УДК 004.052

В. Яковина, П. Сердюк, О. Нитребич, Д. Федасюк
Національний університет “Львівська політехніка”,
кафедра програмного забезпечення

ВИКОРИСТАННЯ МАРКОВСЬКИХ ЛАНЦЮГІВ ВИЩОГО ПОРЯДКУ В ЗАДАЧАХ МОДЕЛЮВАННЯ НАДІЙНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У моделях прогнозування надійності програмного забезпечення (ПЗ) засобами архітектурного підходу припущення про незалежність виконання компонент є спрощенням реального процесу роботи програмного забезпечення. Вдосконалено модель Гокаля з використанням Марковських ланцюгів вищого порядку, що дає змогу врахувати залежності виконання ПЗ у прогнозуванні його надійності.

Ключові слова: надійність ПЗ, Марковські ланцюги вищих порядків, критерій AIC.

Assumption of independent components execution in software reliability models built using architectural approach is a simplification of real software execution. In this paper Gokhale model with higher order Markov chains has been improved to appreciate software execution dependencies in it's reliability prediction.

Key words: software reliability, higher-order Markov chain, AIC criterion.

Вступ

В останні роки складність комп’ютерних програм, що виконують важливі та відповідальні функції, швидко зростає, тому питання надійності програмного забезпечення стають все актуальнішими. Використання адекватних моделей надійності ПЗ на етапах проектування та написання програмного коду дає змогу зменшити витрати на етапі тестування. У роботі згідно з [1] під надійністю розумітимемо ймовірність безвідмовної роботи програмного забезпечення за певний період часу за певних умов. Сьогодні існує багато підходів та моделей, які дозволяють прогнозувати або оцінювати надійність програмного продукту. Загалом, ці моделі поділяють на моделі “білої скриньки” та моделі “чорної скриньки” [2] залежно від використання інформації про архітектуру ПЗ.

Моделі чорної скриньки не враховують архітектуру програмного продукту, а прогнозування надійності здійснюється лише на основі аналізу вхідних/вихідних даних. В останнє десятиліття все більше уваги приділяється моделям, що враховують архітектуру ПЗ, тобто моделям “білої скриньки”, що своєю чергою поділяються на адитивні моделі, моделі, що ґрунтуються на шляхах виконання ПЗ та моделі, що використовують компонентний підхід [3]. Зрозуміло, що моделі “білої скриньки” адекватніше можуть описати надійність програмного забезпечення, адже вони оцінюють внутрішню будову ПЗ та взаємодію його компонент, тобто його архітектуру, тому цей тип моделей називають ще інакше – моделі, побудовані на базі архітектурного підходу.

Сьогодні значна кількість робіт присвячена дослідженню моделей на основі компонентного підходу, що використовують граф потоку керування для опису архітектури комп’ютерної програми [2, 3, 5, 6]. У цьому підході архітектуру ПЗ можна змоделювати як ланцюг Маркова з дискретним часом, неперервним часом та напівмарківським процесом [3]. Пізніше кожну з моделей можна класифікувати на поглинаючу (містить поглинаючий стан – стан, з якого система вийти не може) та непоглинаючу (не містить поглинаючих станів). Окрім того, цей клас моделей можна ще поділити на композиційні та ієрархічні моделі. До композиційних належать моделі, які одночасно комбінують архітектуру програмного продукту та характер його помилок для обчислення надійності ПЗ.

В ієрархічних моделях спочатку розв'язується архітектурна модель, а потім поведінка помилок системи додається до існуючого результату для прогнозування надійності ПЗ. Моделі цього класу переважно використовують теорію Марковських ланцюгів першого порядку, припускаючи, що виконання компонент ПЗ є незалежним [3–5].

Недостатню обґрунтованість припущення про незалежність виконання компонент реального ПЗ показано у роботах [1, 6], де запропоновано для врахування залежності виконання компонент ПЗ використовувати Марковські моделі вищих порядків.

Однак використання Марковських ланцюгів вищих порядків вимагає розв'язання таких основних трьох задач:

- визначення порядку Марковського ланцюга;
- визначення ймовірностей переходів між компонентами;
- обчислення надійності програмного продукту.

Ця робота присвячена дослідженню практичних аспектів використання Марковського ланцюга вищого порядку з метою підвищення адекватності прогнозування надійності програмного забезпечення. У першому розділі роботи описано використання критерію AIC для визначення порядку Марковського ланцюга, у другому розділі продемонстрована модифікована модель Гокаля вищого порядку, а в третьому розділі наведено шаблон проектування ПЗ, призначений для програмної реалізації Марковських ланцюгів вищого порядку, який дає змогу зберігати інформацію для використання таких ланцюгів.

Визначення порядку Марковського ланцюга

Для визначення порядку Марковського ланцюга запропоновано використовувати критерій AIC, що не є тестом перевірки гіпотези, не використовує рівень значущості. Окрім того, цей критерій дає стабільні результати і не залежить від порядку, в якому обчислюються моделі.

У загальному випадку AIC [7]:

$$AIC = 2k - 2\ln(L), \tag{1}$$

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

Розглянемо програмний продукт, що складається з S компонент (функціональні одиниці, які можна тестувати незалежно одна від одної). Вважаємо, що архітектура ПЗ моделюється Марковським процесом, який містить S станів (виконання відповідної компоненти).

У контексті моделювання Марковських ланцюгів вищого порядку Тонг [8] запропонував функцію ризику, що базується на AIC підході та дає змогу обчислити оптимальний порядок марковського ланцюга:

$$R(N) = -2\ln l_{N,M} - 2(S^{M+1} - S^{M} - (S^{N+1} - S^{N})), \tag{2}$$

де M – найвищий порядок моделі, N – порядок моделі, що розглядається. $l_{N,M}$ – відношення правдоподібності для порівняння гіпотези H_N та H_M (якщо $P_{ij...kl}$ – ймовірність переходів для ланцюга Маркова, де суфікс містить N+1 характеристику, тоді гіпотеза H_N : $P_{ij...kl} = P_{j...kl}$, $i = \overline{1,S}$; аналогічно визначається гіпотеза H_M). Значення змінної N, за якого функція $R(N)$ набуває свого найменшого значення є оптимальним порядком Марковського процесу.

Визначивши порядок Марковського процесу, можна використовувати відомий математичний апарат класичних Марковських процесів, оскільки будь-який Марковський процес вищого порядку можна подати через Марковський процес першого порядку.

Модифікована модель Гокаля вищого порядку

Для оцінювання надійності програмного продукту часто використовують модель Гокаля [9], в якій архітектура ПЗ зображується дискретним Марковським процесом. Ця модель є ієрархічною,

тобто спочатку обчислюються параметри архітектурної моделі, а потім враховується поведінка відмов кожної компоненти для оцінювання надійності програмного продукту.

Відповідно до моделі Гокаля надійність програмної системи обчислюють за формулою

$$R = \prod_{l=1}^{S} R_l \ , \tag{3}$$

де R_l – надійність кожної компоненти, S – кількість компонент ПЗ. Застосувавши для цієї моделі Марковський процес вищого порядку (нехай N – порядок моделі), надійність кожної компоненти знаходитиметься із:

$$R_l = e^{\sum_i V_{ij..kl} \cdot t_{ij..kl} \int_0 l_l(t)dt} \ . \tag{4}$$

У цій формулі $l_l(t)$ – інтенсивність відмов кожної компоненти, $V_{ij..kl}$ – очікувана кількість відвідувань компоненти l залежно від виконання попередніх N компонент, $t_{ij...kl}$ – час виконання компоненти l залежно від виконання попередніх N компонент.

Для того, щоб знайти очікувану кількість відвідувань відповідної компоненти залежно від виконання попередніх N компонент, необхідно спочатку знайти загальну кількість відвідування цієї компоненти за формулою [9]:

$$V_i = q_i + \sum_{j=1}^{S} V_j p_{ji}, i = \overline{1,S}, \tag{5}$$

де q_l – початковий вектор ймовірностей; p_{ij} – ймовірність переходу із компоненти i в компоненту j; V_i – очікуване число виконання відповідної компоненти.

Тоді $V_{ij...kl}$ обчислюєть за формулою

$$V_{ij...kl} = V_i p_{ij...kl}, \tag{6}$$

де $p_{ij...kl}$ – ймовірність переходу в компоненту l залежно від виконання попередніх N компонент.

Початковий вектор ймовірностей, матрицю ймовірності переходу між компонентами та час виконання компоненти можна отримати за допомогою тестувань програми, за допомогою запису протоколу виконання всіх компонент або за допомогою відповідної моделі використання ПЗ [10].

Для знаходження інтенсивності відмов кожної компоненти можна скористатись моделями “чорної скриньки”, побудованих на основі результатів модульного тестування: модель Гоеля-Окумото, Муси, S-подібну або модель з динамічним показником складності проекту [11].

Отримавши числові значення усіх параметрів моделі, можна обчислити надійність кожної компоненти за формулою (4) та значення надійності комп’ютерної програми, загалом (3).

Динамічне представлення Марковського ланцюга у прогнозуванні надійності ПЗ

Крім побудови моделі надійності ПЗ та визначення ортимального порядку Марковського ланцюга, є багато інших відкритих питань, пов'язаних з практичним використанням розроблених моделей для аналізу надійності програмного забезпечення, наприклад, питання поділу програмного забезпечення на компоненти і побудови граф потоку керування; питання отримання ймовірностей переходів та відносного часу виконання компоненти тощо.

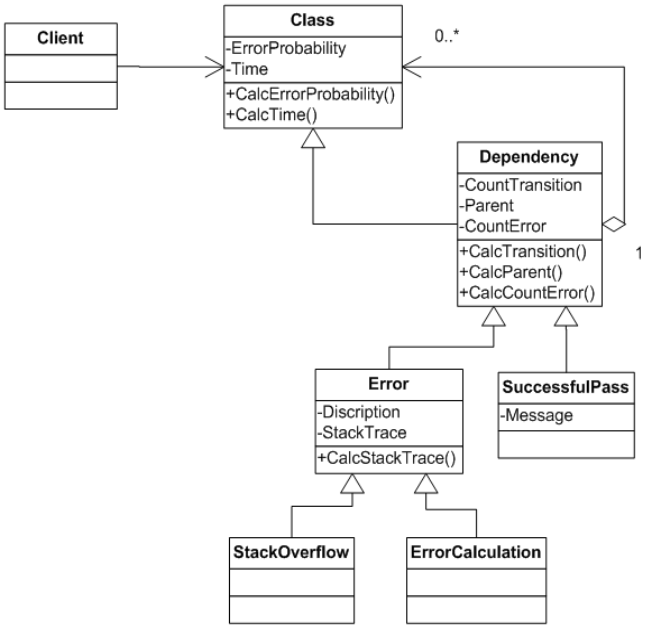
Сьогодні існують різні припущення про те, як поділити програмне забезпечення на компоненти, визначити параметри відповідної моделі надійності і застосувати її для оцінки надійності програмного продукту з використанням компонентного підходу [2, 3].

У моделях компонентного підходу, компонентом програмного забезпечення є логічно незалежний модуль системи, що виконує чітко визначені функції [12]. Тим не менш, наприклад, в роботах [3, 12] програмними компонентами є файли, що не є зовсім коректно, тому що один файл може містити декілька логічно незалежних частин, які взаємодіють одна з одною та можуть бути відтестовані незалежно.

Натомість пропонуємо використовувати клас як компонент програмного забезпечення, тому що це функціональна одиниця, що може бути незалежно перевірена за допомогою модульного тестування. Крім того, для класу можна знайти інтенсивність відмов, час, проведений в кожному класі і ймовірність переходів в інші класи [13].

Для побудови графа потоку керування, автор розробив шаблон на базі суфіксного дерева, який дозволяє створювати послідовності виконання компонент будь-якої довжини. Новий шаблон проектування програмного забезпечення побудований на основі шаблону “Компонувальник” [14], що використовується для опису складних систем, з модифікацією для представлення переходів у графові. Важливо, що шаблон не відображатиме неіснуючих послідовностей, на відміну від Марковських ланцюгів першого порядку.

Модифікована діаграма класів для шаблону “Компонувальник”, що може бути легко застосована для отримання графа потоку керування, зображена на рисунку.



Модифікований шаблон проектування, що дозволяє представити суфіксне дерево Марковського ланцюга вищого порядку

Внутрішні вершини графа міститимуть лічильник помилок і час виконання компоненти, кількість переходів буде зберігатись в класі Dependency, листки суфіксного дерева відображатимуть помилки або успішне виконання тесту. Також важливо зберігати посилання на батьківський клас, щоб мати можливість обчислити ймовірність переходів між компонентами. На етапі тестування ПЗ під час виконання нових тестів змінюватиметься лише лічильник переходів, а загальна структура буде мінятись лише тоді, коли створюватимуться нові послідовності виконання компонент. Отже, наш шаблон міститиме тільки необхідні послідовності вищого порядку, детальну інформацію про ймовірності переходів, ймовірності помилок та час виконання. Розроблений шаблон має такі переваги: нижчі вимоги до пам'яті та легка динамічна зміна порядку.

Висновки

У роботі проаналізовано використання Марковських ланцюгів вищого порядку для задачі моделювання надійності програмних систем у разі взаємозалежності виконання компонент. Показано, що використання критерію AIC дає змогу оптимізувати та формалізувати процес визначення оптимального порядку Марковського ланцюга для побудови моделі надійності програмного забезпечення. Запропоновано модифіковану модель Гокаля вищого порядку для оцінки надійності програмного продукту. Для компактного збору інформації для Марковського

ланцюга вищого порядку запроваджено новий шаблон проектування програмного забезпечення, що дає змогу витрачати менше пам'яті та динамічно змінювати порядок. Подальші дослідження будуть присвячені верифікації побудованої моделі, практичному застосуванню шаблона та порівнянню із існуючими альтернативними підходами.

1. Burkhart W., Fatiha Z. "Testing Software and Systems" // 23rd Ifip Wg 6.1 International Conference (2011), 236. 2. Goseva-Popstojanova K., Mathur A.P., Trivedi K.S. "Comparison of architecture–based software reliability models" // 12th International Symposium on Software Reliability Engineering (2001), 22-31. 3. K. Goševa-Popstojanova, S. Trivedi "Architecture-based approach to reliability assessment of software systems" // Performance Evaluation 4 (2001),179-204. 4. H. Pham "System Software Reliability" // Springer series in reliability engineering, Springer-Verlag London Limited (2006). 5. S. Krishnamurthy, A. Mathur "On the estimation of reliability of a software system using reliabilities of its components" // Proceedings of the Eighth International Symposium on Software Reliability Engineering (1997), 146–155. 6. T. Takagi, Z. Furukawa, T. Yamasaki "Accurate Usage Model Construction Using High-Order Markov Chains" // Supplementary Proceedings of 17th International Symposium on Software Reliability Engineering (2006), 1-2. 7. H. Akaike "A new look at the statistical model identification" // *IEEE Trans. Auto. Control. (1974), 716-723. 8. H. Tong, "Determination of the order of a Markov chain by Akaike's information criterion", Journal of applied probability 12 (1975), 488-497. 9. Gokhale S.S., Wong W.E., Horgan J.R., Kishor S. Trivedi "An analytical approach to architecture-based software performance reliability prediction" // Performance Evaluation 58, Issue 4 (2004), 391-412. 10. Heiko Koziolek "Operational Profiles for Software Reliability" // Dependability Engineering 2 (2005), 119-142. 11. Я.М. Чабанюк, В.С. Яковина, Д.В. Федасюк, М.М. Сенів, У.Т. Хімка "Побудова і дослідження моделі надійності програмного забезпечення з індексом величини проекту" // Інженерія програмного забезпечення 1 (2010), 24–29. 12. S.S. Gokhale, W.E. Wong, J.R. Horgan, S. Kishor, "An analytical approach to architecture-based software performance reliability prediction", Performance Evaluation 58 (4), 2004. 13. V. Yakovyna, I. Parfeniuk "Determination of transition probabilities between software components, written in java, based on monitoring of its execution. Proceedings of the XIIth International Conference" // CADSM'2013, 382 (2013). 14. Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.: Питер (2001). – 368 с.*