ТЕСТУВАННЯ, ВАЛІДАЦІЯ ТА ВЕРИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

УДК 004.415.5

Ю.С. Манжос

АНАЛІЗ СЕМАНТИЧНИХ ДЕФЕКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Національний аерокосмічний університет «ХАІ» ім. М.Є.Жуковського
кафедра інженерії програмного забезпечення
e-mail:Manhos@ukr.net

Доведена необхідність створення незалежної верифікації, що базується на контролі семантичних інваріантів. Проведено аналіз статистичних характеристик програмного забезпечення. Обґрунтовано використання апарату теорії випадкових процесів для побудови моделі аналізу семантичних дефектів. Запропоновано модель контролю семантичних програмних дефектів, що базується на використання теорії випадкових процесів та дозволяє визначити функціональну залежність ефективності семантичного контролю від статистичних характеристик програмного забезпечення. Визначено кількісний інтервал ефективності семантичного контролю для програмного забезпечення систем реального часу. Розглянуті варіанти інформаційної технології аналізу семантичних дефектів програмного забезпечення. Подальші дослідження потребують розширення технології на нові програмні мови та використання інших програмних інваріантів.

Доказана необходимость создания независимой верификации, основанной на контроле семантических инвариантов. Проведен анализ статистических характеристик программного обеспечения. Обосновано использование аппарата теории случайных процессов для построения модели анализа семантических дефектов. Предложена модель контроля семантических программных дефектов, которая основана на использовании теории случайных процессов и позволяет определить функциональную зависимость эффективности семантического контроля от статистических характеристик программного обеспечения. Определен количественный интервал эффективности семантического контроля для программного обеспечения систем реального времени. Дальнейшие исследования требуют расширения технологии на новые программные языки и использование иных программных инвариантов.

The necessity of creating the independent software verification based on the software invariant has been proofed. The software statistical characters analysis has been implemented. The using of the random process theory has been proofed for building of the model of semantic software defect checking. The model of semantic software defect checking has been proposed. The model based on the theory of random process in order to determine the functional dependence of semantic checking effectiveness from statistical characteristic of software. The number interval of semantic checking effectiveness for real-time software has been calculated. The next researching needs the extension of software language set and the using of other software invariants.

Ключові слова: випадковий процес, ймовірнісна модель, незалежна верифікація, програмне забезпечення, програмний інваріант, семантичний контроль, семантичний простір, физична розмірність, вероятностная модель, независимая верификация, программное обеспечение, программный инвариант, семантический контроль, семантичний простір, случайный процесс, физическая размерность, independent verification, physical dimension, probabilistic model, random process, semantic check, semantic space, software, software invariant

Вступ

Сучасне життя неможливе без використання інформаційно-керуючих систем, що не тільки автоматизують технологічні процеси у різних галузях, але й забезпечують безпеку функціонування різноманітних технічних комплексів, зокрема − енергетичних, транспортних, нафтопереробних. Ось чому якості програмного забезпечення інформаційно-керуючих систем необхідно приділяти першочергову увагу. Необхідною умовою якості є проведення незалежної верифікації, що може базуватися на різних теоретичних засадах. Перспективним є аналіз збереження програмних інваріантів − властивостей програм, що зберігаються при будь-яких комбінаціях вхідних даних. Одним із таких інваріантів є фізична розмірність програмних змінних, або програмна семантика.

Порушення семантики буде свідчити про наявність так званих семантичних дефектів.

Розробка інформаційної технології незалежної верифікації, що базуватиметься на контролі збереження семантичного інваріанту, дозволить оцінювати семантичну коректність і, опосередковано, якість програмного забезпечення. Однак розробці цій технології повинна передувати теоретична оцінка її ефективності.

Існуючі моделі не дозволяють оцінити в повній мірі ефективність аналізу семантичних дефектів програмного забезпечення − семантичного контролю. Необхідно провести додаткові дослідження та оцінити ефективність семантичного контролю для потреб незалежної верифікації.

Аналіз досліджень і публікацій

Безпека держави визначається безпекою процесів в економіці, техніці, екології, державному управлінні та інших галузях людської діяльності. Це вимагає застосування програмно-технічних комплексів, що керують у реальному часі складними технологічними об'єктами та гарантують безпеку продукції, процесів виробництва і експлуатації систем, унеможливлюючи відмови і збитки, пов'язані заподіянням шкоди життю, здоров'ю громадян, а також майну і середовищу [1]. ІУС мають обмежений термін для виконання у реальному часі важливих, або критичних з погляду ресурсних витрат задач. Невиконання може мати катастрофічні наслідки, тому велику частку інформаційно-керуючих систем відносять до систем критичного застосування.

Типові представники – військові системи: літаки, кораблі, танки, тактичні і стратегічні ракети; космічні системи; командні, управляючі, комунікаційні та інтелектуальні системи. До 80% функцій реалізовано програмно, тому їх відносять до класу систем з інтенсивним використанням програмного забезпечення. Це обумовлює тенденцію експоненціального зростання обсягів коду [2], який сягнув десяти мільйонів для F-18 [3] та ста мільйонів інструкцій для інформаційно-керуючих систем АЕС.

Надійність і безпека систем визначаються якістю проектування на етапах, що передують розробці програм [4]. Помилки цих етапів стають джерелами складних дефектів [5, 6]. Але програма не тільки акумулює недоліки попередніх етапів, вона дозволяє виявити та усунути дефекти. Трудомісткість при цьому зростає на порядок. Ретельне тестування дозволяє досягти 10, а додаткові заходи – одного дефекту на 1000 рядків коду [7].

Разом з цим зростає й «вага» дефектів як джерел відмов, оскільки із зростанням складності збільшується і кількість залишкових дефектів, що зменшує потенційну безпеку функціонування [8]. За різними оцінками, дефекти призводять до 70% відмов у енергетиці, ракетно-космічних комплексах, банківській діяльності, медицині [9, 10]. Тенденція має динаміку наростання в часі.

Базовою нормативною вимогою безпечної експлуатації систем, що обумовлена залишковими дефектами, є проведення незалежної верифікації [8] під час сертифікації – дії третьої, незалежної сторони, що доводить відповідність програмного забезпечення конкретним стандартам та нормативній документації, і оцінює кількість залишкових дефектів методами, відмінними від методів розробника, що гарантує технологічну диверсність (різноманіття) незалежної верифікації. Актуальним є підвищення ефективності у створенні, застосуванні та поліпшенні об'єктивності оцінок характеристик програмного забезпечення, тому що врахування вимог стандартів обмежене можливостями експерта, а існуюча система оцінки якості під час експертизи ґрунтується на ручному аналізі великих обсягів документації, тому є суб'єктивною та інерційною. Як наслідок, існують проблеми повноти, достовірності і трудомісткості експертних оцінок та викликані цим ризики залишкових дефектів.

Розробка та впровадження в експертизу і сертифікацію автоматизованих засобів, що реалізують диверсні, відмінні від використовуваних розробниками, методи для об’єктивної оцінки якості програмного забезпечення, збільшать повноту та достовірність експертних оцінок надійності.

Концепція, що є основою методу незалежної семантичної верифікації, полягає у розгляді характеристик функціональності інформаційно-керуючих систем у семантичному просторі [11, 12]. Основу семантичного простору становить семантичний базис – n (ортогональних) векторів одиничної довжини, кожний з яких відповідає одній з головних одиниць обраної системи одиниць. Наприклад, для міжнародної системи одиниць СІ базис має дев’ять ортогональних векторів: що мають такі значення та розмірність: *Довжина* [метр]; *Маса* [кілограм]; *Час* [секунда]; *Сила електричного струму* [Ампер];

Термодинамічна температура [Кельвін]; *Сила світла* [кандела]; *Кількість речовини* [моль]; *Площинний кут* [радіан]; *Тілесний кут* [стерадіан].

Семантичний простір – n-вимірний метричний простір, що складається з множини X елементів, кожний з яких відповідає деякому фізичному типу, що характеризується відстанню – дійсною функцією ρ(x,y), визначеною для будь-яких пар (x,y) з X. Кожному з елементів СП відповідає семантичний вектор (СВ), який проведено з початку координат. Координатами вектора є розмірність фізичного типу в обраній системі одиниць. Вся сукупність СВ **S** утворює семантичний простір. СП є лінійним (афінним) простором над полем СВ, тому що існує правило складання, яке дозволяє кожній парі елементів X, Y∈ S знайти відповідний елемент Z ∈ S, який називається сумою, та існує правило множення на число, що дозволяє для кожного X∈ S та λ∈ S знайти елемент U∈ S – добуток елементу X на число λ, що позначається λX [13, 14].

Семантичний контроль дозволяє перевірити коректність програми наступним чином. Семантики операндів адитивних операцій (додавання, віднімання, порівняння, присвоєння) повинні збігатися. Семантика результату множення має відповідати семантичному вектору, що є сумою семантичних векторів операндів. Семантика результату ділення має відповідати семантичному вектору, що є різницею семантичних векторів операндів. Таким чином, адитивні операції дозволяють контролювати семантичний інваріант шляхом контролю семантик операндів. Виявлення неузгодженості семантик свідчить про порушення семантичного інваріанту та наявність програмного дефекту. Навпаки, узгодженість семантик не може свідчити про відсутність програмних дефектів. Найлегша реалізація семантичного контролю можлива за допомогою визначення спеціальних класів. Недоліком цього шляху є обмеженість мов програмування. Найбільш доцільним є використання спеціального аналізатора програмного коду, що контролює збереження семантичного інваріанту.

Постановка завдання

Існуючі моделі [14] не дозволяють оцінити ефективність семантичного контролю якості програмного забезпечення без знання ймовірності залишкових дефектів та довести тим самим доцільність практичної реалізації та втілення семантичного аналізу. Потрібні додаткові дослідження для розв'язання таких задач:

1. Проаналізувати статистичні характеристики реального програмного забезпечення та обґрунтувати використання розвинутої теорії випадкових процесів для побудови більш довершеної моделі аналізу семантичних дефектів.

2. Розробити та дослідити модель аналізу семантичних дефектів.

3.Розглянути шляхи реалізації інформаційної технології семантичного аналізу програмного забезпечення.

Аналіз статистичних характеристик програмного забезпечення

Необхідною умовою застосування методів теорії випадкових процесів для аналізу ефективності семантичного аналізу є пуасонівський характер розподілу операцій та операндів у програмному коді [15].

Потоки операцій та операндів мають певні властивості:

− ординарність, через те, що операції та операнди з'являються поодинці і ймовірність потрапляння кількох операцій чи даних на елементарну дільницю коду значно менше ймовірності появи однієї операції чи операнду;

− відсутністю післядії, через те, що для будь-яких дільниць коду, що не перетинаються, кількість операцій та даних є незалежними випадковими величинами тобто ймовірність існування певної кількості операцій чи операндів не залежить від їх кількості на інших дільницях;

− стаціонарністю, через те, що ймовірність появи певної кількості операцій та операндів на дільниці коду залежить тільки від довжини дільниці, а не від її розташування.

Через те, що потокам операцій і операндів властиві – ординарність, стаціонарність та відсутність післядії, ці потоки є найпростішими або стаціонарними пуасоновськими потоками, що дозволяє для дослідження якості програмного забезпечення використати методи теорії випадкових процесів [16], застосування яких обумовлює необхідність визначення статистичних характеристик – інтенсивності появи операндів та операцій.

Для аналізу статистичних розподілів, характерних для програмного забезпечення

інформаційно-керуючих систем, було створено і використано простий статичний аналізатор програмного коду, що здійснював лексичний та частину синтаксичного розбору програмного коду мовою С++. Як зразок для дослідження було використано програмне забезпечення інформаційно-керуючої системи космічного апарата «Спектр», в розробці якої приймав участь автор на початку 90-х років.

Під час аналізу для різних категорій операцій підраховувалися частоти появи так званого інтервалу − кількості операцій інших категорій, що розташовані між двома операціями однієї категорії. Наприклад, кількість появ неадитивних операцій між двома адитивними. Для операндів інтервал визначався як кількість операндів, що мають різні фізичні розмірності (семантики), що розташовані між двома операндами за еквівалентними семантиками.

Результати статистичного аналізу програмного коду, показані на рис. 1. Наприклад, операційному інтервалу 6 відповідає 200 адитивних операцій, тобто у коді знайдено 200 пар адитивних операцій, що мають між собою 5 інших операцій. Аналогічно, знайдено 1600 інтервалів, обмежених адитивними операціями, що мають 3 неадитивних операції.

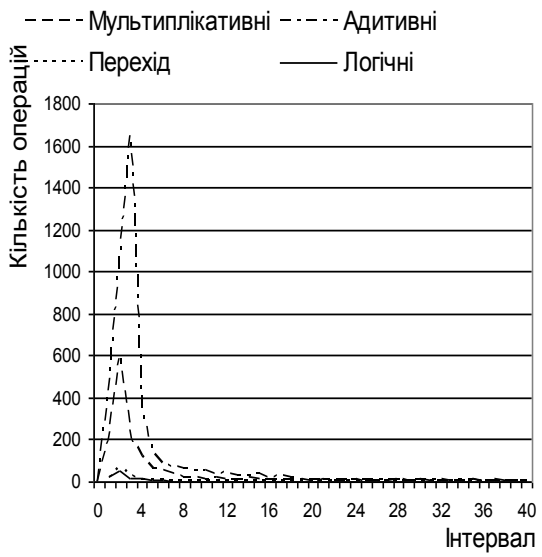

Рис. 1 Залежність абсолютної кількості операцій від програмного інтервалу

Нормалізація даних, відображених на рисунку 1, дозволила винайти функціональні залежності щільностей ймовірності для операцій різних категорій від операційних інтервалів, що зображені на рис. 2. Слід зазначити, що для кожної з функцій виконується умова:

$$\forall k, \int_0^{\infty} f_k(i)di = 1$$

тобто загальна площа (повна ймовірність події, пов'язаної з появою операції певної категорії) дорівнює одиниці.

Таким чином, ми маємо експериментально обґрунтовану можливість використання апарату теорії випадкових процесів для оцінки ефективності аналізу семантичних дефектів програмного забезпечення.

Ймовірнісна модель

Надалі будемо вважати, що аналіз семантичних дефектів буде здійснюватися абстрактним семантичним аналізатором, що має можливість розпізнавати семантичні дефекти через порушення семантичного інваріанту у статичному режимі, тобто без виконання програми.

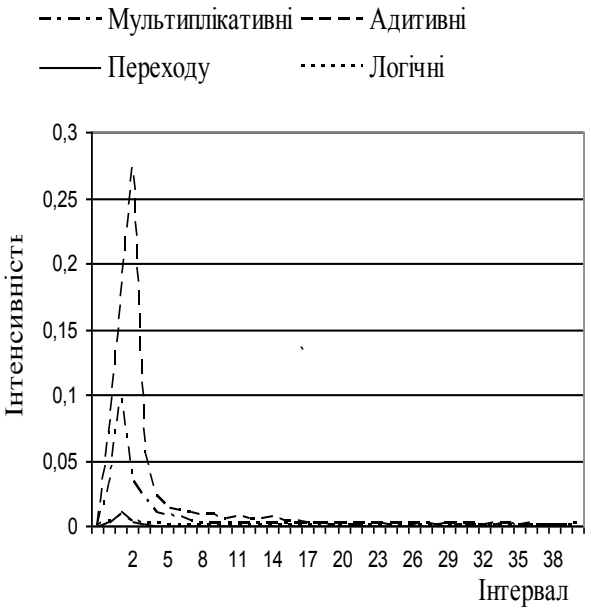

Рис. 2 Щільність ймовірності появи операцій як функція інтервалу

Будемо вважати, що дефект може проявлятися як спотворення будь-якої операції чи операнду. Факт спотворення може бути зафіксований аналізатором. Статичний аналізатор зручно розглядати як автомат з визначеною множиною станів та ймовірностями переходів між станами.

Причому, частину ймовірностей переходів можливо визначити завдяки аналізу статистичних характеристик програмного забезпечення. Інші ймовірності, наприклад ймовірність існування програмних дефектів, залишаться невідомими.

На підставі вище зазначеного, ефективність методу можливо визначити через відношення умовних ймовірностей розпізнання програмних дефектів:

$$\eta = \frac{P_F}{P_F + P_N}, \qquad (1)$$

де P_F, P_N – відповідно ймовірності виявлення та не виявлення програмних дефектів за умовами їх існування.

Таким чином, визначення ефективності семантичного аналізу вимагає кількісних значень ймовірностей P_F, P_N. Для знаходження цих величин розглянемо зображену на рисунку 3 ймовірнісну модель аналізу семантичних дефектів програмного забезпечення.

Під час побудови моделі, вважалося, що аналізатор програмного коду по черзі аналізує всі елементи програми U, що з ймовірностями ε_O, ε_C можуть бути відповідно операндами та операціями (командами), які позначені відповідно літерами O, C. Аналізуючи операнд, слід вважати, що він може бути дефектним, або коректним. Позначимо відповідні ймовірності як ε_{EO}, $1-\varepsilon_{EO}$.

У випадку коректного операнду, статичний аналізатор переходить до стану Ok. Наявність дефектного операнду призводить до переходу аналізатора у стан E_O. Коректність операнду визначається коректним застосуванням операнду, що може бути як програмною змінною, так і константою. Таким чином, збіг фізичних розмірностей програмних змінних не дає можливості їх розрізнити. У той же час нееквівалентність розмінностей дозволить розпізнати дефект ПЗ, викликаний некоректним використанням програмних змінних. Позначимо ймовірність збігу та нееквівалентності розмінностей відповідно ε_T, $1-\varepsilon_T$.

Таким чином, збіг розмінностей не дозволяє розпізнати дефект ПЗ, це позначено на моделі переходом аналізатора у стан N. У той же час нееквівалентність розмінностей дозволить розпізнати дефект. Це позначено на моделі переходом аналізатора до стану F. Відсутність дефектів операндів, що має місце з ймовірністю $1-\varepsilon_{EO}$, призводить до переходу аналізатора зі стану O до Ok.

З кожного зі станів аналізатора F, N, Ok, що відповідають розпізнанню та не розпізнанню, а також відсутності програмних дефектів, має безумовний перехід до стану початкового стану U.

Повернемося до початкового стану U. «Операції» та «операнди» утворюють повну групу, тобто $\varepsilon_O + \varepsilon_C = 1$. Таким чином, з ймовірністю ε_C СА переходить до стану C. Надалі будемо вважати, що всю сукупність операцій поділено на категорії: операції переходу, адитивні, мультиплікативні та виклики функцій. До адитивних операцій віднесено додавання, віднімання, порівняння та присвоєння. До мультиплікативних – множення, ділення та піднесення до степеню.

Якщо позначимо ймовірності категорій операцій як ε_J, ε_A, ε_M, ε_F, то перехід аналізатора до аналізу операцій відповідних категорій має тотожні ймовірності.

У випадку операцій переходу, що відповідає стану C_J, можлива альтернатива «дефект існує», «дефект відсутній». Позначимо ці ймовірності відповідно як ε_{EJ}, $1-\varepsilon_{EJ}$. Існування дефекту, пов'язаного з невірним переходом, не може бути розпізнано аналізом фізичної розмірності. Таким чином, з ймовірністю ε_{EJ} аналізатор перейде до стану N, а з ймовірністю $1-\varepsilon_{EJ}$, що відповідає відсутності дефектів, – до стану Ok.

У випадку виклику функцій, що відповідає стану C_F, можлива альтернатива «дефект існує», «дефект відсутній». Позначимо ці ймовірності відповідно як ε_{EF}, $1-\varepsilon_{EF}$.

Таким чином, з ймовірністю ε_{EF} СА перейде до стану E_F «дефект виклику функції», а з ймовірністю $1-\varepsilon_{EF}$, що відповідає відсутності дефектів, – до стану Ok. Існування дефекту, пов'язаного з невірним використанням викликів функцій, не може бути розпізнано аналізом фізичної розмірності тільки у випадку, коли фізичні розмірності всіх фактичних та формальних параметрів збігаються. Позначимо ймовірність такої події ε_{FT}. Таким чином з ймовірністю ε_{FT} аналізатор перейде до стану N, а з ймовірністю $1-\varepsilon_{FT}$ до стану F.

У випадку аналізу адитивних операцій, що відповідає стану C_A, можлива альтернатива

«дефект існує», «дефект відсутній». Позначимо ці ймовірності відповідно як ε_{EA}, $1-\varepsilon_{EA}$. Таким чином, з ймовірністю ε_{EA} аналізатор перейде до стану E_A «дефект адитивної операції», а з ймовірністю $1-\varepsilon_{EA}$, що відповідає відсутності дефектів, – до стану Ok. Існування дефекту, пов'язаного з невірним використанням адитивних операцій, може бути розпізнано аналізом фізичної розмірності тільки у випадку, коли у результаті програмного дефекту має місце використання замість адитивної операції мультиплікативної.

Позначимо ймовірність такої події ε_{AM}. Таким чином, з ймовірністю ε_{AM} аналізатор перейде до стану F, а з ймовірністю $1-\varepsilon_{AM}$ до стану N, що відповідає неможливості розпізнання програмних дефектів за умовою їх існування.

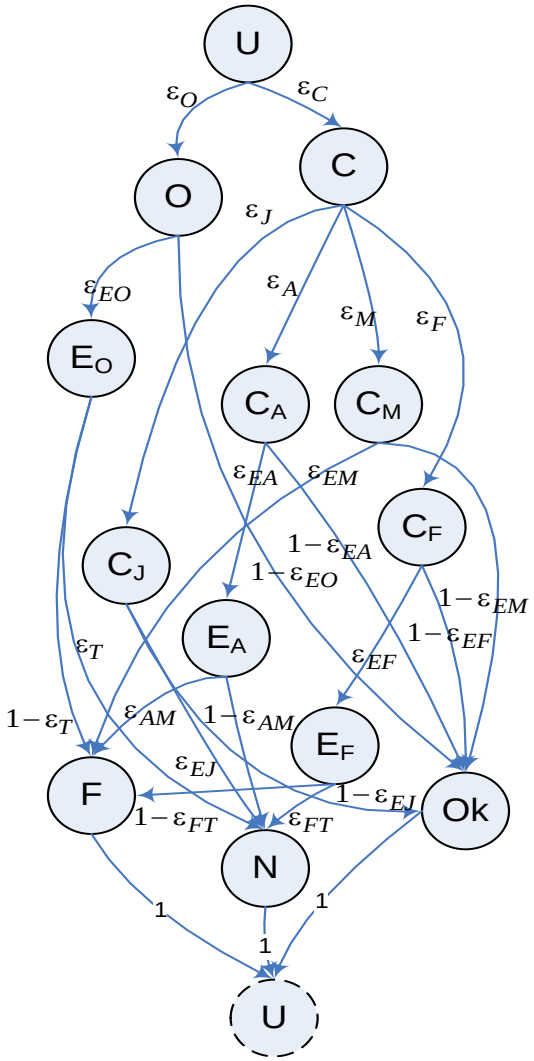

Рис.3 Ймовірнісна модель аналізу семантичних дефектів

У випадку аналізу мультиплікативних операцій, що відповідає стану C_M, можлива альтернатива «дефект існує», «дефект відсутній». Позначимо ці ймовірності відповідно як ε_{EM}, $1-\varepsilon_{EM}$. Таким чином, з ймовірністю ε_{EM} аналізатор перейде до стану F, що відповідає стану розпізнанню дефекту а з ймовірністю $1-\varepsilon_{EM}$, що відповідає відсутності дефектів мультиплікативних операцій – до стану Ok.

На підставі аналізу ймовірностей переходів для кожного зі станів імовірнісної моделі дефектів, та позначивши невідомі ймовірності знаходження аналізатора у кожному зі станів P_N, P_{OK}, P_F, P_U, P_C, P_{CM}, P_O, P_{EO}, P_{EA}, P_{CA}, P_{CF}, P_{CJ}, P_{EF}, побудуємо систему лінійних рівнянь (2).

$$\begin{cases} P_N + P_{OK} + P_F = P_U \\ P_U\varepsilon_C = P_C \\ P_C\varepsilon_M = P_{CM} \\ P_{CM}\varepsilon_{EM} + P_{EO}\left(1-\varepsilon_T\right) + \\ \quad + P_{EA}\varepsilon_{AM} + P_{EF}\left(1-\varepsilon_{FT}\right) = P_F \\ P_U\varepsilon_O = P_O \\ P_O\left(1-\varepsilon_{EO}\right) + P_{CM}\left(1-\varepsilon_{EM}\right) + P_{CA}\left(1-\varepsilon_{EA}\right) + \\ \quad + P_{CF}\left(1-\varepsilon_{EF}\right) + P_{CJ}\left(1-\varepsilon_{EJ}\right) = P_{OK} \\ P_O\varepsilon_{EO} = P_{EO} \\ P_C\varepsilon_A = P_{CA} \\ P_C\varepsilon_F = P_{CF} \\ P_{CA}\varepsilon_{EA} = P_{EA} \\ P_{CF}\varepsilon_{EF} = P_{EF} \\ P_C\varepsilon_J = P_{CJ} \\ P_{EA}\left(1-\varepsilon_{AM}\right) + P_{EF}\varepsilon_{FT} + \\ \quad + P_{EO}\varepsilon_T + P_{CJ}\varepsilon_{EJ} = P_N \end{cases} \qquad (2)$$

Коректне розв'язання системи (2) для знаходження P_F, P_N вимагає урахування умови нормування – умови одиничної ймовірності повної групи подій, що відповідають знаходженню аналізатора у кожному із станів:

$$P_U + P_C + P_{CM} + P_F + P_O + \\ + P_{OK} + P_{EO} + P_{CA} + P_{CF} + \\ + P_{CJ} + P_{EA} + P_{EF} + P_N = 1 \quad (3)$$

Заміна в системі (2) одного з рівнянь, наприклад рівняння (2.6) на (3) дозволяє отримати коректну систему (4).

$$\begin{cases} P_N + P_{OK} + P_F = P_U \\ P_U \varepsilon_C = P_C \\ P_C \varepsilon_M = P_{CM} \\ P_{CM} \varepsilon_{EM} + P_{EO}(1-\varepsilon_T) + \\ \quad + P_{EA} \varepsilon_{AM} + P_{EF}(1-\varepsilon_{FT}) = P_F \\ P_U \varepsilon_O = P_O \\ P_U + P_C + P_{CM} + P_F + P_O + \\ \quad + P_{OK} + P_{EO} + P_{CA} + P_{CF} + \\ \quad + P_{CJ} + P_{EA} + P_{EF} + P_N = 1 \\ P_O \varepsilon_{EO} = P_{EO} \\ P_C \varepsilon_A = P_{CA} \\ P_C \varepsilon_F = P_{CF} \\ P_{CA} \varepsilon_{EA} = P_{EA} \\ P_{CF} \varepsilon_{EF} = P_{EF} \\ P_C \varepsilon_J = P_{CJ} \\ P_{EA}(1-\varepsilon_{AM}) + P_{EF} \varepsilon_{FT} + \\ \quad + P_{EO} \varepsilon_T + P_{CJ} \varepsilon_{EJ} = P_N \end{cases} \quad (4)$$

Розв'язання системи (4) дозволяє знайти необхідні ймовірності розпізнання та пропуску семантичних програмних дефектів як функції статистичних характеристик програмного забезпечення:

$$\begin{cases} P_F = \dfrac{\varepsilon_C \varepsilon_M \varepsilon_{EM} + \varepsilon_O \varepsilon_{EO}(1-\varepsilon_T) + }{3 + \varepsilon_C + \varepsilon_O \varepsilon_{EO} + \varepsilon_C \varepsilon_A \varepsilon_{EA} + \varepsilon_C \varepsilon_F \varepsilon_{EF}} \to \\ \quad \to \dfrac{+\varepsilon_C \varepsilon_A \varepsilon_{EA} \varepsilon_{AM} + \varepsilon_C \varepsilon_F \varepsilon_{EF}(1-\varepsilon_{FT})}{3 + \varepsilon_C + \varepsilon_O \varepsilon_{EO} + \varepsilon_C \varepsilon_A \varepsilon_{EA} + \varepsilon_C \varepsilon_F \varepsilon_{EF}} \\ P_N = \dfrac{\varepsilon_C \varepsilon_A \varepsilon_{EA}(1-\varepsilon_{AM}) + \varepsilon_C \varepsilon_F \varepsilon_{EF} \varepsilon_{FT} + }{3 + \varepsilon_C + \varepsilon_O \varepsilon_{EO} + \varepsilon_C \varepsilon_A \varepsilon_{EA} + \varepsilon_C \varepsilon_F \varepsilon_{EF}} \to \\ \quad \to \dfrac{+\varepsilon_O \varepsilon_{EO} \varepsilon_T + \varepsilon_C \varepsilon_J \varepsilon_{EJ}}{3 + \varepsilon_C + \varepsilon_O \varepsilon_{EO} + \varepsilon_C \varepsilon_A \varepsilon_{EA} + \varepsilon_C \varepsilon_F \varepsilon_{EF}} \end{cases}$$

Позначивши абсолютну ймовірність існування програмних дефектів як

$$\varepsilon_D = \varepsilon_O \varepsilon_{EO} + \varepsilon_C (\varepsilon_M \varepsilon_{EM} + \varepsilon_A \varepsilon_{EA} + \varepsilon_F \varepsilon_{EF} + \varepsilon_J \varepsilon_{EJ})$$

та припустивши рівномірний розподіл дефектів за категоріями операцій

$$\varepsilon_{EM} = \varepsilon_{EA} = \varepsilon_{EF} = \varepsilon_{EJ} = d,$$

маємо, що ефективність семантичного аналізу ПЗ, що обчислюється за (1) визначається як

$$\eta = \varepsilon_C \varepsilon_M + \varepsilon_O (1-\varepsilon_T) + \varepsilon_C \varepsilon_A \varepsilon_{AM} + \varepsilon_C \varepsilon_F (1-\varepsilon_{FT})$$

З урахуванням додаткових умов повних груп подій

$\varepsilon_O + \varepsilon_C = 1$ та $\varepsilon_A + \varepsilon_M + \varepsilon_F + \varepsilon_J = 1$

Та після перетворень маємо:

$$\eta = 1 - \varepsilon_T - \varepsilon_C \left(\varepsilon_F \varepsilon_{FT} + \frac{\varepsilon_A}{2} + \varepsilon_J - \varepsilon_T \right), \quad (5)$$

де ε_C – умовна ймовірність (частка) операцій у програмному коді, що приблизно визначається як:

$$\varepsilon_C = \frac{N_C}{N_C + N_O},$$

де N_C, N_O – загальна кількість операцій (команд) та операндів у ПЗ;

ε_J – умовна ймовірність (частка) операцій переходу серед загальної кількості використаних операцій у програмному коді, що приблизно визначається як:

$$\varepsilon_J = \frac{N_J}{N_C + N_J},$$

де N_J – загальна кількість переходів;

ε_F – умовна ймовірність (частка) операцій виклику функцій серед загальної кількості використаних операцій у програмному коді, що приблизно визначається як:

$$\varepsilon_F = \frac{N_F}{N_C + N_F},$$

де N_F – загальна кількість викликів функцій;

ε_A – умовна ймовірність (частка) адитивних операцій серед загальної кількості використаних операцій у програмному коді, що приблизно визначається як:

$$\varepsilon_A = \frac{N_A}{N_C + N_A},$$

де N_A – загальна кількість адитивних операцій (додавань, віднімань, порівнянь, присвоєнь) у ПЗ;

ε_T – умовна ймовірність збігу фізичних розмінностей для різних програмних даних, що визначається як:

$$\varepsilon_T = \sum_{i=1}^{n} (1 - P_{Ti}) P_{Ti},$$

де P_{Ti} – частка операндів, що має еквівалентні розмірності та визначається як;

$$P_{Ti} = \frac{N_{Ti}}{N_O + N_{Ti}},$$

де N_O, N_{Ti} – відповідно загальна кількість операндів та кількість операндів, що мають відповідну фізичну розмірність, а n – загальна кількість використаних у програмі операндів;

ε_{FT} – умовна ймовірність збігу фізичних розмінностей формальних параметрів для різних програмних функцій, що визначається як:

$$\varepsilon_{FT} = \sum_{i=1}^{n} (1 - P_{FTi}) P_{FTi},$$

де P_{FTi} – частка функцій, що мають еквівалентні розмірності формальних параметрів та визначається як;

$$P_{FTi} = \frac{N_{FTi}}{N_F + N_{FTi}},$$

де N_F, N_{FTi} – відповідно загальна кількість функцій та кількість функцій, що мають еквівалентні фізичні розмірності формальних параметрів, а n – загальна кількість використаних у програмі функцій.

Таким чином, ефективність семантичного аналізу (5) цілком визначається як функція від статистичних характеристик програмного коду та даних.

Визначимо середню ефективність методу за допомогою інтегральної оцінки як гіпероб'єм для (5) у просторі з базисом $\varepsilon_T, \varepsilon_C, \varepsilon_J, \varepsilon_F, \varepsilon_{FT}, \varepsilon_A$, обмеженому межами, що визначають діапазон умовних ймовірностей $[0,1]$.

$$m_\eta = \frac{\int_V \left(1 - \varepsilon_T - \varepsilon_C \left(\varepsilon_F \varepsilon_{FT} + \frac{\varepsilon_A}{2} + \varepsilon_J - \varepsilon_T\right)\right) dv}{\int_v dv}, \quad (6)$$

де інтеграл по об'єму обчислюється у шестивимірному просторі, за умовою, що визначається обмеженням суми ймовірностей $0 \le \varepsilon_F + \varepsilon_A + \varepsilon_J \le 1$, а елемент об'єму визначається як

$$dv = d\varepsilon_T d\varepsilon_C d\varepsilon_J d\varepsilon_F d\varepsilon_{FT} d\varepsilon_A$$

В результаті чисельного інтегрування (6) для умов $0 \le \varepsilon_F + \varepsilon_A + \varepsilon_J \le 1$ та $0 \le \varepsilon_C, \varepsilon_T, \varepsilon_{FT} \le 1$ маємо m_η=0.49. Реальне програмне забезпечення інформаційно-керуючих систем має вузькі межі ймовірностей функцій, переходів, збігу фізичних розмірностей типів даних та збігу фізичних розмінностей формальних параметрів функцій. Результати інтегрування (6), обчислені для різних реальних варіантів обмежень статистичних характеристик програмного забезпечення приведені у таблиці 1

Таблиця1

Середня ефективність методу

Умови	m_η
$0 \le \varepsilon_F, \varepsilon_T, \varepsilon_{FT} \le 0.1$	0.87
$0 \le \varepsilon_F, \varepsilon_T \le 0.05$ $0 \le \varepsilon_F \le 0.1$ $0.3 \le \varepsilon_J \le 1$	0.86
$0 \le \varepsilon_F, \varepsilon_T \le 0.05$ $0 \le \varepsilon_F \le 0.1$ $0 \le \varepsilon_J \le 0.3$	0.90

Таким чином, для програмного забезпечення, характерного для інформаційно-керуючих систем космічних апаратів, семантичний контроль забезпечує ефективність більше ніж 80%. Тобто більше 80% семантичних дефектів – некоректних використань операндів, операцій, функцій може бути виявлено у режимі статичного аналізу без виконання великої кількості тестових завдань.

Інформаційна технологія аналізу семантичних дефектів

Інформаційна технологія аналізу семантичних дефектів була впроваджена у «Інтегрованій інструментальній системі підтримки експертизи та незалежної верифікації критичного програмного забезпечення», призначеної для аналізу та оцінки якості програмного забезпечення з використанням програмних інваріантів [17].

Програмне забезпечення, що утворює множину файлів мовою C/C++, аналізується разом з проектною документацією, що має характеристики програмних змінних.

Базові процедури незалежної верифікації:

– нормалізація об'єкта експертизи;

– інструментування програмного коду для побудови семантичної моделі програмного забезпечення;

– контроль семантичної коректності програмного забезпечення.

Нормалізація об'єкту експертизи здійснюється безпосередньо перед експертизою. Складається з формування списків програмних файлів та технічної документації, а також синтаксичного аналізу програмного коду. Результатом роботи є синтаксична модель.

Інструментування програмного забезпечення – базова процедура для подальших робіт з оцінки якості. Завершується побудовою семантичної моделі − системи лінійних рівнянь, побудованих на підставі семантичного відображення, що здійснює програмний код. По завершенні побудови семантичної моделі експерт вводить характеристики програмних змінних та визначає сигнатури функцій, що не мають програмного коду, для перевірки коректності їх використання.

Контроль семантичної корекції програмного забезпечення – основна процедура незалежної верифікації та експертизи, під час якої доводиться відсутність внутрішніх протиріч семантичної моделі. Тобто перевіряється узгодженість системи лінійних рівнянь та семантик програмних змінних.

Загальна оцінка семантичної коректності, у разі невиявлення програмних дефектів, формується як виважена сума локальних семантичних оцінок окремих програмних модулів, що визначаються за формулою (5). Як вагові коефіцієнти використовуються кількості операцій у модулях.

Апробація методу і розробленого інструментального засобу виконана під час незалежної верифікації ПЗ: ПТК системи автоматичного регулювання реакторного відділення ПТК САР-4 РО; системи керування перевантажувальної машини типу МПС-В-1000Z2-3-У4.2 енергоблоку №2 Запорізької АЕС; систем сигналізації аварійного захисту (СІАЗ) та регуляторів обмеження потужності (РОМ) для 1-го та 2-го блоків Ровенської АЕС розробки АТ НВО „ХАРТРОН".

З метою підвищення надійності програмно-технічних комплексів критичного застосування, була розроблена інформаційна технологія, що забезпечила динамічний контроль семантичної коректності обчислювальних процесів, що має у порівнянні з відомими меншу ресурсоємність та більш глибоку діагностуючу здібність виявлення семантичних програмних дефектів.

За основу інваріанто-орієнтованого підходу покладено семантичне відображення, що виконується програмним кодом над семантиками (фізичними розмірностями) даних. Технологія дозволяє без розробки додаткових версій функціонального програмного забезпечення ддіагностувати помилки обчислювальних процесів, спростити дистанційну модифікацію та супровід програмного забезпечення, і, як наслідок, підвищити надійність сучасних інформаційно-керуючих систем критичного використання.

Підвищення надійності досягається завдяки контролю збереження семантичного інваріанту під час виконання програми, а також. обчисленню статистичних характеристик, необхідних для визначення меж надійності у разі невиявлення інваріантних порушень.

Коректність коду контролюється завдяки переназначенню типів даних та операцій. При цьому, коректність мультиплікативних операцій контролюється перевизначеними адитивними операціями. Порушення семантики формує виключні ситуації або генерує керуючі сигнали у відповідні процеси.

Діагностування здійснюється у реальному часі з точністю до адитивної операції. Для полегшення визначення місць дефектного програмного коду, паралельно з ввиконанням основної задачі і діагностики відновлюється програмний код. Для цього, перевизначені операції, паралельно з виконанням керуючих алгоритмів, на підставі своїх аргументів та коду власної операції формують дерево арифметичного виразу, що відповідає фрагменту програмного кода. У випадку виявлення семантичних дефектів сформоване дерево приводиться до текстової форми та перенаправляється у відповідний потік виводу, який може бути зв'язаний як з окремим файлом, так і з іншим обчислювальним процесом, що забезпечує обробку телеметричної інформації.

Стендові випробування довели високу ефективність інформаційної технології для динамічного контролю обчислювальних процесів у реальному часі.

Висновки

Ймовірнісна модель аналізу семантичних дефектів, побудована з використанням апарату теорії випадкових процесів, запропонована у статті, дозволила визначити ефективність

семантичного контролю як функцію статистичних характеристик програмного забезпечення. Інтегральна оцінка дозволила точніше визначити межі ефективності, що склала біля 80%.

Розробка інформаційної технології аналізу семантичних дефектів довела її високу ефективність як під час статичної оцінки якості програмного забезпечення, так і під час контролю обчислювальних процесів у реальному часі, та дозволила виявити значну кількість програмних дефектів.

Подальші дослідження доцільно проводити у напрямку комплексного використання кількох програмних інваріантів та розширенню мов програмування, що потребує створення нового покоління інформаційної технології.

Література

1. *Харченко В.С.* Гарантоспособность и гаранто-способные системы: элементы методологии // Радіоелектронні і комп'ютерні системи. – 2006. – № 5(17). – С. 7 – 19.

2. *Guidelines* for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems, Command and Control Systems, Management Information System. Department of the Air Force, DC, June 1996. – Vol. 1. – 875 p.

3. *Engelland J.D.* Integrated Avionics Systems: Managing the System and the Software // Briefing present by General Dynamics to Boldstroke, Maxwell AFB, November 8, 1990. – 28 p.

4. *Липаев В.В.* Обеспечение качества программных средств. Методы и стандарты. – М.: СИНТЕГ, 2001. – 380 с.

5. *Кантор М.* Управление программными проектами. Практическое руководство по разработке успешного программного обеспечения: Пер. с англ. – М.: Издательский дом „Вільямс", 2002. – 176 с.

6. *Шафер Д., Фатрелл Р., Шафер Л.* Управление программными проектами: достижение оптимального качества при минимуме затрат: Пер. с англ. – М.: Издательский дом „Вильямс", 2003. – 1136 с.

7. *Guidelines* for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems, Command and Control Systems, Management Information System. Department of the Air Force, DC, June 1996. – Vol. 1. – 875 p.

8. *Липаев В.В.* Проблемы разработки и обеспечения качества крупномасштабных программных средств // Программирование. – 2005. – № 1. – С. 73 – 77.

9. *Харченко В.С., Асидех Ф.А.* Многоверсионные технологии и отказоустойчивые решения в бизнес-критических компьютерных системах // Открытые интегрированные информационные технологии. – Харьков: Нац. аерокосм. ун-т «Харк. авіац. ін-т», 2002, №15. – С. 35– 45.

10. *Кузнецов С.* Что надо сделать, чтобы компьютеры не вредили людям ? // Открытые системы. – 2006. – № 4. – С. 72 – 75.

11.*Манжос Ю.С.* Оценка эффективности независимой верификации программного обеспечения // Авіаційно-космічна техніка і технологія.– 2004. – № 7.– С.210 – 214.

12.*Манжос Ю.С.* Типізація даних у системах критичного застосування // Системи обробки інформації. – Харків: Харківський військовий університет, 2002. – Вип. 3(19).– С. 54 – 13.*Манжос Ю.С.* Принципы семантического контроля программного обеспечения // Авіаційно-космічна техніка і технологія. – Харків: Нац. аерокосм. ун-т „Харк. авіац. ін-т", 2002. – Вип. 32. – С. 307 – 315.

14.Манжос Ю.С. Семантический контроль программного обеспечения систем критического применения // Авіаційно-космічна техніка і технологія.– Харків: Нац. аерокосм. ун-т „Харк. авіац. ін-т", 2002. – Вип. 34. – С. 207–212.

15. *Манжос Ю.С.* Статистичні характеристики програмного забезпечення критичного застосування // Х.: Національний аерокос-мічний університет «Харк. авіац. ін-т», ІКТМ 2006, с. 454

16. *Вентцель Е.С., Овчаров Л.А.* Теория вероятностей и ее инженерные приложения. – М.: Наука, 1988. – 480 с.

17. *Конорев Б.М., Манжос Ю.С., Харченко В.С.* Инвариантно-ориентированная оценка качества программного обеспечения космических систем /Под ред. Конорева Б.М., Харченко В.С. – Харьков: Национальное космическое агентство Украины, Государственный центр регулирования качества поставок и услуг, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», 2009. – 228с.

Відомості про автора:

Манжос Юрій Семенович, доцент кафедри інженерії програмного забезпечення Національного аерокосмічного університету «ХАІ» імені М.Є. Жуковського, кандидат технічних наук; наукові напрями – верифікація програмного забезпечення, e-mail:Manhos@ukr.net

Стаття надійшла до редакції 19.02.2010

