4. ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Науковий вісник НЛТУ України
Scientific Bulletin of UNFU
https://nv.nltu.edu.ua
https://doi.org/10.15421/40290525
Article received 22.05.2019 p.
Article accepted 30.05.2019 p.
УДК 004.[052+416.2]

ISSN 1994-7836 (print)
ISSN 2519-2477 (online)

@ Correspondence author
V. S. Yakovyna
vitaliy.s.yakovyna@lpnu.ua

В. С. Яковина, Б. В. Угриновський
Національний університет "Львівська політехніка", м. Львів, Україна

СТАРІННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В КОНТЕКСТІ ЙОГО НАДІЙНОСТІ: ОГЛЯД ПРОБЛЕМАТИКИ

Проведено огляд та аналіз літературних джерел, в яких досліджено явища старіння програмного забезпечення. Процес старіння охарактеризовано як погіршення продуктивності і збільшення кількості відмов, що має негативний вплив на показники надійності програмного забезпечення. Встановлено, що помилки програмного забезпечення та їх накопичення протягом виконання програми є причиною виникнення старіння програмного забезпечення. Визначено основні поняття та характеристики, що стосуються явища старіння, зокрема ефекти, чинники та метрики старіння, час до виснаження ресурсів, час до відмови старіння та робоче навантаження. Розглянуто класифікацію чинників старіння програмного забезпечення. Встановлено, що чинники можуть бути загальні для всіх систем і спеціальні для конкретних систем, зокрема мобільних. Здійснено порівняльний аналіз основних методів та підходів до моделювання процесу старіння програмного забезпечення. З'ясовано, що розроблення гібридних підходів та моделей, які включають переваги аналітичних моделей та моделей на основі вимірювань, є перспективним напрямом у вивченні проблеми старіння ПЗ. Показано, що мобільні операційні системи та додатки є особливо чутливими до ефектів старіння, оскільки вони працюють тривалий час без перезавантаження та часто мають обмежені ресурси, такі як пам'ять. Обгрунтовано актуальність урахування впливу цього явища для забезпечення надійності сучасних мобільних і вбудованих систем.

Ключові слова: відмова старіння; помилка старіння; чинники старіння; час до виснаження ресурсу; ланцюг Маркова; часові ряди.

Вступ. Старіння програмного забезпечення (ПЗ) – це погіршення продуктивності ПЗ із моменту його запуску на пристрої, що може призвести до його повного припинення функціонування та необхідності перезапуску (Grottke et al., 2008). Отже, старіння ПЗ призводить до погіршення характеристик надійності. У сучасному світі глобалізації та Інтернету речей програмне забезпечення стало одним з найважливіших елементів для забезпечення роботи компаній та організацій у їхньому бізнес-середовищі. Надійність (Polovko & Gurov, 2008) є критичною властивістю більшості сучасної техніки загалом і ПЗ зокрема. Питання підвищення рівня надійності є ключовим для сучасного етапу розвитку програмно-апаратних систем.

Поняття старіння ПЗ впроваджено в середині 90-х років XX ст. (Huang et al., 1995; Parnas, 1994) та традиційно розглядається в контексті інженерії ПЗ та надійності (Ahamad, 2016). Ранні свідчення старіння ПЗ було знайдено вже в 1960-х роках: військова система Safeguard зазнала впливу зависань, що відбувалися після того, як були заповнені буфери повідомлень про помилки (Bernstein & Kintala, 2004). Вивчення цього явища зосереджено як на теоретичному (Shereshevsky et al., 2003; Wang et al., 2007), так і на емпіричному (Grottke et al., 2006; Matias & Freitas, 2006) особливостях ефектів старіння.

Старіння ПЗ проявляється на різних типах систем та програмному забезпеченні, зокрема і в мобільних додатках таких систем, як Android (Maji et al., 2010; Cotroneo et al., 2016) і iOS. Дослідження процесу старіння ПЗ мобільних і вбудованих систем перебувають на ранній стадії, тому метою цієї роботи є аналіз явища та визначення характеристик, чинників та моделей і методів дослідження процесу старіння ПЗ, виявлення актуальних проблем в області старіння ПЗ, зокрема для мобільних платформ.

Явище старіння в контексті надійності ПЗ. Для пояснення явища старіння наведемо основні поняття та концепції теорії надійності, оскільки старіння є наслідком дефектів та помилок ПЗ, які спричиняють погіршення показників його надійності.

Відмова (Polovko & Gurov, 2008) системи – це відхилення у роботі послуги, що надається системою, від її специфікації. Таке відхилення може бути у формі неп-

Інформація про авторів:
Яковина Віталій Степанович, д-р техн. наук, професор, завідувач кафедри програмного забезпечення.
Email: vitaliy.s.yakovyna@lpnu.ua; https://orcid.org/0000-0003-0133-8591
Угриновський Богдан Володимирович, аспірант, кафедра програмного забезпечення. **Email:** bohdanuhryn@gmail.com
Цитування за ДСТУ: Яковина В. С., Угриновський Б. В. Старіння програмного забезпечення в контексті його надійності: огляд проблематики. Науковий вісник НЛТУ України. 2019, т. 29, № 5. С. 123–128.
Citation APA: Yakovyna, V. S., & Uhrynovskyi, B. V. (2019). Software Aging in the Context of its Reliability: a Systematic Review. *Scientific Bulletin of UNFU, 29(5)*, 123–128. https://doi.org/10.15421/40290525

равильного обслуговування або взагалі відсутності обслуговування. Система також може бути в деградованому режимі, в якому служба повільна або обмежена, що називають *частковою відмовою*. В (Avizienis et al., 2004) проблема відмови обговорюється з точки зору "*ланцюга загроз*" (рис. 1), що показує причинно-наслідкові зв'язки між дефектом (fault), помилкою (error) і відмовою (failure).

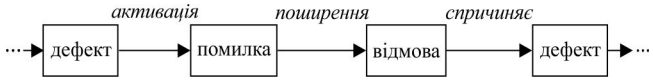

Рис. 1. Ланцюг загроз

Помилка – це частина внутрішнього стану системи, яка може призвести до виникнення відмови. Помилки можуть бути перетворені в інші помилки; цей процес називають *поширенням помилок*. Поширення помилок призводить до збою системи, якщо помилка поширюється на інтерфейс сервісу системи, спричиняючи відхилення наданого сервісу від специфікації.

Водночас, кожна помилка викликана активацією *дефекту*. Дефект, який в даний час не призводить до помилки, вважають *прихованим*. Якщо певний набір вхідних даних є достатній для активації прихованого дефекту, то це може призвести до виникнення помилки, що називають *активацією дефекту*. Швидкість, з якою активується прихований дефект, залежить значною мірою від інтенсивності та способу використання системи, а кількісну характеристику останнього аспекту називають *операційним профілем* (Musa, 1993).

У (Avizienis et al., 2004) представлено схему класифікації дефектів за вісьмома критеріями, наприклад: системні межі (внутрішні або зовнішні), феноменологічна причина (природна чи техногенна), ціль (зловмисна чи не зловмисна), стійкість (постійні або тимчасові), розмірність (апаратне чи програмне забезпечення), а також етап створення або виникнення (розроблення або функціонування). Виходячи з цієї класифікації, помилки в програмному коді називають помилками ПЗ чи програмними помилками, що можуть бути описані як внутрішні, спричинені людиною, не зловмисні, постійні помилки ПЗ.

Групи помилок Mandelbug і Bohrbug (Grottke & Trivedi, 2005) класифікують типи несправностей з використанням більш об'єктивних критеріїв, пов'язаних з властивостями самої несправності. Помилки Mandelbug важко ізолювати і можуть спричиняти відмови, які відтворюються не систематично. Bohrbug, з іншого боку, є легко ізольованою несправністю, яка завжди проявляється послідовно під чітко визначеним набором умов.

Явище старіння ПЗ полягає в тому, що зі збільшенням періоду виконання системи або процесу її інтенсивність відмов зростає, що може бути пов'язано з накопиченням помилок у стані системи та споживанням таких ресурсів системи, як фізична пам'ять (Grottke et al., 2008).

Багато відмов старіння насправді є наслідком дефектів ПЗ. На рис. 2 показано модифіковану версію загального "ланцюга загроз", специфічну для відмов, пов'язаних із старінням.

Поширення *помилок, пов'язаних зі старінням* (тобто помилок, які можуть викликати відмови старіння), вимагає, щоб стан внутрішнього середовища системи відповідав певним критеріям. Більшість помилок старіння, які ще не викликають відмову, зберігаються у внутрішньому стані системи, де такі помилки накопичуються, якщо є послідовні активування *дефектів, пов'язаних зі старінням*. Зазвичай, саме це накопичення помилок старіння приводить внутрішнє середовище системи до стану, в якому розповсюджуються помилки старіння, що викликають відмови старіння.

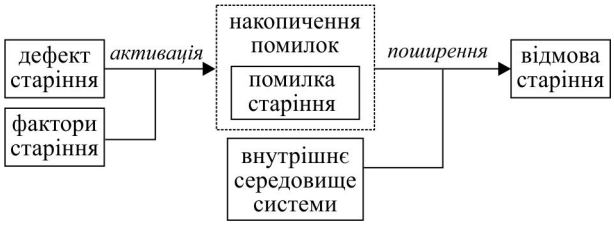

Рис. 2. Версія "ланцюга загроз", специфічна для відмов старіння

Використовуючи схему класифікації в (Avizienis et al., 2004), дефекти, пов'язані зі старінням, зазвичай є спричинені людьми, не зловмисними, постійними помилками розроблення ПЗ.

Усі помилки старіння належать групі помилок Mandelbug з двох причин. По-перше, може бути тривала затримка між активацією несправностей і остаточним виникненням відмови і саме така затримка дає змогу накопичувати помилки. По-друге, виникнення відмови, спричинене накопиченням помилки та поширенням помилок, може залежати від внутрішнього середовища системи.

Поняття та характеристики старіння ПЗ. Схеми активації дефектів, пов'язаних зі старінням (тобто чинники або їх комбінації, які активують цю помилку), називають *чинниками старіння* (Grottke et al., 2008; Abdullah et al., 2017). *Внутрішні* чинники старіння є внутрішніми подіями (наприклад, функція викликає спрацьовування виконання тих частин коду, де знаходиться помилка, пов'язана зі старінням), *зовнішні* чинники старіння – це тригери, які безпосередньо виконуються елементами в середовищі системи, наприклад, користувачами системи.

Накопичення помилок зазвичай спричиняється поганим управлінням ресурсами, що призводить до вичерпання ресурсів, таких як витік пам'яті, не перервані потоки та не звільнені блокування файлів: у таких випадках очікуваний час на відмову старіння називають *часом до виснаження* (time to exhaustion) (Grottke et al., 2008). Окрім цього, на накопичення помилок впливає кількість і тип роботи, що виконується системою, яку називають *робочим навантаженням* (workload).

Проміжок часу до відмови (Grottke et al., 2008), пов'язаної із старінням, є випадковим періодом часу від моменту запуску системи або створення процесу до виникнення відмови, пов'язаної з помилками старіння. Розподіл ймовірності цього випадкового часу (так само як і його очікуване значення, середня тривалість до відмови старіння) в основному залежить від інтенсивності, з якою система піддається впливу чинників старіння. Отже, на нього впливають кількість і тип робіт, що виконуються процесом, що називають *робочим шляхом* процесу, а отже, робочим профілем і інтенсивністю використання системи.

Ефект старіння (Abdullah et al., 2017) ПЗ полягає в поступовому переході від правильного внутрішнього стану до ймовірно помилкового і помилкового, коли послідовна активація дефектів, пов'язаних зі старінням, викликає накопичення помилок старіння.

Ефекти старіння в системі можна виявити тільки під час роботи системи, контролюючи *показники старіння* (Grottke et al., 2008). Показники старіння є маркерами для виявлення старіння. Показники старіння – це змінні, які індивідуально або в комбінації можуть свідчити про стан системи. Вони можуть розглядатися на декількох рівнях, таких як операційна система, процес додатка, компонент програми, проміжне ПЗ, віртуальна машина (VM) і монітор VM (VMM).

Хоча у багатьох випадках старіння ПЗ зумовлено помилками, пов'язаними зі старінням, навіть за відсутності таких несправностей в коді, ефекти старіння можуть виникати як наслідок природної динаміки поведінки системи. Цей вид старіння називають *природним старінням* (Grottke et al., 2008). Прикладами природного старіння є проблеми фрагментації, які стосуються файлових систем, індексних файлів баз даних, а також основної фізичної пам'яті. Такі ефекти старіння не пов'язані з несправним кодом або дизайном архітектури, але вони є наслідком використання системи чи додатків протягом свого життя.

Згідно з (Grottke et al., 2008), ефект старіння не є оборотним без зовнішнього втручання. Згідно з цією характеристикою, старіння ПЗ може бути прибрано чи відтерміновано шляхом зовнішнього втручання. Зокрема, шляхом виявлення причин та чинників, які сприяють феномену старіння ПЗ, можна знаходити рішення для затримки або уникнення процесу старіння.

Техніку запобігання і затримки старіння називають процесом *омолодження ПЗ* (Huang et al., 1995). Цей підхід передбачає регулярні чи нерегулярні скидання внутрішнього стану системи так, щоб очистити накопичені помилки. Омолодження ПЗ може бути реалізовано на декількох рівнях деталізації та застосовано до багатьох типів елементів у системі, таких як операційна система, окремі процеси або об'єкти даних.

Чинники старіння ПЗ. У статті (Abdullah et al., 2017) чинники старіння ПЗ поділяють на два типи: зовнішні та внутрішні (табл. 1).

Табл. 1. Зовнішні та внутрішні чинники старіння ПЗ

Зовнішній чинник	Внутрішній чинник
Динамічне середовище	Дефекти відналагодження
Покращення та зміни функцій	Залишкові дефекти
Стабільність і послідовність бізнесу	Витоки пам'яті
Еволюція вимог	Переповнення пам'яті
Вартість обслуговування	Не звільнені блокування файлів
Людина	Накопичення помилок округлення

Внутрішні чинники переважно пов'язані з помилками у програмному коді та архітектурі ПЗ, а зовнішні чинники – зі старінням ПЗ відносно зовнішнього середовища. У роботі (Abdullah et al., 2015) зовнішні чинники групують в три основні класи: функціональні чинники, людський чинник, чинники середовища. У табл. 2 наведено ці класи чинників старіння та метрики, які можна використовувати для вимірювання рівня старіння ПЗ.

Ще одним чинником старіння ПЗ є метрики програмного коду. Метрики ПЗ – це числова міра складності програми, що залежить від властивостей тексту програми (Basili & Perricone, 1984) і обчислюється статично без необхідності виконання програми. Розмір ПЗ, його складність, використання деяких видів структур програмування, пов'язаних з управлінням ресурсами, використання арифметичних операторів та інші особливості пов'язані з виникненням помилок старіння. У роботі (Cotroneo et al., 2011) підтверджується, що ефекти старіння ПЗ пов'язані зі статичними характеристиками ПЗ.

Табл. 2. Чинники і метрики старіння ПЗ

Чинники	Метрики
Функціональні	Продуктивність ПЗ, корисність, служба підтримки, час відгуку, помилки ПЗ
Середовища	Служба підтримки, бізнес вимоги, зміни в бізнес процесах, зміни середовища, зміни технологій (програмних та апаратних)
Людські	Навчання, служба підтримки, популярність і технології, зміни у вищому керівництві, досвід

Різні досліджувані системи можуть мати власні набори чинників, які специфічно впливають на явище старіння ПЗ. Наприклад, у роботі (Cotroneo et al., 2016) для дослідження процесу старіння в операційній системі Android сформовано набір чинників, що охоплює: 1) набори додатків для дослідження; 2) фізичні пристрої, що мають різні характеристики; 3) робоче навантаження – запуски і завершення додатків; 4) робоче навантаження – події введення; 5) вільний для використання простір файлового сховища.

Моделі та методи дослідження процесу старіння ПЗ. Згідно з (Cotroneo et al., 2014), всі дослідження процесу старіння ПЗ можна розділити на дослідження на основі моделей, дослідження на основі вимірювань, гібридні дослідження, дослідження стратегій планування омолодження ПЗ, дослідження методів омолодження і дослідження даних відмов реального ПЗ. Ще однією класифікацією досліджень (Valentim et al., 2016) є їх поділ на аналітичні, емпіричні та гібридні.

Дослідження на основі моделей аналітично моделюють явище старіння ПЗ для того, щоб забезпечити абстрактне уявлення про нього і зробити його застосовним для математичного оброблення. Існує багато типових моделей старіння ПЗ, як правило, на основі ланцюгів Маркова.

Марковські та напівмарковські процеси з їх варіантами сьогодні є основою для багатьох моделей. Перша робота (Huang et al., 1995) про старіння ПЗ, яка доводила, що омолодження ПЗ може зменшити вартість простою системи, моделювала явище старіння за допомогою ланцюга Маркова з неперервним часом (Continuous-Time Markov Chain – CTMC), який і досі є базовою моделлю для опису цього явища.

Марковські ланцюги широко використовуються для аналізу складніших систем та опису складніших проявів відмов. Наприклад, у (Xie et al., 2004) з допомогою CTMC описано поведінку кластерних систем, а в (Okamura & Dohi, 2011) розглянуто різні ступені відмов для моделювання поступового зниження продуктивності, що також описується з допомогою CTMC. Марковські ланцюги також застосовуються для аналізу старіння в нових контекстах та системах, зокрема таких як мобільні системи. Так, в (Xianga et al., 2019) побудовано інтегровану CTMC на основі моделі поведінки користувача та моделі процесу старіння.

Хоча більшість досліджень застосовують класичні марковські і напівмарковські процеси, є й інші типи моделей. Наприклад, автори (Wang et al., 2007) використовували Марковські регенераційні процеси (MRGP) у поєднанні зі стохастичними мережами Петрі (SPN) для

побудови загальної моделі оцінки оптимального графіка омолодження в програмній системі.

Проблема знаходження оптимального графіка омолодження сформульована також як Процес прийняття рішень Маркова (*Markov Decision Process* – MDP). Наприклад, у (Pfening et al., 1996) застосовано MDP для побудови моделі омолодження ПЗ в телекомунікаційній системі, що містить виникнення переповнення буфера. В (Okamura & Dohi, 2011) використано частково спостережувані процеси прийняття рішень Маркова (*Partially Observable Markov Decision Processes* – POMDP).

Мережі Петрі та їх численні варіанти – це формалізми, суворо пов'язані з марковськими моделями, але більш компактні і в деяких випадках легші для визначення. Цей формалізм дає змогу більш легко виразити показники продуктивності з кількома рівнями і є особливо корисним для вираження метрик у складніших системах, таких як кластерні системи (Wang et al., 2007), (Vaidyanathan et al., 2001).

Дослідження на основі вимірювань (Cotroneo et al., 2014) показників реальних систем дають змогу виконувати емпіричний аналіз старіння ПЗ для того, щоб визначити, чи знаходиться система в стані, схильному до відмов через старіння ПЗ, для прогнозування часу до відмови через старіння і для планування омолодження програми. Дослідження, засновані на вимірюваннях, також надають детальну інформацію про явища старіння в реальних системах, що корисно для кращого розуміння природи та масштабів старіння ПЗ. Можна виділити такі групи підходів для дослідження процесу старіння на основі вимірювань: аналіз часових рядів, машинне навчання, підходи на основі порогів.

Аналіз часових рядів широко застосовується під час моніторингу ресурсів для того, щоб підтвердити чи спростувати наявність чи тенденцію до виникнення старіння. Наприклад, у роботі (Cotroneo et al., 2016) представлено експериментальну методологію для аналізу проблем старіння ПЗ в операційній системі Android, яка використовує статистичні методи тест Манна-Кендала та процедуру оцінки нахилу Сена, щоб визначити, які чинники посилюють погіршення продуктивності та споживання ресурсів. Моделі часових рядів ARMA та ARX використовувалися в (Li et al., 2002) на веб-сервері Apache, щоб оцінити вичерпання ресурсів.

Аналіз часових рядів також застосовується для вивчення взаємозв'язку явища старіння з робочим навантаженням у складних системах. Наприклад, у Linux системі (Cotroneo et al., 2010) аналіз параметрів робочого навантаження дає змогу виділити підсистеми, параметри яких корелюють з тенденціями старіння, що свідчить про потенційні джерела старіння. Моделі ARIMA (*Autoregressive Integrated Moving Average*) та Холта-Вінтерса були використані в (Magalhaes & Silva, 2010) для виявлення аномалій продуктивності, викликаних старінням, які спрямовані на веб та компонентні додатки.

Машинне навчання. Підходи машинного навчання є досконалішою формою аналізу даних, які застосовують алгоритми з області штучного інтелекту для виявлення тенденцій і класифікації стану системи як надійної або схильної до відмови.

У роботі (Huo et al., 2018) проведено порівняння ефективності трьох алгоритмів машинного навчання (дерево рішень (*Decision Tree*), метод опорних векторів (*Support Vector Machine*), глибока мережа довіри (*Deep Belief Network*) для виявлення старіння ПЗ в системі Android.

Інший приклад застосування підходу машинного навчання для прогнозування відмов старіння наведено в (Alonso et al., 2011) у контексті трирівневої системи J2EE, де виконано аналіз кількох алгоритмів (дерево прийняття рішень, алгоритми лінійного та квадратичного аналізу, наївний алгоритм Байєса, метод опорних векторів та метод k-найближчих сусідів). Результати роботи показують ефективність методів машинного навчання для розробки засобів моніторингу та вдосконалення методів омолодження ПЗ.

Підходи на основі порогів визначають пороги для деяких показників старіння і омолодження викликається тоді, коли контрольовані показники перевищують такі пороги. Наприклад, показники можуть стосуватися споживання ресурсів. Труднощі виникають у визначенні найкращих показників і правильних порогів для них. Прикладом такого підходу є робота (Silva et al., 2009), яка застосовує пороги щодо середнього часу відгуку та показників якості послуг.

Гібридні дослідження поєднують переваги підходів на основі моделей і вимірювань: описують явище аналітично, найчастіше за допомогою марковських моделей, і визначають параметри моделі за допомогою вимірювань системних показників. Незважаючи на практичну важливість гібридних підходів, у цьому напрямку зроблено невелику кількість досліджень (Cotroneo et al. 2014).

У (Vaidyanathan & Trivedi, 2005) представлено результати аналізу, які враховують навантаження на систему та будують модель для оцінки часу вичерпання ресурсів. У роботі побудовано напівмарковську модель, засновану на вимірюваннях, обчислено час до виснаження для кожного розглянутого ресурсу та стану (з використанням функції винагороди) та пропонується напівмарковська модель доступності, яка використовує реальні дані, а не припущення про поведінку системи.

Дослідження на основі моделей аналізують абстрактні моделі, роблячи деякі спрощені припущення про систему, такі як припущення про базові стохастичні розподіли, що характеризують систему. Ці моделі можуть застосовуватися до широкого кола систем. Однак, наприклад, омолодження ПЗ тільки на основі моделей є малоефективним (Cotroneo et al. 2014), оскільки воно не може використовувати особливості і адаптуватися до умов конкретної системи. Тобто підходи на основі моделей повинні покладатися на реальні дані для того, щоб визначити параметри моделі.

Дослідження на основі вимірювань прогнозують старіння ПЗ на основі прямих вимірювань і надають емпіричні дані про явища старіння ПЗ. Перевагою такого підходу є те, що прогнозування старіння ПЗ може адаптуватися до поточного стану системи і точно передбачити виникнення явищ старіння. Однак, такий підхід не може бути узагальненим, оскільки він залежить від реалізації конкретної системи, що досліджується. Окрім цього, підходи на основі вимірювань не призначені для оцінки довгострокових характеристик надійності, таких як доступність.

Поєднання переваг і найкращих якостей підходів на основі моделей та на основі вимірювань для отримання ефективної та корисної інформації про процес старіння та планування омолодження використовується в гібрид-

них підходах (Cotroneo et al., 2014; Valentim et al., 2016), які починають відігравати дедалі важливішу роль і заслуговують на більшу увагу.

Висновки. Старіння ПЗ є наслідком накопичення помилок внаслідок тривалого виконання системи та виснаження ресурсів системи, що призводить до погіршення продуктивності і збільшення інтенсивності відмов ПЗ. Сучасні мобільні та вбудовані пристрої вимагають високих показників якості та надійності. Також вони є особливо вразливими до проявів ефектів старіння, оскільки тривалий час працюють без перезавантаження та мають обмежену кількість ресурсів, таких як пам'ять. Отже, вивчення явища старіння в мобільних системах і розроблення методів та засобів запобігання проявам старіння є актуальними задачами і потребують подальших досліджень.

Для того, щоб виявити та боротися з ефектами старіння ПЗ, потрібно досліджувати чинники, що впливають на процес старіння. У роботі визначено загальний набір чинників, характерний для всіх систем і явища старіння загалом, проте важливим завданням є визначення специфічних чинників для конкретних систем, зокрема мобільних.

Розроблення гібридних підходів та моделей, що включають переваги аналітичних моделей та моделей на основі вимірювань, є перспективним напрямом у вивченні проблеми старіння ПЗ. Підходи на основі вимірювань показників системи забезпечують аналітичні моделі актуальними даними, що робить можливим виконувати точніші обчислення та прогнозування наслідків старіння ПЗ.

Перелік використаних джерел

Abdullah, Z. H., Yahaya, J. H., & Deraman, A. (2015). Towards anti-Ageing model for the evergreen software system. *Proceedings of 2015 International Conference on Electrical Engineering and Informatics*, (pp. 388–393). https://doi.org/10.1109/ICEEI.2015.7352532

Abdullah, Z. H., Yahaya, J. H., Mansor, Z., & Deraman, A. (2017). Software Ageing Prevention from Software Maintenance Perspective – A Review. *Journal of Telecommunication, Electronic and Computer Engineering, 9*(3–4), 93–96.

Ahamad, S. (2016). Study of software aging issues and prevention solutions. *The International Journal of Computer Science and Information Security, 14*(8), 307–313.

Alonso, J., Belanche, L., & Avresky, D. (2011). Predicting software anomalies using machine learning techniques. *Proceedings of 2011 IEEE International Symposium on Network Computing and Applications*, (pp. 163–170). https://doi.org/10.1109/NCA.2011.29

Avizienis, A., Laprie, J.-C., Randell, B., & Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing, 1*(1), 11–33. https://doi.org/10.1109/TDSC.2004.2

Basili, V., & Perricone, B. (1984). Software Errors and Complexity: An Empirical Investigation. *Communications of the ACM, 27*(1). https://doi.org/10.1145/69605.2085

Bernstein, L., & Kintala, C. M. R. (2004). Software Rejuvenation. *CrossTalk, 6*(8), 23–26.

Cotroneo, D., Fucci, F., Iannillo, A. K., Natella, R., & Pietrantuono, R. (2016). Software aging analysis of the android mobile os. *27th International Symposium on Software Reliability Engineering*, (pp. 478–489). IEEE. https://doi.org/10.1109/ISSRE.2016.25

Cotroneo, D., Natella, R., & Pietrantuono, R. (2011). Is Software Aging related to Software Metrics. *Second International Workshop on Software Aging and Rejuvenation*, (pp. 126–132) IEEE. https://doi.org/10.1109/WOSAR.2010.5722096

Cotroneo, D., Natella, R., Pietrantuono, R., & Russo, S. (2010). Software aging analysis of the linux operating system. *21st International Symposium on Software Reliability Engineering*, (pp. 86–94) IEEE. https://doi.org/10.1109/ISSRE.2010.24

Cotroneo, D., Natella, R., Pietrantuono, R., & Russo, S. (2014). A Survey of Software Aging and Rejuvenation Studies. *ACM Journal on Emerging Technologies in Computing Systems, 10*(1), article 8. https://doi.org/10.1145/2539117

Grottke, M., & Trivedi, K. S. (2005). Software faults, software aging, and software rejuvenation. *Journal of the Reliability Association of Japan, 27*(7), 425–438.

Grottke, M., Jr, R. M., & Trivedi, K. S. (2008). The fundamentals of software aging. *International Conference on Software Reliability Engineering Workshops*, (pp. 1–6). IEEE. https://doi.org/10.1109/ISSREW.2008.5355512

Grottke, M., Li, L., Vaidyanathan, K., & Trivedi, K. S. (2006). Analysis of software aging in a web server. *IEEE Transactions on Reliability, 55*(3), 411–420. https://doi.org/10.1109/TR.2006.879609

Huang, Y., Kintala, C., Kolettis, N., & Fulton, N. (1995). Software rejuvenation: analysis, module and applications. *Proceedings of Twenty-Fifth International Symposium on Fault-Tolerant Computing*, (pp. 381–390). https://doi.org/10.1109/FTCS.1995.466961

Huo, S., Zhao, D., Liu, X., Xiang, J., Zhong, Y., & Yu, H. (2018). Using machine learning for software aging detection in Android system. *Tenth International Conference on Advanced Computational Intelligence*. https://doi.org/10.1109/ICACI.2018.8377553

Li, L., Vaidyanathan, K., & Trivedi, K. (2002). An approach for estimation of software aging in a web server. *Proceedings International Symposium on Empirical Software Engineering*. https://doi.org/10.1109/ISESE.2002.1166929

Magalhaes, J., & Silva, L. (2010). Prediction of performance anomalies in web-applications based-on software aging scenarios. *Second International Workshop on Software Aging and Rejuvenation*, (pp. 131–146). IEEE. https://doi.org/10.1109/WOSAR.2010.5722095

Maji, A. K., Hao, K., Sultana, S., & Bagchi, S. (2010). Characterizing failures in mobile OSes: a case study with Android and Symbian. *International Symposium on Software Reliability Engineering*. IEEE Computer Society, (pp. 249–258). https://doi.org/10.1109/ISSRE.2010.45

Matias, R., & Freitas, P. J. (2006). An experimental study on software aging and rejuvenation in web servers. *Proceedings of 30th Annual International Computer Software and Applications Conference*, (pp. 189–196). https://doi.org/10.1109/COMPSAC.2006.25

Musa, J. D. (1993). Operational profiles in software reliability engineering. IEEE *Software, 10*(2), 14–32. https://doi.org/10.1109/52.199724

Okamura, H., & Dohi, T. (2011). A pomdp formulation of multistep failure model with software rejuvenation. *Proceedings of IEEE Third International Workshop on Software Aging and Rejuvenation*, (pp. 14–19). https://doi.org/10.1109/WoSAR.2011.11

Parnas, D. L. (1994). Software aging. *Proceedings of 16th International Conference on Software Engineering*, (pp. 279–287). https://doi.org/10.1109/ICSE.1994.296790

Pfening, A., Garg, S., Puliafito, A., Telek, M., & Trivedi, K. (1996). Optimal software rejuvenation for tolerating soft failures. *Performance Evaluation, 27*/28, 491–506. https://doi.org/10.1016/S0166-5316(96)90042-5

Polovko, A. M., & Gurov, S. V. (2008). *Fundamentals of Reliability Theory*. St. Petersburg: BHV-Petersburg, 704 p. [In Russian].

Shereshevsky, M., Crowell, J., Cukic, B., Gandikota, V., & Liu, Y. (2003). Software aging and multifractality of memory resources. *Proceedings of IEEE International Conference on Dependable Systems and Networks*, (pp. 721–730). https://doi.org/10.1109/DSN.2003.1209987

Silva, L., Alonso, J., & Torres, J. (2009). Using virtualization to improve software rejuvenation. *IEEE Transactions on Computers, 58*(11), 1525–1538. https://doi.org/10.1109/TC.2009.119

Vaidyanathan, K., & Trivedi, K. (2005). A comprehensive model for software rejuvenation. *IEEE Transactions on Dependable and Secure Computing, 2*(2). https://doi.org/10.1109/TDSC.2005.15

Vaidyanathan, K., Harper, R., Hunter, S., & Trivedi, K. (2001). Analysis and implementation of software rejuvenation in cluster systems. *Performance Evaluation Review, 29*(1), 62–71. https://doi.org/10.1145/378420.378434

Valentim, N. A., Macedo, A., & Matias, R. (2016). A Systematic Mapping Review of the First 20 Years of Software Aging and Rejuvenation Research. *International Symposium on Software Reliability Engineering Workshops* (*ISSREW*), (pp. 21–26). IEEE. https://doi.org/10.1109/ISSREW.2016.42

Wang, D., Xie, W., & Trivedi, K. (2007). Performability analysis of clustered systems with rejuvenation under varying workload. *Performance Evaluation, 64*(3), 247–265. https://doi.org/10.1016/j.peva.2006.04.002

Xianga, J., Wenga, C., Zhaoa, D., Tiana, J., Xionga, S., Lia, L., & Andrzejak, A. (2019). A New Software Rejuvenation Model for Android. *International Symposium on Software Reliability Engineering Workshops* (*ISSREW*), (pp. 42–28). IEEE. https://doi.org/10.1109/ISSREW.2018.00021

Xie, W., Hong, Y., & Trivedi, K. (2004). Software rejuvenation policies for cluster systems under varying workload. *Proceedings of 10th IEEE Pacific Rim International Symposium on Dependable Computing*. https://doi.org/10.1109/PRDC.2004.1276563

V. S. Yakovyna, B. V. Uhrynovskyi
Lviv Polytechnic National University, Lviv, Ukraine

SOFTWARE AGING IN THE CONTEXT OF ITS RELIABILITY: A SYSTEMATIC REVIEW

This paper presents the review and analysis of literary sources devoted to the study of the software aging phenomenon. The aging process is characterized as performance deterioration and increase of failure rate that has a negative impact on the software reliability. The study has found that software errors and their accumulation during program execution were the cause of the aging software. The basic concepts and characteristics related to the phenomenon of aging, such as effects, factors and aging metrics, time to resource exhaustion, time to aging-related failure and workload are determined. One of the software aging characteristics determines that it can be removed or delayed by external intervention. The technique of prevention and delay of aging is called software rejuvenation. The paper considered a common set of factors that is characteristic of all systems and the phenomenon of aging in general. The factors can be classified into two following types: external, such as software errors and code metrics, and external, such as environment, human and functional. The important task is to identify specific factors for specific systems, in particular, mobile platforms. The study reviewed and compared the main methods and approaches to study and modeling of software aging process. Aging phenomenon is studied at the theoretical level using analytical models and at the empirical level using data analysis. The paper states that the hybrid approaches could be used in researches because they incorporate the benefits of approaches based on analytical models and on measurements. Aging characteristics indicate that mobile operating systems and applications exposed to aging and there is a need to study this phenomenon in order to ensure the reliability of modern software. Mobile systems are vulnerable to manifestations of aging effects, since they work for a long time without rebooting and have a limited amount of resources, such as memory. To sum up, it is necessary to continue research of the mobile software aging, in particular to identify the aging factors of mobile applications and explore the application of methods and models for mobile systems.

Keywords: aging-related failure; aging-related error; aging factors; time to exhaustion; Markov chain; time series.