**УДК 004.052; 004.415.2**

**В.С. Яковина, Ю.І. Парфенюк**
Національний університет “Львівська політехніка”,
кафедра програмного забезпечення

# ВИКОРИСТАННЯ ЗАСОБІВ UML ДЛЯ ПРОГНОЗУВАННЯ НАДІЙНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ЕТАПІ ЙОГО ПРОЕКТУВАННЯ



**Перевірка відповідності вимогам на ранніх стадіях є критичним етапом у процесі розробки програмного забезпечення (ПЗ). Для перевірки функціональних вимог існує багато досліджень та рішень, проте для визначення надійності ПЗ на ранніх етапах розробки немає чітко сформованих підходів. Оскільки виявлення відхилень від вимог на ранніх етапах дасть змогу уникнути значних витрат для виправлення помилок на пізніших етапах, необхідно реалізувати можливість визначення показників надійності ПЗ на етапі збору вимог та раннього проектування системи. Запропоновані підходи оцінювання показників надійності на етапі проектування ПЗ на основі UML-діаграм.**

**Ключові слова: надійність програмного забезпечення, UML-діаграма, проектування програмного забезпечення, життєвий цикл програмного забезпечення.**

**Checking compliance to requirements on the early stages is a crucial issue of software development. There are a number of solutions to check the functional requirements, but there are no clear approaches to estimate software reliability at early lifecycle stages. Since revealing deviations from requirements at the early stages allows to avoid substantial amount of costs to correct errors at the later stages, it is necessary to realize the possibility of evaluation the software reliability at requirements analysis and system design stages. The approaches to evaluate software reliability using UML diagrams at software design stage are proposed in this paper.**

**Key words: software reliability, UML diagram, software design, software lifecycle.**

## Вступ

Програмне забезпечення (ПЗ) сучасної обчислювальної техніки набуває надзвичайно важливого значення у повсякденному житті. Мобільні пристрої, комп’ютери, системи життєзабезпечення, системи управління польотами, військові системи – управляються програмним забезпеченням. Саме тому гостро постає проблема забезпечення відповідного рівня надійності ПЗ.

Для розв’язання задач оцінки та прогнозування надійності сьогодні використовуються різноманітні підходи та розроблено багато моделей [1], за допомогою яких здійснюється аналіз та прогнозування надійності ПЗ на різних етапах його розроблення та життєвого циклу, а саме – на етапах проектування, розроблення, налагодження, перевірки, експлуатації та обслуговування [2].

Оскільки надійність лімітується внесеними дефектами, а вартість виправлення дефекту на ранніх етапах життєвого циклу є меншою, ніж на пізніших, задача оцінювання та аналізу надійності на ранніх етапах життєвого циклу є важливою для індустрії створення програмних продуктів, розв'язання якої дасть можливість оптимізувати витрати на створення ПЗ із заданими показниками якості. Крім того, аналіз надійності на етапі проектування дасть змогу усунути архітектурні помилки, виправлення яких є особливо трудомістким, а в деяких випадках – неможливим.

Саме тому розроблення методів і засобів аналізу та прогнозування надійності на ранніх етапах є надзвичайно актуальним науково-практичним завданням.

## Аналіз літературних джерел

Існуючі моделі аналізу та прогнозування надійності ПЗ поділяються на дві основні групи: моделі “чорної” (black) та “білої” скриньок (white box).

Дефект у разі програмного забезпечення – це некоректна логіка, некоректна команда чи невідповідна команда, яка під час виконання може спричинити відмову [1]. Інакше кажучи, дефект – це джерело відмов, а відмови – це реалізація дефектів. Коли відбувається відмова, їй відповідає деякий дефект у програмі, але наявність дефекту може не спричиняти відмови і програма ніколи не вийде з ладу, доки дефектні твердження не будуть виконані. Отже, на відміну від апаратного забезпечення, статистика відмов ПЗ повинна враховувати сценарії використання та покриття програмного коду тестами, в іншому ж випадку аналіз надійності ПЗ може призвести до некоректних результатів, навіть з використанням адекватного математичного апарата.

Для точнішого визначення надійності програмного забезпечення використовуються моделі “білої” скриньки, адже вони оцінюють внутрішню будову (архітектуру) програмного забезпечення, тому цей тип моделей називають моделі, побудовані на основі архітектурного підходу. Відповідно до цього підходу надійність ПЗ розглядається як функція надійності його складових: модулів, компонент, які можна розробляти та тестувати та досліджувати незалежно від інших, наприклад, клас [3].

Моделі на основі архітектурного підходу розглядають програму як мережу чи граф надійності. Вузлами такої мережі є модулі або підпрограми, а напрямлені дуги відображають послідовність виконання програми по модулях. Отримавши оцінку надійності кожного вузла, надійність переходів між вузлами, матрицю ймовірностей переходів в мережі та припускаючи незалежність відмов кожного вузла, можна отримати надійність програми загалом як вирішення проблеми надійності мережі [1–3].

Загальноприйнятим інженерним засобом, що використовується під час розроблення ПЗ, є мова UML [4]. На ранніх етапах життєвого циклу використовуються такі засоби UML: діаграма випадків використання на етапі збору вимог до ПЗ, діаграма класів та діаграма послідовності дій – на етапі проектування архітектури ПЗ.

UML-діаграма випадків використання призначена для описання функціональних можливостей програмного забезпечення. Існує багато робіт, що розглядають використання інструментарію UML [5–7], зокрема діаграм випадків використання у задачах оцінювання та прогнозування надійності ПЗ [7].

Для проектування компонент системи та взаємодії між ними використовується UML-діаграма послідовності дій. Використання цього типу діаграми за аналізу надійності ПЗ було описано у [5], де був проведений аналіз UML-діаграми послідовності для визначення надійності ПЗ на основі сценаріїв, компонент, часу зайнятості компоненти та ймовірності відмови під час виконання певного методу.

## Постановка задачі

У попередніх роботах [8] показана доцільність використання модифікованої моделі Гокаля [9] для визначення показників надійності ПЗ. Для проведення аналізу та оцінювання надійності ПЗ з використанням цієї моделі необхідно отримати такі вхідні дані:

- ймовірність переходу із компоненти у компоненту;
- інтенсивність відмов кожної компоненти;
- очікувана кількість відвідувань відповідної компоненти залежно від перебування у попередніх компонентах;
- початковий вектор ймовірностей;
- час перебування у компоненті залежно від перебування у попередніх компонентах.

Початковий вектор ймовірностей, матрицю ймовірності переходу між компонентами та час перебування у компоненті можна отримати за допомогою моніторингу роботи ПЗ [10]. Проте немає розробленого підходу отримання цих вхідних даних на етапі збору вимог та проектування ПЗ.

Отже, метою цієї роботи є розроблення підходу для отримання структури та параметрів моделі надійності програмного забезпечення марковського типу на ранніх етапах його розроблення з використанням засобів мови UML.

**Виклад отриманих результатів**

Одним із перших етапів життєвого циклу розроблення ПЗ є збір та аналіз вимог. Оскільки стандартною практикою аналізу вимог до ПЗ є використання мови UML [4], а саме – діаграми випадків використання, доцільним є використання цього інженерного засобу для аналізу надійності ПЗ, зокрема для наближеного отримання значень матриці ймовірностей переходів між компонентами.

Оскільки на етапі аналізу вимог наявний мінімальний обсяг інформації і фактично відсутня архітектура ПЗ, доцільно використати методи системного аналізу [11] для отримання числових значень параметрів майбутньої системи, зокрема відносної частоти використання певних сценаріїв, на основі яких можна оцінити значення матриці ймовірностей переходів між компонентами. Проте існуючі моделі та підходи не передбачають отримання інформації від замовника як основного експерта, який встановлює вимоги до системи.

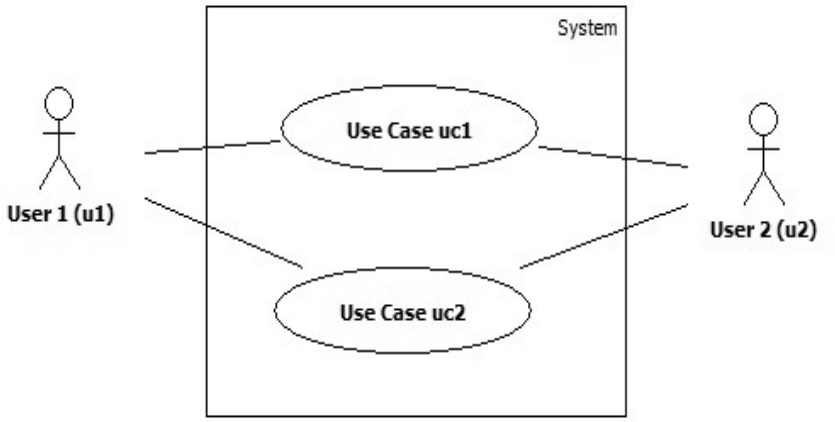


*Рис. 1. Приклад UML-діаграми випадків використання*

Введемо такі позначення [7] (рис. 1): $q_1$ та $q_2$ – ймовірності використання системи користувачами $u_1$ та $u_2$ відповідно; $P_{11}$ та $P_{12}$ – ймовірності, що користувач $u_1$ використовує функціональні можливості $f_1$ та $f_2$. Імовірності $P_{21}$ та $P_{22}$ визначаються аналогічно для користувача $u_2$. У такому випадку ймовірність виконання випадку використання $x$ обчислюється так:

$$P(x) = \sum_{i=1}^{m} q_i \cdot P_{ix} \,, \tag{1}$$

де $m$ – кількість типів користувачів (акторів).

На основі ймовірностей (або відносних частот) виконання усіх випадків використання можна оцінити ймовірності виконання компонент ПЗ та переходів між ними. Необхідно пам’ятати, що на цьому етапі життєвого циклу можна тільки наближено говорити про компоненти ПЗ (а згадані імовірності отримують апріорно на основі експертних оцінок). Підхід, описаний у [7], потребує знання числових значень ймовірностей використання системи кожним типом акторів, а також ймовірностей виконання кожного випадку використання типами користувачів. Проте на цьому етапі життєвого циклу розробник не має необхідних даних, а для замовника, який, як вже зазначалось, є основним експертом з випадків використання системи, задача оцінювання таких ймовірностей виявляється занадто складною. Для усунення цього обмеження запропоновано використати метод аналізу ієрархій (MAI), який передбачає декомпозицію задачі та використання експертом попарних порівнянь [11].

На рис. 2 показано ієрархію для визначення ймовірностей виконання двох випадків використання двома акторами, які відповідають діаграмі на рис. 1.

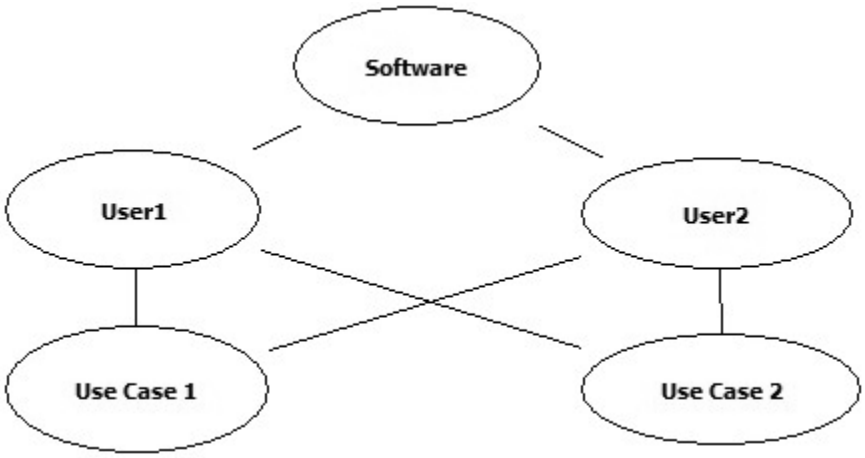


*Рис. 2. Ієрархічна структура визначення ймовірності виконання випадків використання*

Відповідно до [12], опитування замовника (експерта) здійснюється у вигляді анкетування за допомогою попарного порівняння згідно зі шкалою відносної важливості методу аналізу ієрархій. Опитування передбачає попарне порівняння вузлів рівнів ієрархії. Наприклад, користувачів User1 та User2, а також випадків використання Use Case 1 та Use Case2. Під попарним порівнянням мають на увазі відносну частоту використання програмного продукту певними акторами та відносну частоту виконання заданим актором певних сценаріїв використання. На основі анкетування заповнюється матриця попарних порівнянь, на основі якої здійснюється розрахунок ймовірностей згідно з методом аналізу ієрархій.

Для спрощення опитування замовника можливе використання модифікації MAI, яка ґрунтується на порівнянні за стандартами [11]. Для спрощення анкетування можна ввести, наприклад, стандарти пріоритетів використання: High (H) – високий, Medium (M) – середній та Low (L) – низький. Під час використання стандартів необхідно у відповідний спосіб змінити ієрархію, увівши рівень стандартів. Використання подібної системи пріоритетів є поширеною практикою в індустрії розробки ПЗ під час побудови специфікацій програмних продуктів та аналізу вимог.

Розробляючи архітектуру об’єктно-орієнтованих систем, використовується UML-діаграма класів (рис. 3). За допомогою діаграми класів здійснюється опис компонент системи, а також зв’язків між ними.

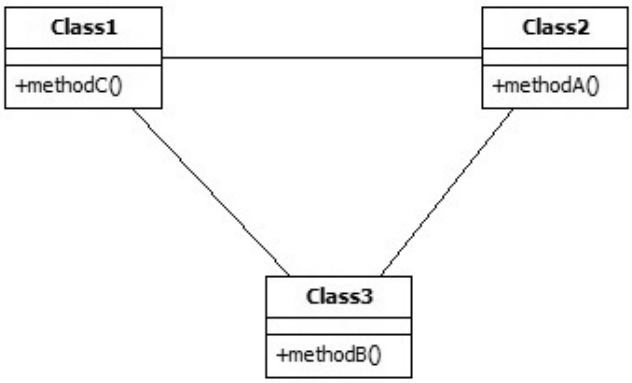


*Рис. 3. UML-діаграма класів*

З діаграми класів можна сформувати матрицю ймовірностей переходів між компонентами, а саме – її структуру, проте необхідно зауважити, що у цьому випадку неможливо отримати ймовірності переходів та час перебування у компонентах, оскільки така діаграма призначена лише для розроблення архітектури системи, а не відображення поведінки користувачів.

Для проектування компонент системи та взаємодії між ними використовується UML-діаграма послідовності дій. У цій роботі діаграма послідовності дій використовується для оцінки кількості переходів між компонентами, з яких можна визначити матрицю ймовірності переходів між компонентами, а також час зайнятості [7] – час перебування у компоненті для обчислення надійності системи на основі моделі марковського типу [8].

На рис. 4 продемонстровано приклад UML-діаграми послідовності дій із зазначенням необхідних показників переходів між компонентами та відносного часу перебування у компоненті.

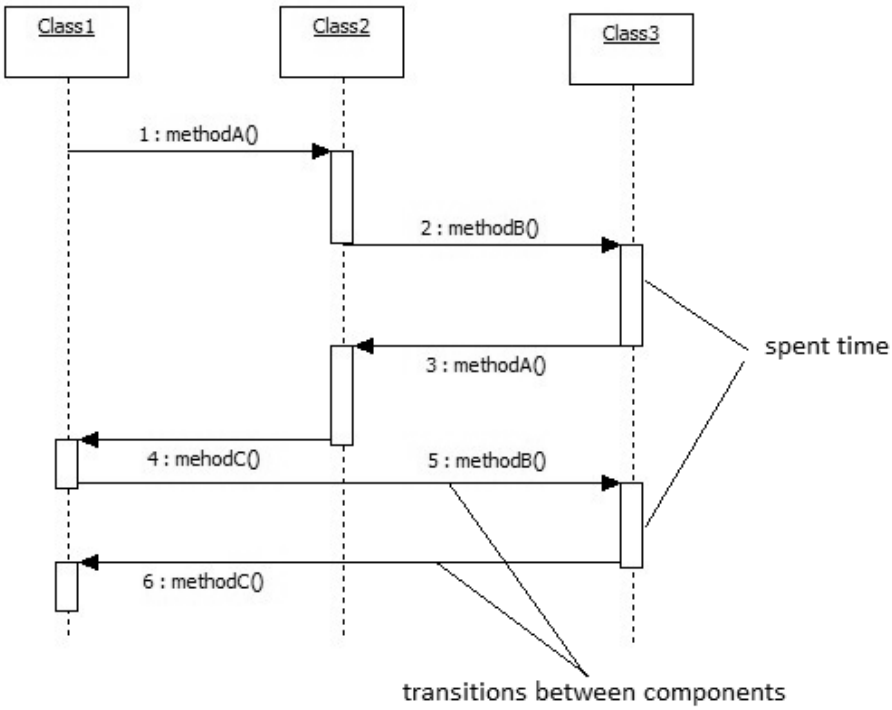


*Рис. 4. UML-діаграма послідовності дій*

Провівши аналіз розроблених UML-діаграм послідовності дій під час проектування ПЗ необхідно порахувати сумарну кількість переходів з компоненти у компоненту і сформувати матрицю переходів між компонентами відповідно до підходу, описаного у [10]. Для визначення часу перебування у кожній компоненті ПЗ необхідно просумувати час перебування у компоненті. Як компоненту пропонується використовувати клас, який є основною складовою під час розроблення ПЗ, відповідно до об’єктно-орієнтованого підходу, та може бути незалежно протестований. Під час переходу між компонентами (класами) виконуються виклики відповідних методів класу, час перебування у компоненті визначається часом виконання методу.

Для того, щоб отримати матрицю ймовірностей переходів усієї системи, необхідно в подібний спосіб урахувати усі сценарії використання системи, оскільки діаграма послідовності зазвичай будується для кожного сценарію використання системи, а імовірність виконання кожного сценарію можна отримати, наприклад, з (1).

## Висновки

На етапі проектування ПЗ важливо мати можливість аналізувати та прогнозувати надійність ПЗ, адже помилки, допущені на етапі проектування, важко виправляються на подальших етапах життєвого циклу ПЗ. Саме тому розроблено підхід до визначення надійності на етапі описання архітектури ПЗ за допомогою UML-діаграм, а саме: діаграми випадків використання, діаграми класів та діаграми послідовності дій. Використання цих діаграм дає можливість оцінити імовірність виконання кожного сценарію використання програмної системи, отримати структуру марковської моделі на основі діаграми класів, та отримати початкову оцінку значень параметрів моделі на

основі діаграми діяльності для кожного сценарію використання ПЗ. Відповідно до розробленого підходу, можна оцінити значення елементів матриці ймовірностей переходів між компонентами та час перебування у кожній компоненті. Ці показники використовуються для обчислення надійності ПЗ на основі модифікованої моделі Гокаля.

Такий підхід дає можливість визначати та уточнювати матрицю ймовірностей переходів між компонентами ПЗ та час перебування у компоненті на ранніх етапах життєвого циклу програмного забезпечення, що повинно скоротити витрати на розроблення ПЗ із заданим ступенем надійності.

*1. Бобало Ю.Я. Математичні моделі та методи аналізу надійності радіоелектронних, електротехнічних та програмних систем: монографія / Ю.Я. Бобало, Б.Ю. Волочій, О.Ю. Лозинський, Б.А. Мандзій, Л.Д. Озірковський, Д.В. Федасюк, С.В. Щербовських, В.С. Яковина. – Львів: Видавництво Львівської політехніки, 2013. – 300 с. 2. Xie M. Software Reliability Modelling // World Scientific, Singapore, 1991. 3. Yakovyna V., Serdyuk P., Nytrebych O. Markov chain dynamic representation model for reliability testing // Proceedings of the 12th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics CADSM 2013, Lviv–Polyana, 2013. – P. 384–385. 4. Grady Booch, Ivar Jacobson & Jim Rumbaugh OMG. Unified Modeling Language Specification, Version 1.3 First Edition: March 2000. Retrieved 12 August 2008. 5. Singh H., Cortellessa V., Cukic B., Gunel E. and Bharadwaj V. A bayesian approach to reliability prediction and assessment of component based systems. In Proc. of 12th International Symposium on Software Reliability Engineering (ISSRE'01), 2001. 6. Cortellessa V. and Mirandola R. Deriving a queueing network based performance model from UML diagrams. In Proc. of 2nd IEEE Workshop on Software and Performance. (WOSP2000), September 2000. 7. Cortellessa V., Singh H., Cukic B. Early reliability assessment of UML based software models. In: Proceedings of the 3rd international workshop on Software and performance, Rome, Italy, 2002. 8. Yakovyna V., Nytrebych O., Fedasyuk D. Modified High-Order Software Reliability Gokhale Model // Proceedings of the VII th International Scientific and Technical Conference "Computer Science and Information Technologies" CSIT'2012, Lviv, 2012. – P. 194–195. 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. Yakovyna V., Parfeniuk I. Determination of transition probabilities between software components, written in Java, based on monitoring of its execution // Proceedings of the 12th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics CADSM 2013, Lviv–Polyana, 2013. – P. 382–383. 11. Катренко А.В. Системний аналіз об'єктів та процесів комп'ютеризації / А.В. Катренко. – Львів: Новий світ, 2000. – 424 с. 12. Yakovyna V., Parfeniuk u. Evaluation Matrix Transition Probabilities between Software Components based on UML Use Case Diagram // Proceedings of the VII th International Scientific and Technical Conference "Computer Science and Information Technologies" CSIT'2012, Lviv, 2012. – P. 196–197.*