та дає можливість представити не тільки самі моделі, але і взаємозв'язок між ними (рис. 1).

Кількість моделей надійності ПЗ сьогодні перевищує сотню і продовжує зростати [6]. Залежно від визначення поняття надійності ПЗ і глибини дослідження обирають різні критерії і характеристики моделей надійності. Тому наведена на рис. 1 схема класифікації моделей надійності ПЗ, як і розглянуті критерії, не є єдиною [13].

Схема, наведена на рис. 1, поділяє всі моделі надійності ПЗ на аналітичні та емпіричні. Емпіричні моделі надійності, своєю чергою, поділяються на моделі складності і моделі, що визначають час, необхідний на "доведення" програми. Аналітичні моделі надійності ПЗ поділяють на динамічні (дискретні і неперервні) і статичні (за областю помилок, за областю даних) [6].

Життєвий цикл ПЗ загалом зазвичай поділяють на такі етапи [14]: специфікація вимог, проектування, кодування, тестування, експлуатація і супровід. Етап проектування може включати попереднє проектування і деталізоване проектування. Етап тестування може містити модульне, інтеграційне та регресійне тестування, тестування в умовах експлуатації. Етап супроводу може містити один або два цикли, кожен з яких має всі етапи стадії розробки. Згідно з ДСТУ 3918-1999 (ISO/IEC 12207:1995) [14] основними процесами життєвого циклу ПЗ є процеси: замовлення, постачання, розроблення, експлуатації та супроводу. Своєю чергою, процес розроблення ПЗ містить такі дії: реалізацію процесу, аналіз системних вимог, проектування архітектури системи, аналіз вимог до ПЗ, проектування архітектури ПЗ, розроблення детального проекту ПЗ, кодування і тестування ПЗ, інтеграцію ПЗ, кваліфікаційне тестування ПЗ, системну інтеграцію, кваліфікаційне тестування системи, встановлення ПЗ та забезпечення приймання ПЗ [14].

На ранніх стадіях життєвого циклу ПЗ потрібна модель прогнозування надійності, оскільки даних про відмови немає. Моделі такого типу призначені для передбачення кількості помилок у програмі перед тестуванням, і в деяких літературних джерелах [15] належать до детерміністичних (статичних) моделей надійності ПЗ. На етапі тестування показники надійності ПЗ покращуються завдяки відлагодженню програми. Модель зростання надійності на цьому етапі потрібна для оцінювання поточного рівня надійності, часу і ресурсів, потрібних для досягнення заданого рівня надійності ПЗ. Впродовж цього етапу оцінка надійності ґрунтується на аналізі цих відмов. Моделі

такого типу належать [15] до імовірнісних (динамічних) моделей надійності ПЗ. Після введення програми в експлуатацію при визначенні її надійності необхідно враховувати додавання нових модулів, усунення старих модулів, усунення виявлених помилок, поєднання нового коду з попередньо написаним кодом, зміну середовища користувача, зміну апаратного забезпечення тощо. На цьому етапі використовують еволюційні моделі надійності [16].

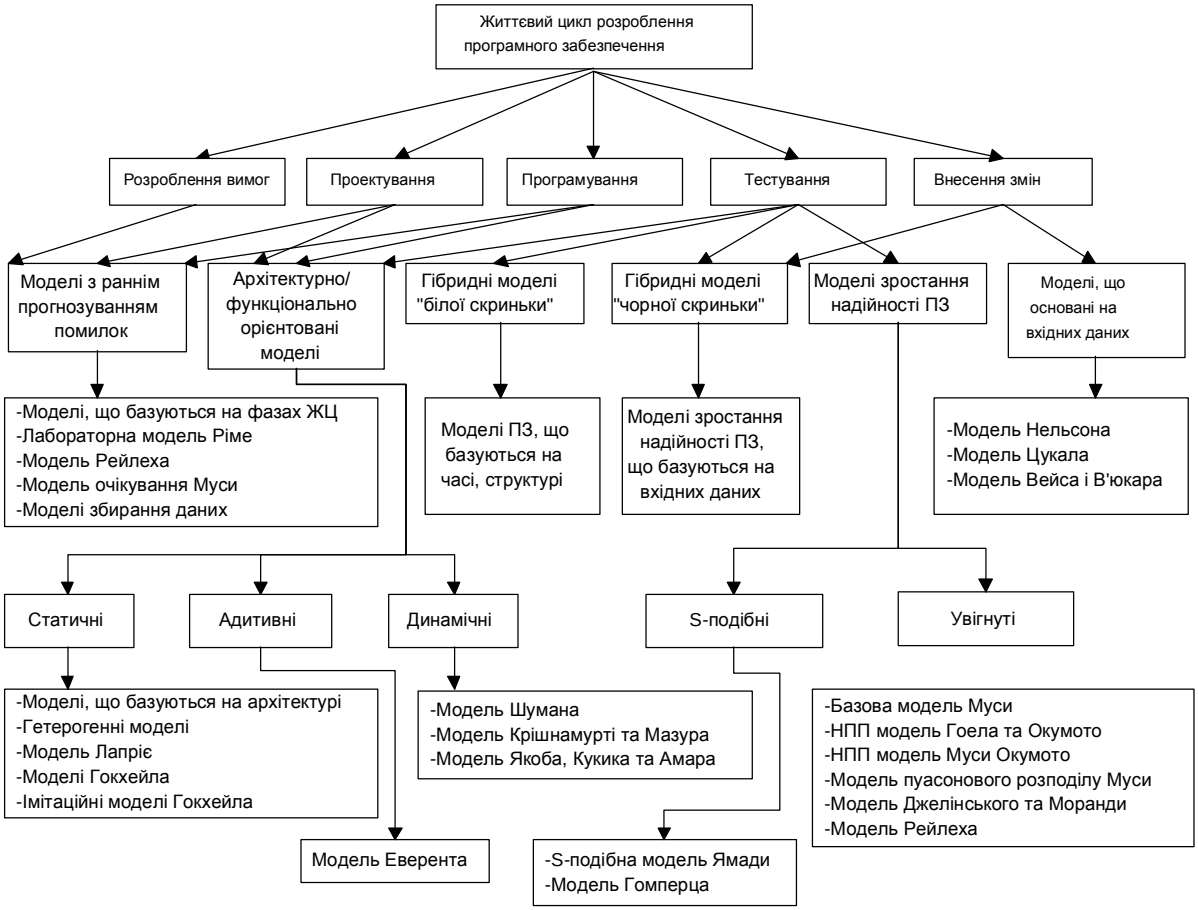

Рис. 1. Класифікація моделей надійності ПЗ та їх взаємозв'язок з життєвим циклом ПЗ

За моделями, в основу яких покладено підрахунок відмов (динамічні моделі), припускають, що концептуально в програмі наявна скінченна кількість помилок. Враховуючи, що кількість помилок є цілим числом, за динамічними моделями обчислюють кількість початкових помилок на етапі відлагодження програми і кількість помилок, що залишились під час чи в кінці 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 [15–17]. Як тільки інтенсивність відмов, пов'язана з помилками певного типу, визначено, інтенсивність відмов програми загалом обчислюють як добуток кількості помилок, що залишились в програмі, на інтенсивність відмов, породжених помилкою кожного типу [16]. Під час етапу відлагодження кількість помилок, що залишились, змінюється. Одним із способів моделювання такого процесу відмов є представлення кількості помилок, що залишились, як стохастичного процесу [16, 17].

Оцінка надійності ПЗ є все важливішою під час розроблення та тестування нових програмних продуктів. Перед тим, як нове ПЗ випустити в користування, його ретельно перевіряють на наявність помилок, які могли з'явитися в процесі розроблення ПЗ. Незважаючи на те, що виявлені помилки негайно видаляються, нові помилки можуть бути введені під час процесу відлагодження. Програмне забезпечення, яке містить помилки і випускається на ринок, несе високі ризики,

пов’язані з відмовами. Відлагодження і тестування, з іншого боку, зменшує кількість помилок, але збільшує витрати на розроблення. Отже, існує необхідність визначити оптимальний час для зупинки тестування ПЗ. Під час тестування системи надійність є важливим критерієм при ухваленні рішення, коли саме потрібно випустити ПЗ. Декілька інших критеріїв, такі як кількість помилок, що залишилися, частота відмов, вимоги до надійності або загальна вартість системи можуть бути використані для визначення оптимального часу тестування ПЗ [18].

У дослідженнях в області інженерії надійності ПЗ широко використовують ряд моделей зростання надійності ПЗ на основі неоднорідного процесу Пуассона, які запропоновано для оцінювання надійності ПЗ. Насправді моделі надійності ПЗ на основі неоднорідного процесу Пуассона стали доволі успішними практичними інструментами аналізу надійності ПЗ. У цих моделях розглядають процес відлагодження ПЗ, який можна охарактеризувати цільовою функцією. Надійність ПЗ можна оцінити як визначену цільову функцію. Параметри моделі, як правило, оцінюють з використанням методу максимальної правдоподібності або методу регресії. Різні моделі цього класу було побудовано на основі різних припущень.

Модель Гоеля–Окумото [18] також відома як експоненційна модель зростання надійності ПЗ, що ґрунтується на неоднорідному процесі Пуассона. Ця модель ґрунтується на таких припущеннях:

1) усі відмови в програмі є незалежними;

2) кількість відмов, виявлених у будь-який час, пропорційна поточній кількості помилок у програмі; це означає, що ймовірність виявлення кожної помилки є сталою величиною;

3) виявлені відмови видаляються до наступного тестування;

4) кожного разу, як відбувається відмова ПЗ, помилка, яка спричинила цю відмову, негайно виправляється, і жодних нових помилок до системи не вноситься [18].

Поведінка відмов ПЗ з часом у цій моделі описується таким диференційним рівнянням:

$$\frac{\partial m(t)}{\partial t} = b[a - m(t)], \tag{1}$$

де $m(\mathrm{t})$ – кількість відмов протягом часу t; a – очікувана загальна кількість помилок, що існує в ПЗ перед тестуванням; b – коефіцієнт виявлення відмов.

Муса у 1985 році [18], розглядаючи взаємозв’язки між часом виконання та календарним часом, запропонував модель, подібну до моделі Гоеля–Окумото. Нехай $m(\mathrm{t})$ – це кількість відмов, виявлених в результаті запуску тестів за певний час. Муса [18] запропонував таке диференційне рівняння:

$$\frac{\partial m(t)}{\partial t} = \frac{c}{nT}[\mathrm{a} - \mathrm{m}(t)], \tag{2}$$

де a – кількість помилок в програмі; c – фактор компресії тестування; T – час до відмови на початку тестування; n – загальна кількість відмов, можливих за час існування програми; t – час виконання або процесорний час, який затрачено на виконання тестів за час дослідження [18].

Охба [18] запропонував гіперекспоненційну модель зростання надійності ПЗ. Ця модель базується на припущенні про те, що ПЗ містить певну кількість кластерів модулів, кожен з яких містить різну початкову кількість помилок, і кожен з них має свій коефіцієнт відмов. Прикладами модулів можуть бути нові модулі та повторно використані модулі, прості та складні модулі, модулі, які взаємодіють з апаратним забезпеченням та ті, що не взаємодіють з ним тощо. Варто зауважити, що гіперекспоненційний розподіл є сумою експоненційних розподілів [18].

Подібне розширення експоненційної моделі зростання надійності ПЗ запропонували Ямада та Осакі [18], які поділяють ПЗ на k модулів. В цій моделі вважають, що інтенсивність відмов у межах різних модулів є різною, а інтенсивність відмов у межах однакових модулів є однаковою. Очікувану кількість відмов, виявлених для кожного модуля, вважають експоненційною. В такому випадку очікувану кількість відмов усього ПЗ можна отримати так:

$$m(\mathrm{t}) = a\sum_{i=1}^{k} p_i[1 - \mathrm{e}^{-b_i t}], \tag{3}$$

де k – кількість модулів у програмній системі; b_i – коефіцієнт виявлення однієї відмови у i-му модулі; p_i – ймовірність відмов для i-го модуля; a – очікувана кількість відмов ПЗ, виявлена у результаті, або загальна кількість помилок, наявна в ПЗ перед тестуванням.

Ще одним типом моделей надійності ПЗ на основі неоднорідного процесу Пуассона є S-подібні моделі. В цих моделях крива зростання надійності є S-подібною. Це означає, що крива перетинає експоненційну криву знизу, і перетинає лише один раз. Коефіцієнт виявлення відмов, який залежить від часу, є найбільшим у той час, коли тестування почалось, після чого він спадає експоненційно. Інакше кажучи, деякі помилки є екрановані іншими помилками на початку фази тестування, і поки ці помилки не виправляться, приховані помилки так і залишаться невиявленими [18]. Ямада також зазначив, що процес тестування програмного забезпечення часто містить процес навчання, під час якого тестувальники ознайомлюються з ПЗ, середовищами та вимогами до програмної системи [18]. S-подібні моделі у вигляді неоднорідного процесу Пуассона ґрунтуються на таких припущеннях:

1) коефіцієнт виявлення відмов відрізняється між відмовами;

2) кожного разу, як виникає відмова, помилка в програмному забезпеченні, яка спричинила відмову, негайно видаляється, і нові помилки в систему не вносяться [18].

Моделі надійності програмного забезпечення з урахуванням недосконалого відлагодження

В основу більшості вищеописаних моделей надійності ПЗ покладено припущення про ідеальне відлагодження, коли під час процесу тестування виявляється відмова ПЗ, знаходиться відповідна помилка та виправляється, але при цьому нові помилки не вносяться. Однак, це припущення є спрощенням реального процесу розроблення програмних систем, оскільки в процесі розроблення не завжди вдається видалити помилку із системи, під час видалення однієї помилки можуть бути внесені інші, відмова може бути викликана кількома взаємопов'язаними помилками тощо.

Загалом моделі недосконалою відлагодження будуються на основі таких припущень:

1) коефіцієнт виявлення відмов відрізняється між відмовами;

2) кожного разу, як виникає відмова ПЗ, помилка в ПЗ, яка спричиняє цю відмову, виправляється, і нові помилки можуть бути внесені в систему.

Концепція недосконалого відлагодження була вперше представлена в моделях надійності ПЗ Гоеля. Він у своїй роботі [17] описав це як одне з найважливіших практичних обмежень існуючих моделей. Гоель та Окумото запропонували модель недосконалого відлагодження, яка є розширенням існуючої моделі Джелінскі–Моранди. В цій моделі кількість відмов у системі на час $t, X(\mathrm{t})$ розглядається як марковський процес, де ймовірність переходів між станами системи визначається ймовірністю недосконалого відлагодження. Час між переходами $X(\mathrm{t})$ має експоненційний розподіл із значеннями, залежними від кількості помилок у системі. Функція відмов на інтервалі між (i-1)-ю та i-ю відмовами визначається виразом:

$$Z(\mathrm{t}_i) = [\mathrm{N} - p(\mathrm{i}-1)]\,l\,, \tag{4}$$

де N – початкова кількість помилок в системі; p – ймовірність недосконалого відлагодження; l – інтенсивність відмов.

Модель Фама–Нордмана–Жанга містить явище недосконалого відлагодження, припускаючи, що помилки можуть бути внесені під час фази відлагодження. У цій моделі припускають, що коефіцієнт внесення помилок є лінійною функцією, залежною від часу, а функція виявлення відмов є незростаючою та залежною від часу [19]. Цільова функція має вигляд:

$$m(\mathrm{t}) = \frac{a(1-\mathrm{e}^{-bt})(1-\frac{a}{b}) + a\ \mathrm{at}}{1+b\,e^{-bt}}\,, \tag{5}$$

де $a(t)$ – коефіцієнт внесення помилок під час фази відлагодження; $a(t)$ – очікувана кількість початкових помилок; $b(t)$ – коефіцієнт виявлення помилок.

Модель Фама–Жанга [20] припускає, що коефіцієнт внесення помилок є експоненційною функцією часу тестування, а функція виявлення помилок є незростаючою. Цільова функція набуває такої форми:

$$m(t) = \frac{1}{(1+b\ e^{-bt})}((c+a)(1-e^{-bt}) - \frac{ab}{b-a}(e^{-at} - e^{-bt})), \quad (6)$$

де $a(t)$ – коефіцієнт внесення помилок під час фази відлагодження; $a(t)$ – очікувана кількість початкових помилок; $b(t)$ – коефіцієнт виявлення помилок.

У моделі недосконалого відлагодження Ямади [21] припускають, що функція кількості помилок є екпоненційною, а функція виявлення відмов – сталою. Цільова функція представлена виразом:

$$m(t) = \frac{ab(e^{at} - e^{-bt})}{a+b}, \quad (7)$$

де $a(t)$ – коефіцієнт внесення помилок під час фази відлагодження; $a(t)$ – очікувана кількість початкових помилок; $b(t)$ – коефіцієнт виявлення помилок.

Інша модель недосконалого відлагодження Ямади [21] розглядає незмінну функцію внесення помилок та сталу функцію виявлення відмов. Цільова функція має вигляд:

$$m(t) = a(1-e^{-bt})(1-\frac{a}{b}) + a\ at, \quad (8)$$

де $a(t)$ – коефіцієнт внесення помилок під час фази відлагодження; $a(t)$ – очікувана кількість початкових помилок; $b(t)$ – коефіцієнт виявлення помилок.

Капур та Гарг [22] представили процес недосконалого відлагодження в моделі Гоеля–Окумото. Вони припустили, що відношення кількості видалених помилок до кількості помилок, що залишились в системі, зменшується через ефект недосконалого відлагодження. Чанг та Лью [23] запропонували модель негауссового простору станів для формулювання ефекту недосконалого відлагодження в надійності ПЗ. Шир [24] розробив модель зростання надійності ПЗ, що містить ефект недосконалого відлагодження та проблему розладнання. Капур та ін. [25] представили дискретну модель зростання надійності ПЗ з логістичним коефіцієнтом видалення помилок та концепцію двох типів недосконалого відлагодження. Прасад та ін. [26] використовували недосконале відлагодження та проблему розладнання в своїй моделі зростання надійності ПЗ, що базувалась на добре відомому експоненційному розподілі.

Фам та Нордман [19] сформулювали узагальнену модель надійності програмного забезпечення на основі неоднорідного пуассонового процесу та вивели аналітичний вираз для цільової функції. Моделі надійності ПЗ на основі неоднорідного пуассонового процесу з урахуванням недосконалого відлагодження ґрунтуються на таких припущеннях:

1) під час видалення помилки, що стала причиною відмови, існує ймовірність внесення нових помилок;

2) ймовірність виявлення помилки в програмі пропорційна кількості помилок, що залишились у системі.

Моделі надійності програмного забезпечення з урахуванням розподілу зусиль тестування

Для кращого опису поведінки надійності ПЗ використовують моделі надійності з використанням функцій зусиль тестування [27]. Зусилля тестування визначаються як людино-години, кількість тестових випадків, час роботи процесора, які були затрачені під час фази тестування, а їх розподіл в часі визначається виглядом функції, використаної в певній моделі надійності. При цьому що більше зусиль тестування було витрачено під час розроблення

програмного засобу, то менша кількість помилок залишається в коді програми і, відповідно, меншою є інтенсивність відмов ПЗ.

Найчастіше в моделях надійності ПЗ використовують такі функції розподілу зусиль тестування: постійна, експоненційна, логістична, функції розподілів Релея та Вейбулла [28–30]. Експоненційний та релеїв розподіли можна розглядати як часткові випадки розподілу Вейбулла.

Постійна функція. За цією функцією розглядають зусилля тестування як сталу величину. У класичних моделях зростання надійності ПЗ дослідники вважають, що зусилля тестування програмного забезпечення є константою:

$$w(t)=w_0, \tag{9}$$

Кумулятивні зусилля тестування можна отримати за виразом:

$$W(t)=wt, \tag{10}$$

Як видно з (10), загальні зусилля тестування прямують до безмежності. У випадку, якщо функція зусиль тестування не розглядається, можна вважати, що використовується постійна функція з одиничним значенням:

$$w(t)=1. \tag{11}$$

Розподіл Вейбулла. У випадках, коли розподіл зусиль тестування погано описується експоненційною кривою чи кривою Релея, використовують функцію розподілу Вейбулла, яка має вигляд [30]:

$$w(t)=N\beta m t^{m-1} e^{-\beta t^m}, \tag{12}$$

де N – очікувана загальна кількість зусиль тестування, необхідна для тестування ПЗ; β, m – параметри масштабу та форми відповідно. Значення m>3 не використовуються при описі розподілу зусиль тестування ПЗ, оскільки в цьому випадку форма кривої не відповідає реальним процесам розроблення ПЗ.

Кумулятивні зусилля тестування в цьому випадку записують як:

$$W(t)=N(1-e^{-\beta t^m}). \tag{13}$$

Експоненційний розподіл. Така крива є частковим випадком кривої Вейбулла при m=1. Експоненційна крива є зручною для опису середовища тестування, в якому показник зусиль тестування монотонно знижується [31].

Експоненційну функцію зусиль тестування описують виразом:

$$w(t)=N\beta e^{-\beta t}, \tag{14}$$

а кумулятивну функцію зусиль тестування – виразом

$$W(t)=N(1-e^{-\beta t}), \tag{15}$$

де N – очікувана загальна кількість зусиль тестування, необхідна для тестування ПЗ; β – параметр масштабу.

Розподіл Релея. Крива Релея є частковим випадком функції розподілу зусиль тестування Вейбулла при m=2. Функція зусиль тестування Релея спочатку збільшується, досягаючи свого піка, і потім спадає, зменшуючи зусилля тестування асимптотично до нуля [30]:

$$w(t)=N\beta t e^{\frac{-\beta}{2}t^2}. \tag{16}$$

Кумулятивні зусилля тестування в такому випадку визначаються як:

$$W(t)=N(1-e^{\frac{-\beta}{2}t^2}). \tag{17}$$

У (16) і (17) N – очікувана загальна кількість зусиль тестування, необхідна для тестування ПЗ; β – параметр масштабу.

Логістичний розподіл. Логістичну криву вперше запропоновано в [32] як альтернативу кривій Релея. Вони мають подібну поведінку, окрім початкової стадії процесу тестування. Логістичну криву зусиль тестування записують як:

$$w(t)=\frac{NA\eta}{(e^{\frac{\eta t}{2}}+Ae^{\frac{-\eta t}{2}})^2}, \quad (18)$$

де A – параметр-константа; η – міра споживання затрат на тестування; N – кумулятивні зусилля тестування.

Кумулятивні зусилля тестування у випадку логістичного розподілу мають вигляд:

$$W(t)=\frac{N}{1+Ae^{-\eta t}}. \quad (19)$$

Логарифм логістичної функції зусиль тестування. Бохарі та Ахмад [32] запропонували логарифм логістичної функції зусиль тестування для прогнозування поведінки помилок та відмов ПЗ. Вони показали, що логарифм логістичної функції зусиль тестування добре підходить і є достатньо гнучким для оцінювання надійності програмних продуктів різних класів [0].

Вигляд логарифма логістичної функції розподілу зусиль тестування є таким:

$$w(t)=\frac{N(\frac{t}{\lambda})^{-\beta}\beta}{(1+(\frac{t}{\lambda})^{-\beta})^2 t}, \quad (20)$$

де N – кумулятивні зусилля тестування; $l>0$ – параметр масштабу; $b>0$ – параметр форми.

Функція кумулятивних зусиль тестування в такому випадку має вигляд:

$$W(t)=\frac{N}{1+(\frac{t}{\lambda})^{-\beta}}. \quad (21)$$

Найпростіша модель зростання надійності ПЗ з урахуванням недосконалого відлагодження, що використовує функцію розподілу зусиль тестування, повинна відповідати таким критеріям:

- процес видалення помилок розглядається як неоднорідний процес Пуассона;
- ПЗ є суб'єктом, в якому у випадковий час може статись відмова, спричинена проявом помилок, які залишились у системі;
- кількість помилок, виявлених на проміжку часу $(t,t+\Delta t)$ при поточних затратах на тестування, є пропорційною до кількості помилок, що залишились в системі;
- коефіцієнт пропорційності є однорідним і не залежить від часу;
- поведінка зусиль тестування залежить від часу і може бути змодельована логістичним розподілом;
- кожного разу, коли виникає відмова, помилка, що її спричинила, негайно і коректно видаляється, при цьому жодних нових помилок в систему не вноситься;
- виправлення помилок займає незначний час і виявлені помилки гарантовано видаляються [33].

Отже, модель зростання надійності ПЗ, що містить функцію зусиль тестування, можна описати таким виразом:

$$\frac{dm(t)}{dt}\frac{1}{w(t)}=r[a-m(t)], \quad (22)$$

де $w(t)$ – функція зусиль тестування; r – коефіцієнт виявлення відмов; a – коефіцієнт затрат зусиль тестування в логістичній функції; $m(t)$ – цільова функція, що обчислює очікувану кількість програмних відмов на час t.

Вираз (22) містить дві компоненти, що впливають на кількість виявлених помилок: функцію зусиль тестування $w(t)$ та коефіцієнт виявлення відмов r. Розв'язуючи диференційне рівняння (22) з граничною умовою $m(0)=0$, отримаємо:

$$m(t)=a(1-\exp[-r(W(t)-W(0))])=a(1-\exp[-rW(t)]), \quad (23)$$

де $W(t)$ – кумулятивні зусилля тестування (наприклад, процесорні години, об'єм тестових випадків, людино-години та ін.) на час t.

Вираз (23) є моделлю на основі неоднорідного процесу Пуассона з цільовою функцією, яка враховує затрати на зусилля тестування. Затрачені зусилля тестування вказують, наскільки ефективно помилки виявляються в ПЗ. Тому ця функція відіграє важливу роль в моделюванні надійності ПЗ і може бути описана розподілами різного типу [33].

Експоненційна модель Ямади [34] враховує зусилля тестування та містить експоненційну функцію зусиль тестування. Ця модель є увігнутою, а цільова функція має вигляд:

$$m(\mathrm{t}) = \mathrm{a}(1 - \mathrm{e}^{-ra\,(1-\mathrm{e}^{-bt})}). \quad (24)$$

Модель Ямади–Релея [34] враховує зусилля тестування та містить релеєву функцію зусиль тестування. Ця модель належить до S-подібних, цільова функція має вигляд:

$$m(\mathrm{t}) = \mathrm{a}(1 - \mathrm{e}^{-ra(1-\mathrm{e}^{(\frac{bt^2}{2})})}). \quad (25)$$

Раджу [35] дослідив інтеграцію логарифм-логістичної функції зусиль тестування в увігнуту S-подібну модель зростання надійності ПЗ на основі неоднорідного процесу Пуассона для кращого опису процесу виявлення відмов при недосконалому відлагодженні. Ямада та ін. [36] модифікували модель Гоеля–Окумото та об'єднали її з концепцією зусиль тестування. Так вони створили модель зростання надійності ПЗ, на основі неоднорідного процесу Пуассона для кращого опису процесу відмов ПЗ. Хуанг [33] запропонував нову модель зростання надійності ПЗ що використовує логістичну функцію зусиль тестування. Логістична функція зусиль тестування – це крива оцінювання розробки та зусиль тестування ПЗ з хорошими характеристиками прогнозування. В роботі [33] Хуанг узагальнив логістичну функцію зусиль тестування. Узагальнена функція зусиль тестування адекватніше описує процес розроблення ПЗ, тому її використання є більш достовірним для опису розподілу затрат на тестування ПЗ.

Висновки

У роботі наведено огляд і результати аналізу найпоширеніших моделей надійності ПЗ. Показано, що на ранніх стадіях життєвого циклу ПЗ потрібні моделі прогнозування надійності, оскільки на цих етапах даних про відмови немає. Моделі зростання надійності використовують на етапі тестування ПЗ. Після введення програми в експлуатацію при визначенні її надійності враховують додавання нових та усунення застарілих програмних модулів, виправлення виявлених помилок тощо. На цьому етапі використовують еволюційні моделі надійності. Тобто на різних етапах життєвого циклу ПЗ використовують різні типи моделей надійності ПЗ, кожен з яких, крім того, містить значну кількість моделей, побудованих з урахуванням різноманітних припущень про методології та технології розроблення ПЗ, поведінку його відмов тощо. Все це значно ускладнює створення єдиної узагальненої класифікації моделей надійності ПЗ, і це завдання сьогодні все ще потребує свого вирішення.

Для кращого опису поведінки надійності ПЗ використовують моделі надійності з використанням функцій зусиль тестування, які визначаються як людино-години, кількість тестових випадків, час роботи процесора, які були затрачені під час фази тестування. При цьому що більше зусиль тестування було витрачено під час розроблення програмного засобу, то менша кількість помилок залишається в коді програми і, відповідно, меншою є інтенсивність відмов ПЗ. Розглянуто найпоширеніші функції розподілу зусиль тестування, такі як постійна, експоненційна, логістична, функції розподілів Релея та Вейбулла. Наведено приклади моделей надійності ПЗ, які містять розподіл зусиль тестування. Показано, що такі моделі адекватніше описують інтенсивність відмов сучасного складного і масштабного ПЗ.

В основу більшості моделей надійності ПЗ покладено припущення про ідеальне відлагодження, коли під час процесу тестування виявляють відмову ПЗ, знаходять відповідну помилку та виправляють її, але при цьому нові помилки не вносяться. Однак це припущення є спрощенням реального процесу розроблення програмних систем, оскільки в процесі розроблення не завжди вдається видалити помилку із системи, під час видалення однієї помилки можуть бути внесені інші, відмова може бути спричинена кількома взаємопов'язаними помилками тощо.

Розглянуто найвідоміші моделі надійності ПЗ, в яких враховано явище недосконалого відлагодження. На жаль, як в практичних, так і в теоретичних дослідженнях надійності ПЗ явище недосконалого відлагодження використовують недостатньо широко. З урахуванням ускладнення технологій та засобів розроблення програмних продуктів та зростанням складності сучасних програмних систем неврахування недосконалого відлагодження під час аналізу надійності ПЗ призводить до того, що моделі надійності стають все менш адекватними реальному об'єкту дослідження, а результати, отримані з їх використанням, перестають бути достовірними.

1. Фадин А. А. и др. / под ред. А. С. Маркова и В. Л. Цирлова. – М.: Сводный отчёт по безопасности программного обеспечения в России и мире за 2010 год // НПО "Эшелон". – 2011. – 34 с. 2. Смагин В. А. Основы теории надежности программного обеспечения. – СПб.: ВКА им. А. Ф. Можайского, 2009. – 336 с. 3. Черкесов Г. Н. Надежность аппаратно-программных комплексов. – СПб.: "Питер", 2005. – 479 с. 4. William F. Software reliability modeling survey – Naval Surface Warfare Center, 1996. 5. Luy M. R. Handbook of sotware reliability engineering - IEEE Computer Society Press, 1996. 6. Ch. Ali Asad, Muhammad Irfan Ullah, Muhammad Jaffar-Ur Rechman An approach for software reliability model selection – IEEE Computer Society Press, 2004. 7. Shooman M. L. Operational Testing and Software Reliability Estimation During Program Developments – IEEE Computer Society, 1973. 8. Coutinho J. deS. Software Reliability Growth – IEEE Symposium on Computer Software Reliability, 1973. 9. Musa J. D., Okumoto, K. Software Reliability Models: Concepts, Classification, Comparisons, and Practice – Electronic Systems Effectiveness and Life Cycle Costing, 2000. 10. Moranda P. L., Jelinski Z. Final Report on Software Reliability Study – McDonnell Douglas Astronautics Company, 1972. 11. Гуда А. Н., Калинин Т. С., Чернов А. В. Реализация надежного программного обеспечения задач технической диагностики информационно-управляющих систем // Известия высших учебных заведений. Северо-Кавказский регион. Технические науки. № 4. – 2011. – С. 26–31. 12. Чернов А. В., Параценко И. Г. Классификация моделей надежности программного обеспечения // Электронный научный журнал "Инженерный вестник Дона", 2007–2015. 13. Белявский Г. И., Чернов А. В. Математические модели линейных контролируемых дискретных динамических систем // Научно-технические ведомости Санкт-Петербургского государственного политехнического университета. – 2009. – № 2. – С. 145–151. 14. ДСТУ 3918-99 (ISO/IEC 12207: 1995) Інформаційні технології. Процеси життєвого циклу програмного забезпечення. – К.: Держстандарт України, 2002. – 49 с. 15. Cobra Rahmani. Exploitation of Quantitative Approaches to Software Reliability / Cobra Rahmani, Azad Azadmanesh // University of Nebraska at Omaha, 2008. – 32 p. 16. 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. 17. A. L. Goel, Software Reliability Models: Assumptions, Limitations, and Applicability, // IEEE Trans. Software Eng., vol. 11, pp. 1411–1423, 1985. 18. Pham H. System software reliability // Springer-Verlag ondon Limited, 2006. – 440 p. 19. H. Pham, L. Nordmann, and X. Zhang, A general imperfect software debugging model with s-shaped fault detection rate // IEEE Trans. Reliability, vol. 48, pp. 169–175, June 1999. 20. H. Pham and X. Zhang, "An NHPP software reliability models and its comparison" // International J. Of Reliability, Quality and Safety Engineering, vol. 14, no. 3, pp. 269–282, 1997. 21. S. Yamada, K. Tokuno, and S. Osaki, "Imperfect debugging models with fault introduction rate for software reliability assessment," // International J. Syst. Science, vol. 23, no. 12, 1992. 22. Kapur, P. K., and Garg, R. B. 1990. Optimal release policy for software reliability growth models under imperfect debugging. Oper. Res. RAIRO. 24(3), 295–305. 23. Chang, Y. C., and Liu, C. T. 2009. A generalized JM model with applications to imperfect debugging in software reliability // Appl. Math. Model. 33, 3578–3588. 24. Shyur, H.J. 2003. A stochastic software reliability model with imperfect-debugging and change-point. // J. Syst. Software. 66(2), 135–141. 25. Kapur, P. K., Singh, O. M. P., Shatnawi, O., and Gupta, A. 2006. A discrete NHPP model for software reliability growth with imperfect fault debugging and fault generation. // Int. J. Perform. Eng. 2(4), 351–368. 26. Prasad, R.S., Raju, O.N., and Kantam, R.R.L. 2010. SRGM with imperfect debugging by genetic algorithms. // Int. J. Software Eng. Appl. 1(2), 66–79. 27. Ce Zhang, Gang Cui, Hongwei Liu, Fanchao Meng, Shixiong Wu. "A Unified and Flexible Framework

of Imperfect Debugging Dependent SRGMs with TestingEffort" // Journal of Multimedia, Vol. 9, no. 2, FEBRUARY 2014. 28. Chin-Yu Huang. An Assessment of Testing-Effort Dependent Software Reliability Growth Models [Text]/ Chin-Yu Huang, Sy-Yen Kuo, Michael R. Lyu // IEEE Transactions on Reliability, Vol. 56, No. 2, June 2007. – C. 198–211. 29. Sy-Yen Kuo. Framework for Modeling Software Reliability, Using Various Testing-Efforts and Fault-Detection Rates [Text] / Sy-Yen Kuo, Chin-Yu Huang, Michael R. Lyu // IEEE Transactions on Reliability, Vol. 50, no. 3, September 2001. – C. 310–320. 30. Ahmad N. Software Reliability Growth Models with Log-logistic Testing-Effort Function: A Comparative Study [Text]/ N. Ahmad, Md. Zafar Imam // International Journal of Computer Applications (0975 – 8887) Vol. 75. – No.12, August 2013. – C. 6–10. 31. Yamada S. A testing-effort dependent software reliability model and its application [Text] / Shigeru Yamada, Hiroshi Ohtera, Hiroyuki Narihisa // Microelectron. Reliab, Vol. 27, No. 3, 1987. – C. 507–522. 32. Parr F. An Alternative to the Rayleigh Curve Model for Software Development Effort // IEEE Transactions on Software Engineering, Vol.6, no. 3, May 1980, pp. 291–296. 33. C. Y. Huang, M. R. Lyu, "Optimal Release Time for Software Systems Considering Cost, Testing-Effort, and Test Efficiency," // IEEE Trans. on Reliability, Vol. 54, No. 4, pp. 583–591, Dec. 2005. 34. H. Pham, "Software reliability and cost models: perspectives, comparison and practice" // European J. of Operational Research, vol. 149, pp.475–489, 2003. 35. Raju, O.N. 2011. Software reliability growth models for the safety critical software with imperfect debugging. // Int. J. Comput. Sci. Eng. 3(8), 3019–3026. 36. S. Yamada, J. Hishitani, and S. Osaki, "Software reliability growth model with Weibull testing effort: a model and application," // IEEE Trans. Rel., vol. R-42, pp. 100–105, 1993.

О. Кузьмін, В. Федека
Національний університет "Львівська політехніка",
кафедра автоматизованих систем управління

ІМІТАЦІЙНА МОДЕЛЬ МОНІТОРИНГУ НАВКОЛИШНЬОГО СЕРЕДОВИЩА БЕЗПРОВІДНОЮ СЕНСОРНОЮ МЕРЕЖЕЮ

Наведено імітаційну модель та описано відповідний програмний продукт, що дає змогу моделювати навколишнє середовище в умовах надзвичайної ситуації (аварії на хімічно небезпечному об'єкті) або вільного спостереження (без наперед визначеної процедури проведення). Збирають та обробляють інформацію за допомогою безпровідної сенсорної мережі.

Ключові слова: модель, надзвичайна ситуація, інформація, сенсорна мережа.

The article presents a simulation model and described appropriate software, to simulate the environment in an emergency situation (accidents on chemically hazardous objects) or free observation (no pre-defined procedures). Collection and processing of information made using a wireless sensor network.

Key words: model, emergency situation, information, sensor network.

Вступ

Моніторингом називають багаторазове спостереження, вимірювання та оцінювання стану об'єкта. Для моніторингу навколишнього середовища створюють систему спостереження та контролю за природними процесами з метою оцінювання їх стану та прогнозування неминучих змін. Оцінюючи стан довкілля, можна скласти уявлення про масштаби його забрудненості різними речовинами і вжити відповідних заходів щодо усунення цього забруднення. Саме важливість цього