ЯКІСТЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

УДК 629.7.05(004.05)

Манжос Ю.С.
Національний аерокосмічний університет ім. М.Є. Жуковського «ХАІ»

АЛГЕБРИЧНЕ АНАЛІЗУВАННЯ НАДІЙНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розглянуто засади класичного підходу до оцінювання ймовірності безпомилкової роботи програмних систем. Запропоновано модель програмної системи, що складається з програмного коду, як чорної скрині, та системи відображень множин елементів зовнішнього середовища та програмного коду. Розглянуто властивості відображень. Досліджено за допомогою алгебричних методів можливі джерела дефектів програмного забезпечення, обумовлені порушенням інтервальних відношень, та визначено необхідні умови відсутності дефектів. Визначена комбінаційна оцінка ймовірності безпомилкової роботи програмної системи.

Рассмотрены основы классического подхода к оцениванию вероятности безошибочной работы программных систем. Предложена модель программной системы, включающая программное обеспечение как черный ящик и систему отображений множеств элементов внешней среды и программного обеспечения. Рассмотрены свойства отображений. С помощью алгебраических методов исследованы возможные источники дефектов программного обеспечения, обусловленные нарушением интервальных соотношений. Сформулированы необходимые условия отсутствия дефектов. Определена комбинационная оценка вероятности безошибочной работы программной системы.

The article covers the basics of the classical approach to the estimation of the probability of error-free operation of software systems.The paper proposes a model of a software system, including software as a black box and a system of maps of elements of the environment and software.The paper discusses the properties of those maps.Using algebraic methods the possible sources of software defects caused by violation of interval relations. The paper formulates the necessary conditions for the absence of defects.The combinational evaluation of the probability of error-free software was defined with help of set-theoretic approach

Ключові слова: *надійність, дефекти програмного забезпечення, ймовірність безпомилкової роботи*

Вступ

Наявність дефектів програмного забезпечення (ПЗ) обумовлює відмови різноманітних технічних систем (ТС), одними із представників яких є космічні апарати. Але не зважаючи на те, що джерела та природа дефектів програмного забезпечення (ДПЗ) різноманітні, більшість з них спричинені людським фактором. Існує підхід до оцінювання надійності програмних комплексів, за яким ймовірність безпомилкової роботи (БР) визначається через функцію інтенсивності помилок. Ймовірність БР ПЗ не є функцією часу, а має комбінаторну природу, обумовлену як вхідними даними, так і станом програми. Розгляданню цих питань і присвячена ця робота.

1. Постановка задачі досліджень

Як вже було відмічено, класичний підхід до оцінки надійності ПЗ ґрунтується на використанні математичного апарату, що є аналогічним тому, який використовується для оцінювання надійності технічних систем. Але ПЗ, на відміну від ТС, не властиве старіння та зношення, тому експлуатація ПЗ у тих самих умовах не може спричинити зменшення ймовірності БР, що визначається існуванням ДПЗ. Доведення комбінаційної природи ймовірності БР ПЗ вимагає виконання наступних завдань:

1. Проаналізувати класичний підхід до оцінки надійності ПЗ.
2. Проаналізувати джерела ДПЗ.
3. Оцінити ймовірність безпомилкової роботи ПЗ на підставі комбінаторного підходу.

Розглянемо більш детально ці питання

2. Класичне оцінювання надійності програмного забезпечення

Надалі під надійністю ПЗ будемо мати на увазі здатність ПЗ задовольняти вимогам користувача. Існує кілька показників надійності. Найважливіший з них – функція ймовірності БР $P(t)$ того, що на інтервалі $[0,t]$

не станеться помилок системи [1-3], що має зв'язок з функцією помилок: $F(t)=1-P(t)$.

Тоді щільність ймовірності помилок:

$$f(t)=\frac{dF(t)}{dt}=-\frac{dP(t)}{dt} \qquad (1)$$

А інтенсивність потоку помилок:

$$\lambda(t)=\frac{f(t)}{P(t)} \qquad (2)$$

– це умовна ймовірність прояву помилки на часовому інтервалі $[t,t+\Delta t]$, якщо до цього помилок не було.

На підставі (1), (2):

$$P(t)=\exp\left(-\int_0^t \lambda(\tau)d\tau\right) \qquad (3)$$

Визначення ймовірність БР програмної системи потребує знання інтенсивності $\lambda(t)$, яка визначається за результатами тестування.

Величина (3) має межу, яка дорівнює нулю:

$$\lim_{t\to\infty} P(t)=\lim_{t\to\infty}\exp\left(-\int_0^t \lambda(\tau)\right)=0$$

Тобто з часом ймовірність відмови ПЗ має зростає через помилки, спричинені залишковими ДПЗ, які були внесені на різних етапах життєвого циклу. Якщо після тестування було досягнуте $\lambda=10^{-4}\,\text{год}^{-1}$, то ймовірність БР буде $P(t)=\exp\left(t\cdot 10^{-4}\right)$, тобто ймовірність БР за 4 години складе 0.999, а на протязі 10 000 годин зменшиться до 0.1.

Таким чином ймовірність БР ПЗ за класичними розрахунками має монотонно зменшуватися з часом доти, доки не досягне нульового значення. Але, на відміну від ТС, ПЗ не може старитися та зношуватися через неможливістю появи нових дефектів програмного коду під час експлуатації. У той же час можливі дефекти, обумовлені некоректними даними, наприклад «політним завданням» або іншим масивом даних або массивом змінення програми, що керують конфігурацією вбудованого програмного забезпечення [4].

Відомо [1, 5], що у теорії випадкових процесів, що є базисом теорії надійності, найважливішу роль відіграє так званий найпростіший потік подій, який має такі властивості:

1.Ординарність, тобто неможливість «групової» появи подій;

2. Відсутність післядії, тобто кількість подій для будь-яких двох інтервалів часу які не перетинаються є незалежними випадковими величинами;

3. Стаціонарність, тобто всі ймовірнісні характеристики потоку подій не змінюються у часі.

Нажаль ПЗ не властива ординарність, тому, що можливі групові появи подій – відмов. Через взаємні зв'язки програмних модулів та можливість існування програмних дефектів у кожному з них, а також послідовну роботу окремих модулів для ПЗ не виконується умова відсутності післядії. Через те, що у загальному випадку окремі програмні модулі мають різну складність, розроблені різними людьми або навіть різними колективами цім модулям властиві різні ймовірнісні характеристики потоку подій-відмов програмного забезпечення. Тобто, якщо розглядати поведінку ПЗ у часі, то потік відмов не є стаціонарним.

Нарешті ПЗ не властива ергодичність, тобто можливість переходу з будь-якого стану у будь-який інший стан (навіть через ланцюжок інших станів). Таким чином використання апарату теорії ймовірності та випадкових процесів для визначення надійності ПЗ як функції часу не є обґрунтованим.

Розглянемо більш детально джерела програмних дефектів.

3. Джерела програмних дефектів

Сучасні технології розроблення ПЗ для систем критичного призначення, закріплені відповідними стандартами, що вимагають необхідність захисту ПЗ не тільки від некоректних даних, але й від неправильних дій операторів, нажаль не забезпечують повну відсутність залишкових ДПЗ. Це обумовлено неможливістю тестування всіх можливих обчислювальних шляхів, кожний з яких покриває різну підмножину коду [6].

Для спрощення подальшого аналізування позначимо множини всіх шляхів як $\Omega=\{\omega_q\}$, а множину всіх підмножин коду як $C=\{c_k\}$. Шляхи (тобто підмножини програмного коду) можуть перетинатися тобто $\exists a,b$

$\omega_a \cap \omega_b \neq \varnothing$, $c_a \cap c_b \neq \varnothing$. Залежність між шляхами та підмножинами програмного коду, який вони покривають, визначається відображенням $\omega_q \overset{W}{\rightarrow} c_q$ [7]. У свою чергу шляхи визначаються не тільки даними x, що надходять до програмної системи (ПС), але й її внутрішнім станом s, тобто $\omega_q = f(x,s)$. Існування функціональної залежності дозволяє стверджувати, про те, що відображення $\omega_q \overset{f(x,s)}{\rightarrow} c_q$ визначається як вхідними даними та і внутрішнім станом програмної системи.

Наявність програмних дефектів у одній з підмножин коду може позначитися на виконанні кількох шляхів виконання програми. Якщо будь-який з фрагментів коду виконується багато разів, то питома вага його дефектів залежить від кількості виконання фрагменту та інформаційних зв'язків з іншими фрагментами. Загальна інтенсивність потоку помилок, як функція часу, залежить також від швидкості виконання коду V: чим швидше виконуються команди тим вище ймовірність того, що буде виконано дефектний ланцюжок, який і може спричинити відмову. Якщо на вхід ПЗ надходять протестовані комбінації x,s, то інтенсивність потоку помилок буде нульовою і ймовірність безвідмовної роботи дорівнює одиниці (будемо вважати, що операційна система не має дефектів). Таким чином, інтенсивність потоку помилок слід визначати не як «чисту» функцію часу, а як функцію шляху. У зв'язку з тим, що шлях визначається через дані та внутрішній стан, то $\lambda = Vf_{xs}(x_q, s_r)$. Функція $f_{xs}(x_q, s_r) = 0$ для протестованих шляхів, інакше $f_{xs}(x_q, s_r) > 0$.

Розглянемо програмну систему, як «чорну» скриню, на вхід якої надходять вектор даних $X = \{x_1, x_2, \ldots, x\}$ та вектор внутрішнього стану $S = \{s_1, s_2, \ldots, s_k\}$. Виходом є $Y = \{y_1, y_2, \ldots, y_j\}$ – керуючий вектор, елементи якого використовуються для опосередкованого керування виконуючими органами космічного апарату та обчислюються за допомогою вектор-функції $F = \{f_1, f_2, \ldots, f_j\}$, що реалізує код (рис.1). Розмірність вектор-функції визначається розмірністю керуючого вектору $Y_j = F_j(x_1, \ldots, x_i, s_1, \ldots, s_k)$.

Для коректної роботи технічної системи необхідно, щоб елементи векторів X, Y належали певним інтервалам значень, обумовленим зовнішнім середовищем (ЗС). Чисельні значення елементів векторів можуть бути спотворені некоректними вимогами до ПЗ, або помилками розробників, спричинених використанням некоректних програмних типів даних та алгоритмічними або програмними дефектами. Вихід елементів X, Y за межі інтервалів може спричинити системні відмови.

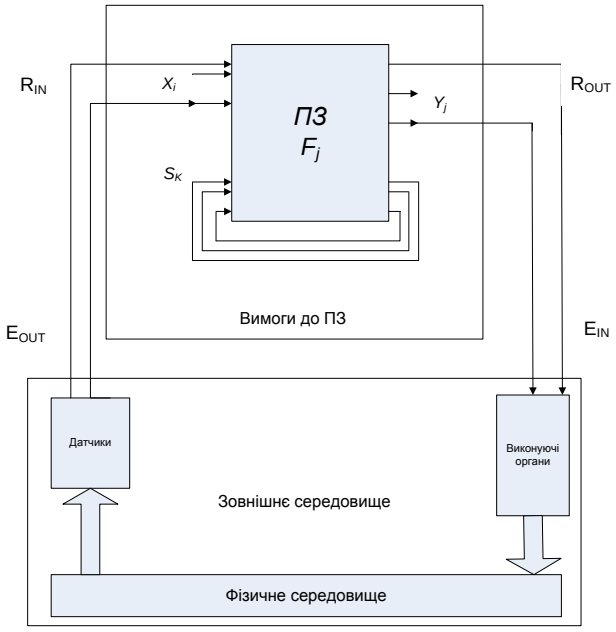

Рис. 1. Модель програмної системи

Позначимо множину можливих інтервалів значень параметрів, що виходить із зовнішнього середовища як $\{(\inf(E_{OUT_1}), \sup(E_{OUT_1}))\}$, а керуючу множину, що надходить до зовнішнього середовища, як $\{(\inf(E_{IN_m}), \sup(E_{IN_m}))\}$. Визначені вимогами множини інтервалів значення елементів даних позначимо як $\{(\inf(R_{IN_n}), \sup(R_{IN_n}))\}$, а елементів вектора керування керування як $\{(\inf(R_{OUT_p}), \sup(R_{OUT_p}))\}$. Надалі під вимогами будемо розуміти як вимоги, що визначив замовник(користувач), так і галузеві та нормативні вимоги, що визначаються відповідними стандартами. Кардинальності пар множин $\{E_{OUT_1}\}, \{R_{IN_n}\}$ та $\{R_{OUT_p}\}, \{E_{IN_m}\}$

можуть не збігатися через помилки у вимогах, а кардинальності пар множин $\{X_i\},\{R_{IN_n}\}$ та $\{R_{OUT_p}\},\{Y_j\}$ через програмні дефекти.

Множини можливих значень даних та керуючого вектора, що реалізовані програмним забезпеченням позначимо як $\{(\inf(X_i),\sup(X_i))\}$, $\{(\inf(Y_j),\sup(Y_j))\}$. У даному випадку мається на увазі, що ПЗ може керувати виконуючими органами, що мають обмежені інтервали керуючих параметрів. При цьому вихід за діапазон може стати причиною відмови елементів об'єкта керування. Надалі під ЗС будемо розуміти апаратне та ПЗ, людину-оператора з яким взаємодіє ЕОМ, а також власне фізичне середовище у якому функціонує космічний апарат.

Як видно з рис.1, поведінка об'єкту керування, визначається композицією відображень:

$$E_{OUT} \overset{R_{ER}}{\to} R_{IN} \overset{F_{RX}}{\to} S \times X \overset{F_{XY}}{\to} Y \overset{F_{YR}}{\to} R_{OUT} \overset{R_{RE}}{\to} E_{IN} \quad (4)$$

Необхідною умовою бездефектності ПЗ є сюр'єктивність та ін'єктивність тобто бієктивність цих відображень [7]. Їх композиція є бієкцією:

$$E_{OUT} \overset{R_{ER}\cdot F_{RX}\cdot F_{XY}\cdot F_{YR}\cdot R_{RE}}{\to} E_{IN} \quad (5)$$

Відповідність $\varphi \subseteq S \times E_{OUT} \times E_{IN}$, що реалізує програмне зебезпечення, із множини $S \times E_{OUT}$ у множину E_{IN} має бути функціональною, що гарантуватиме те, що ПЗ у еквівалентних станах, що визначаються вектором S, при отриманні одних і тих же вхідних даних матиме ту саму реакцію. Ця умова є необхідною умовою відсутності програмних дефектів.

Більш детальне аналізування множин та їх відображень дозволить сформулювати додаткові умови відсутності ДПЗ. Надалі будемо вважати два інтервали еквівалентними, якщо їх межі збігаються. Далі будемо розглядати будь-які інтервали I_1, I_2 як решітки(структури) [7, 8]. Тобто $I_1 < I_2$ або $I_1 \subset I_2$, якщо $\inf(I_2) < \inf(I_1) \wedge \sup(I_2) > \sup(I_1)$. Розглянемо детально відображення, які визначені у композиції відображень (4) та їх вплив на ДПЗ.

1. Відображення множин реальних вхідних даних з датчиків на вхідні дані ПС, що визначається вимогами до ПЗ: $E_{OUT} \overset{R_{ER}}{\to} R_{IN}$.

1.1 *Вимоги до системи повністю узгоджені з вихідними даними зовнішнього середовища* $\forall(a=1,1)\quad E_{OUT_a} = R_{IN_a}$. ДПЗ, обумовлені помилками вимог відсутні.

1.2 *Інтервали вхідних даних, що визначені у вимогах занадто вузькі по відношенню до ЗС*, тобто всі дані, які визначені у вимогах до ПЗ, мають інтервали, що не покривають відповідні реальні інтервали. Можливі відмови через дефекти, що спричинені помилками вимог: $\forall(a=1,1)\quad R_{IN_a} \subseteq E_{OUT_a}$.

1.3 *Вимоги до ПЗ мають помилки* і не покривають всю множину можливих значень всіх параметрів зовнішнього середовища. Можливі відмови, через дефекти, що спричинені помилковими вимогами: $\exists(a,b \in 1,1)$, $a \neq b$, $\sup(E_{OUT_a}) > \sup(R_{IN_a}) \vee \inf(E_{OUT_b}) < \inf(R_{IN_b})$.

1.4. *Вимоги до ПЗ мають грубі помилки* і інтервали можливих значень параметрів, що визначені вимогами, не перетинаються з інтервалами можливих значень параметрів ЗС. ПЗ буде мати велику кількість дефектів: $\forall(a=1,1)\quad E_{OUT_a} \cap R_{IN_a} = \varnothing$.

1.5. *Інтервали вхідних програмних даних, визначені у вимогах занадто широкі* по відношенню до зовнішнього середовища, тобто всі інтервали можливих значень, що визначені вимогами, перекривають всі можливі реальні інтервали ЗС. ПЗ буде мати фрагменти коду, які ніколи не виконуватимуться: $\forall(a \in 1,m)\quad E_{OUT_a} \subseteq R_{IN_a}$.

2. Відображення множини вихідних даних з датчиків, яка визначається вимогами до ПЗ, на множину даних, що надходить на вхід ПЗ, яка визначається реалізацією ПЗ: $R_{IN} \overset{F_{RX}}{\to} X$.

2.1 *Вимоги до системи повністю узгоджені з вихідними даними зовнішнього середовища* $\forall(a \in 1,n)\quad R_{IN_a} = X_a$. У цьому випадку ДПЗ, обумовлені помилковою реалізацією вимог відсутні.

2.2 *Реалізовані у ПЗ інтервали вхідних даних занадто вузькі по відношенню до інтервалів, що визначені вимогами.* Тобто всі вхідні параметри, що реалізовані у ПЗ, мають

інтервали, які не покривають відповідні реальні інтервали можливих значень, що визначені у вимогах до ПЗ. Можливі відмови, через ДПЗ, що спричинені помилками реалізації ПЗ: $\forall(a \in 1,n) \quad X_a \subseteq R_{IN_a}$.

2.3. *Реалізовані помилкові інтервали вхідних даних,* що не покривають всю множину можливих значень всіх параметрів зовнішнього середовища, що визначені у вимогах до ПЗ. Можливі відмови, через ДПЗ, що спричинені помилковими вимогами: $\exists(a,b \in 1,n)$, $a \neq b$, $\sup(R_{IN_a}) > \sup(X_a) \vee \inf(R_{IN_b}) < \inf(X_b)$.

2.4. *Програмна реалізація має грубі помилки* і інтервали можливих значень параметрів, що визначені вимогами, не перетинаються з інтервалами можливих значень параметрів, що були реалізовані у програмі. ПЗ буде мати велику кількість ДПЗ: $\forall(a \in 1,n) \quad X_a \cap R_{IN_a} = \varnothing$.

2.5. *Програма має занадто широкі інтервали вхідних даних* по відношенню до вимог, тобто всі інтервали можливих значень, що визначені вимогами, перекриваються відповідними програмними інтервалами. ПЗ буде мати фрагменти коду, які ніколи не виконуються: $\forall(a \in 1,n) \quad R_{IN_a} \subseteq X_a$.

3. Відображення декартового добутку реалізованих множин вхідних даних та внутрішнього стану програмної системи на реалізовану керуючу множину: $X \times S \xrightarrow{F_{XY}} Y$. Необхідною умовою відсутності дефектів є бієктивність цього відображення.

4. Відображення реалізованої в ПЗ керуючої множини на множину вхідних даних виконуючих органів, що визначається вимогами до ПЗ: $Y \xrightarrow{F_{YR}} R_{OUT}$.

4.1. *Інтервали керуючих даних, що реалізовані у ПЗ, повністю узгоджені з інтервалами, що визначені вимогами до ПЗ* $\forall(a = 1,p) \; Y_a = R_{OUT_a}$.

У цьому випадку ДПЗ, обумовлені помилками інтервальної реалізації керуючих даних відсутні.

4.2. *Інтервали керуючих даних занадто широкі по відношенні* до інтервалів, що визначені вимогами. Можливі відмови, через дефекти, що спричинені помилковою реалізацією інтервальних вимог: $\forall(a = 1,p) \; R_{OUT_a} \subseteq Y_a$.

4.3. *Інтервали елементів керуючого вектора мають помилки* по відношенню до інтервалів, які визначені вимогами. Можливі відмови, через дефекти, що спричинені реалізацією інтервальних вимог до елементів керуючого вектору: $\exists(a,b \in 1,p)$, $a \neq b$, $\sup(Y_a) > \sup(R_{OUT_a}) \vee \inf(Y_b) < \inf(R_{OUT_b})$.

4.4. *Інтервальна реалізація керуючого вектора має грубі помилки* і інтервали можливих значень елементів вектора керування, що реалізовані у ПЗ, не перетинаються з інтервалами можливих значень керуючого вектора, що визначені вимогами. ПЗ матиме велику кількість дефектів: $\forall(a \in 1,p)$, $Y_a \cap R_{OUT_a} = \varnothing$.

4.5. *Інтервали можливих значень керуючих параметрів, що реалізовані у ПЗ занадто вузькі* по відношенню до інтервалів, що визначені у вимогах до ПЗ. Можливі відмови через помилкову інтервальну реалізацію елементів керуючого вектора: $\forall(a \in 1,p)$, $Y_a \subseteq R_{OUT_a}$.

5. Відображення елементів керуючої множини, які визначено вимогами, на множину вхідних даних виконуючих органів, що визначається зовнішнім середовищем: $R_{OUT} \xrightarrow{R_{RE}} E_{IN}$.

5.1. *Елементи вектора керування повністю узгоджені з зовнішнім середовищем.* Тобто всі інтервали елементів керуючого вектора, що визначені у вимогах збігаються з інтервалами вхідних даних зовнішнього середовища: $\forall(a = 1,m)$, $E_{IN_a} = R_{OUT_a}$. ДПЗ, обумовлені помилковими вимогами відсутні.

5.2. *Інтервали елементів керування, що визначені у вимогах, занадто вузькі по відношенню до зовнішнього середовища*, тобто всі елементи керування мають інтервали, які не покривають реальні інтервали можливих значень ЗС. Можливі відмови через неможливість керування об'єктом у повному діапазоні використаних виконуючих органів: $\forall(a = 1,m)$, $R_{OUT_p} \subseteq E_{IN_p}$.

5.3. *Вимоги до системи мають помилки* і не покривають всю множину можливих значень параметрів ЗС. Можливі відмови через ДПЗ, обумовлених помилками у вимогах: $\exists(a,b \in 1,m)$, $a \neq b$, $\sup(E_{IN\,a}) > \sup(R_{OUT_a}) \vee \inf(E_{IN_b}) < \inf(R_{OUT_b})$.

5.4. *Вимоги до системи мають грубі помилки* і інтервали елементів керування, що визначені вимогами, не перетинаються з інтервалами можливих значень елементів керування, що визначені зовнішнім середовищем. ПЗ буде мати через велику кількість дефектів: $\forall(a \in 1,m)$, $E_{IN_a} \cap R_{OUT_a} \neq \varnothing$.

5.5. *Інтервали елементів керування, що визначені вимогами, занадто широкі по відношенню до ЗС,* тобто всі інтервали можливих значень, що визначені вимогами, перекривають всі можливі реальні інтервали зовнішнього середовища: $\forall(a \in 1,m)$, $E_{IN_a} \subseteq R_{OUT_a}$.

Можливі відмови, обумовлені виходом елементів керування за діапазон, через ДПЗ, що спричинені помилковими вимогами.

Кожне із розглянутих відображень може стати джерелом ДПЗ. Тому під час життєвого циклу ПЗ велика увага приділяється тестуванню. Головне завдання тестування ПЗ – перевірка відповідності вектор-функції F вимогам. У свою чергу вимоги через свою помилковість можуть не відповідати зовнішньому середовищу де буде працювати ПЗ. Ось чому заключна перевірка ПЗ відбувається під час дослідної експлуатації.Розглянуті джерела спричиняють ДПЗ через вплив людського фактору як на розроблення ПЗ, так і експлуатацію. У свою чергу процеси дефектоутворення через вплив людського фактору обумовлені термодинамічним підґрунтям процесів людського мозку.

4. Оцінювання надійності програмного забезпечення

Складність реального світу обумовлює необхідність створення ПЗ, що забезпечує необхідну функціональність та дозволяє з необхідною точністю автоматизувати процеси керування космічними об’єктами. Сучасні системи керування в більшості випадків є цифровими, тобто нелінійними, для яких мале змінення будь-якого параметру може спричинити якісне змінення поведінки, тому що складна поведінка реалізується з використанням великої кількості умовних блоків та циклів. Кожний з умовних блоків може збільшувати кількість можливих шляхів у два рази. Залишкові ДПЗ можуть бути виявлені тільки при «покритті» тестуванням відповідних програмних гілок. Для сучасних програмних систем, що мають обсяг коду $10^{6\ldots7}$ рядків, кількість умовних блоків може перевищувати десятки тисяч. Тому протестувати всі можливі обчислювальні шляхи в умовах ресурсних та часових обмежень просто неможливо. Ось чому використовують концепцію класів еквівалентності (КЕ) даних [5], за якою інтервал можливих значень кожного з елементів даних перетворюється на сукупність інтервалів, а поведінка системи тестується для обмеженої кількості комбінацій даних, що надходять до системи.

Якщо R еквівалентність на множині A то множина його КЕ – це фактор-множина множини A за еквівалентністю R, а $B \subseteq A$, що складається з представників всіх КЕ по одному з кожного класу називається повною системою представників КЕ.

Позначимо фактор-множини елементів вектора даних як $X_i = \{x_{j1}, x_{j2}, ..., x_{jm_i}\}$, а класи фактор-множини елементів вектору стану системи як: $S_k = \{s_{j1}, s_{j2}, ..., s_{jp_k}\}$, де i – розмірність вектора даних; k – розмірність вектора внутрішнього стану програмної системи; m_i – розмірність вектора класів еквівалентності для j-ої компоненти вектора даних; p_k – розмірність вектора класів еквівалентності для j-ої компоненти вектора стану. Розмірність векторів класів еквівалентності визначається вимогами.

Повне тестування, що покриває всі можливі шляхи вимагатиме перевірки поведінки сситеми для усіх комбінацій елементів можливих даних та внутрішнього стану. Насправді в різних режимах роботи можливі не всі комбінації вхідних даних

(частина даних буде просто некоректними), але у загальному випадку тестування вимагатиме перевірки коректного функціонування сситеми на декартовому добутку векторів КЕ множин даних та внутрішнього стану:

$$T = T_X \times T_S = X_1 \times ... \times X_i \times S_1 \times ... \times S_k = \\ = \{x_{11}, x_{12}, ..., x_{1m_1}\} \times ... \times \{x_{i1}, x_{i2}, ..., x_{im_i}\} \times \\ \times \{s_{11}, s_{12}, ..., s_{1p_1}\} \times ... \times \{s_{k1}, s_{k2}, ..., s_{kp_k}\} \quad (6),$$

де s_{ab}, x_{ab} – b-й інтервальний КЕ для a-го елементу даних чи стану системи.

Повне тестування на повній системі представників КЕ Т забезпечить ідентифікацію всіх ДПЗ, а їх подальше виправлення – безвідмовність для будь-яких вхідних даних у будь-якому стані. Тобто для вичерпного тестування необхідно використати як вхідні дані таку підмножину, що має по одному представнику множини T. Дуже часто тестування обмежується підмножиною $T_{SX} \subseteq T_X$. Тому ймовірність помилки буде визначатися ймовірністю існування дефектів на фрагменті коду, що належить шляху, який у свою чергу визначається множиною даних, що надходять до системи, яка знаходиться у стані s.

Ймовірність безвідмовної роботи обмежується умовою:

$$P \geq card(T_{SX}) / card(T_X) \quad (7)$$

Загальна кількість шляхів визначається як добуток кількості внутрішніх станів на кількість класів еквівалентності даних, тому зручно шлях визначати як елемент матриці $w_{q,r}$.

Позначимо ймовірність помилки у k-фрагменті коду, що належить q,r-шляху який визначається q-вектором даних та r-вектором стану як:

$$\rho_{k,q,r} = P_{CW}(C_k / w_{q,r}) P_C(C_k) P_{XS}(x_q / s_r) \quad (8)$$

де $P_{XS}(x_q / s_r)$ – умовна ймовірність надходження нетестованої q-множини даних до системи, що знаходиться у r-стані; $P_{CW}(C_k / w_{q,r})$ – умовна ймовірність виконання k-фрагменту коду на q,r-шляху; $P_C(C_k)$ – ймовірність існування ДПЗ на k-фрагменті коду. Тоді ймовірність помилок на q,r-шляху визначається як елемент матриці

$$\beta_{q,r} = 1 - \prod_{k=1}^{n_{q,r}} (1 - \rho_{k,q,r}), \quad (9)$$

де $n_{q,r}$ – матриця, що кількості фрагментів q,r-шляху.

Тоді загальна ймовірність БР визначається як:

$$P = 1 - \prod_{r=1}^{N} \prod_{q=1}^{n_r} (1 - \beta_{q,r}), \quad (10)$$

де $N = card(S)$ – кількість можливих внутрішніх станів програми; n_r – вектор, елементи якого визначають кардинальність декартового добутку класів еквівалентності даних, що надходять до системи, коли вона знаходиться у r-стані.

Якщо вважати, що кожен з фрагментів виконується один раз на протязі обчислювального шляху, а присутність дефектів у фрагментах коду визначається їх складністю, та складність усіх фрагментів залежить від їх довжини (обсягу коду), то ймовірність безпомилкової роботи визначатиметься:

$$P \approx 1 - \prod_{r=1}^{N} \prod_{q=1}^{n_r} \prod_{k=1}^{n_{q,r}} \left(1 - g \frac{L^2(C_k)}{L(w_{q,r})} P_{XS}(x_q / s_r)\right), \quad (11)$$

де L –функція, що визначає обсяг фрагмента та обчислювального шляху (її значення найлегше отримати шляхом статичного аналізу коду); g – коефіцієнт, що визначає середню щільність ДПЗ, його величина залежить від кваліфікації розробників, ретельності тестування та верифікації ПЗ і може бути визначено за результатами тестування. Визначення чисельних значень верхніх меж індексів N, n_r, $n_{q,r}$ вимагає розробки спеціалізованого статичного аналізатора коду.

Визначення матриці ймовірності надходження q-класу еквівалентності під час находження системи у r-стані вимагатиме ретельного аналізування результатів дослідної експлуатації ПС. На відміну від класичної оцінки (4), ймовірність БР залежить від шляху, який у свою чергу визначається множиною даних на вході системи, що знаходиться у певному стані та ймовірністю існування ДПЗ і не залежить від часу. Таким чином прояв дефектів має комбінаційну природу.

У той же час інтенсивність потоку помилок можливо визначити як кількість дефектів, що проявляються за одиницю часу роботи системи. Тому інтенсивність потоку помилок залежить від кількості операцій, що виконуються за одиницю часу, тобто від швидкодії обчислювальної машини, яка входить до контуру керування космічним об'єктом.

Висновки

В роботі розглянуто класичний підхід до оцінювання надійності програмних систем, доведено некоректність використання часової залежності ймовірності безпомилкової роботи програмної системи. Проаналізовано джерела можливих дефектів програмного забезпечення. Визначено фактори, що впливають на прояв дефектів ПЗ та сформовані необхідні умови його відсутності дефектів. Запропоновано комбінаційну оцінку ймовірності безпомилкової роботи ПЗ.

Подальші дослідження слід проводити у напрямку автоматизації статичного аналізування програмного коду, що дозволить оцінити невідомі фактори та автоматично визначати ймовірність безпомилкової роботи програмної системи. Перспективним вважається також дослідження використання методів теорії алгебричної симетрії та теорії фізичних розмірностей для зменшення на підставі функціональних залежностей, що реалізовані програмним забезпеченням, кількості необхідних тестів програмного забезпечення систем керування космічними літальними апаратами.

Список використаних джерел

1. Вентцель, Е.С. Теория вероятностей и ее инженерные приложения: учеб. пособие для втузов / Е.С. Вентцель, Л.А. Овчаров Л.А. – М.: Высш. шк., 2000. – 480 с.

2. Дубровский, П.В. Обеспечение надежноститехнологических процессов: учеб. пособие /П.В. Дубровский - Мин образования РФ Ульяновск, УлГТУ, 2000. – 124 с.

3. Lyu M.R. Handbook of Software Reliability Engineering / M.R. Lyu. – London: McGrow-Hill, 1996. – 805 p.

4. Динамическая отработка программного обеспечения бортовых цифровых вычислительных машин систем управления объектов ракетно-космической техники / Я.Е. Айзенберг, А.В. Бек, Ю.М. Златкин и др. // Космическая наука и технология. – 1997. – Т. 3. – № 1. – С. 61 – 74.

5. Вентцель, Е.С. Теория случайных процессов и ее инженерные приложения / Е.С. Вентцель, Л.А.Овчаров. – М.: Высшая школа, 2000. – 480 с.

6. Карпов, Ю.Г. ModelChecking: Верификация параллельных и распределенных программных систем / Ю.Г. Карпов. – СПб.: БХВ – Петербург, 2010. – 560 с.

7. Калужин, Л.А. Введение в общую алгебру / Л.А. Калужин. – М.: Наука, 1973. – 448 с.

8. Винберг, Э.Б. Курс алгебры / Э.Б. Винберг. – М.: Факториал Пресс, 2001. – 544 с.

Відомості про автора:

Манжос Юрій Семенович – канд.техн.наук, доц. кафедри інженерії програмного забезпечення Національного аерокосмічного університету імені М.Є. Жуковського «ХАІ». Наукові інтереси: інженерія програмного забезпечення.

E-mail: manzhos@ukr.net

.