Issue 9, 2021

https://doi.org/10.23939/sisn2021.09.032

УДК 004.052; 004.416.2

МЕТРИКИ ІНТЕРФЕЙСУ КОРИСТУВАЧА ДЛЯ ВИЯВЛЕННЯ ЯВИЩА СТАРІННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В МОБІЛЬНІЙ СИСТЕМІ ANDROID

Віталій Яковина[1], Богдан Угриновський[2]

Національний університет "Львівська політехніка",
[1] ORCID: 0000-0003-0133-8591, email: vitaliy.s.yakovyna@lpnu.ua
[2] ORCID: 0000-0002-4356-192X, email: bohdan.v.uhrynovskyi@lpnu.ua

Мобільні пристрої та системи, зокрема Android, вразливі до виникнення у них ефектів старіння програмного забезпечення, які проявляються в зниженні продуктивності під час їх тривалого використання. Для виявлення ефектів старіння та протидії їм важливо ідентифікувати ефективні метрики системи та користувацького інтерфейсу. Метрики старіння, що використовуються у сучасних дослідженнях операційної системи Android, не враховують процесів старіння у користувацьких додатках. Тому в цій роботі розглянуто дві нові метрики графічного інтерфейсу користувача, які дають змогу відстежувати зниження продуктивності та збільшення часу відгуку користувацьких додатків: тривалість відображення кадрів та кількість "зіпсованих" кадрів. Реалізовано фреймворк для практичної перевірки та аналізу нових метрик, що забезпечує виконання стресового тестування мобільних додатків операційної системи Android, збирання даних про стан системи під час тестування та формування часових рядів для їх подальшого аналізу та дослідження системних метрик і метрик графічного інтерфейсу користувача. Запропоновані метрики було порівняно із раніше використовуваною метрикою тривалості запуску Android Activity і системними метриками використання пам'яті. Доведено на основі практичних результатів, що метрики тривалості відображення та "зіпсованих" кадрів забезпечують даними, застосовними у переважній кількості сценаріїв використання мобільних додатків. Тому запропоновано використати нові метрики в комбінації із іншими метриками для виявлення старіння в системі та вивчення явища старіння загалом. Зазначено, що метрика тривалості відображення кадрів дає змогу визначити стани системи та порогові значення переходів між цими станами, що забезпечує можливість побудови математичних моделей на основі ланцюгів Маркова чи обчислення часу до відмови через старіння за допомогою регресійних методів. Виявлено необхідність додаткового вивчення залежностей між метриками тривалості відображення кадрів, кількості "зіпсованих" кадрів та використання пам'яті різними процесами системи. Отже, обґрунтовано доцільність використання запропонованих метрик у майбутніх дослідженнях явища старіння користувацьких додатків у операційній системі Android.

Ключові слова: старіння програмного забезпечення; надійність програмного забезпечення; метрики старіння; мобільна система; операційна система Android.

Вступ

Процес старіння програмного забезпечення [1] характеризується накопиченням помилок у системі та користувацьких додатках під час їх тривалого використання без перезавантажень. Накопичення помилок призводить до виникнення ефектів старіння, таких як зниження продуктивності та збільшення кількості відмов, пов'язаних зі старінням [2]. Найчастіше чинниками старіння [2, 3] є помилки, які виникають у цій системі, але проявляються через певний час із затримкою, наприклад, помилки заокруглення чи витоки пам'яті. Ефекти старіння призводять до погіршення характеристик надійності [4] та продуктивності програмного забезпечення, які є критично важливими властивостями більшості сучасних пристроїв та програмних систем, зокрема тих, що використовують операційну систему Android.

Процедура омолодження програмного забезпечення [1] – це проактивний підхід для протидії ефектам старіння. Прикладом такої процедури є плановане перезавантаження системних сервісів, окремих додатків чи всього мобільного пристрою, що дає змогу зменшити кількість помилок старіння і повернути систему в так званий "молодий" стан. Час до відмови через старіння – це один із найважливіших індикаторів старіння у контексті надійності. Якщо йдеться про використання ресурсів системи, наприклад, використання пам'яті чи процесорного часу [5], то індикатором старіння є час до виснаження системних ресурсів. Індикатори старіння важливі для завдань планування та виконання процедури омолодження, оскільки виміряні метрики стану системи можуть бути використані для визначення наявності чи відсутності старіння у системі та прогнозування часу до переходу системи в стан, коли вона схильна до відмов через старіння [6]. Отже, визначення коректних та ефективних метрик старіння – актуальне науково-практичне завдання.

Постановка проблеми

У попередніх роботах [7, 8] виконано аналіз явища старіння програмного забезпечення мобільних систем з погляду його надійності та зроблено висновок, що мобільні пристрої особливо вразливі до ефектів старіння програмного забезпечення. Основними причинами цього є те, що мобільні пристрої можуть працювати тривалий час без вимкнення і перезавантаження операційної системи, а їх апаратні ресурси обмежені.

Android – це найпопулярніша мобільна операційна система на основі ядра операційної системи Linux із відкритим програмним кодом, яка в 2021 р. займає 83,8 % [9] на ринку мобільних систем. В операційній системі Android різні дослідники виявили явище старіння програмного забезпечення [6, 8, 10, 11]. Запропоновано підходи до протидії явищу старіння за допомогою як омолодження [6, 12], так й інших методів, що дають змогу уникнути виникнення помилок, які призводять до старіння [13].

Один із напрямів актуальних досліджень – визначення ефективних метрик для виявлення проявів старіння на рівні операційної системи та у користувацьких додатках. Зокрема, тривалість відображення Android Activity як метрика користувацького інтерфейсу, що використовується в останніх дослідженнях, має обмеження для реальних випадків використання. Тому пропонуємо розглянути нові метрики користувацького інтерфейсу, а саме тривалість відображення кадрів та кількість "зіпсованих" кадрів під час відображення користувацьких додатків.

Аналіз останніх досліджень та публікацій

Для дослідження явища старіння, зокрема для виявлення чинників та ефективних індикаторів старіння в операційній системі Android, різні групи дослідників [6, 10, 11] використовували методологію, яка полягає у виконанні стресових тестів для мобільних додатків, збиранні даних про використання системних ресурсів та роботу різних процесів під час тестування, перетворенні зібраних даних на часові ряди, придатні для аналізу математичними методами. В роботі [8] описано узагальнену методологію, яка складається із шести кроків:

1. Визначення стратегії моніторингу.
2. Визначення алгоритму генерації робочого навантаження.
3. Планування виконання стресових тестів.
4. Виконання стресових тестів.

5. Аналіз зібраних даних.
6. Презентація результатів експерименту.

Результати аналізу старіння програмного забезпечення за допомогою розробленої методології [11] показали, що робоче навантаження, технічні характеристики пристрою і версія системи впливають на продуктивність, а тренди щодо зниження продуктивності корелюють із використанням пам'яті процесами і роботою збирача сміття.

Доменіко Котронео і співавтори в 2016 р. [11] ідентифікували дві групи метрик: метрики, що сприймає користувач, та системні метрики.

Метрики, які сприймає користувач, також можна назвати метриками користувацького інтерфейсу, оскільки вони дають змогу оцінити відгук системи на події введення користувача та продуктивність інтерфейсу загалом.

Системні метрики – це індикатори використання таких системних ресурсів, як оперативна пам'ять та сховище даних, а також такі системні операції, як збирач сміття та менеджер процесів. Порівняння й аналіз системних метрик разом із метриками інтерфейсу дають змогу ідентифікувати такі ділянки операційної системи Android, які піддаються ефектам старіння.

У роботі [6] визначено групу метрик із високим рівнем довіри для визначення проявів старіння в операційній системі Android. Зокрема, це метрики System Server PSS та тривалість запуску Android Activity.

PSS (Proportional Set Size) – це частина оперативної пам'яті, що зайнята певним процесом і складається із приватної пам'яті цього процесу та частини спільної пам'яті одного чи декількох процесів. System Server – це процес, що працює протягом усього періоду служби системи, ініціалізує шар Application Framework і запускає майже всі системні сервіси Java. Експерименти показали, що у процесі System Server спостерігається значне збільшення PSS.

Для отримання даних про використання пам'яті використовують системний засіб dumpsys та його команду meminfo:

```
adb shell dumpsys meminfo
```

Ця утиліта повертає інформацію про використання оперативної пам'яті різними процесами. Важливими метриками використання пам'яті є Total RAM, Total Used RAM, Total Free RAM та PSS окремих процесів.

Тривалість запуску Android Activity (Launch Time) – це кількість часу, який пройшов від моменту запуску процесу до остаточного відображення відповідного Activity (екрана додатка) на дисплеї мобільного пристрою. В Android 4.4 і вище утиліта командного рядка logcat [13] виводить рядок, що містить значення, яке називається Displayed. Приклад рядка в журналі про подію завершення відображення Android екрана додатка має такий вигляд:

```
ActivityManager: Displayed com.my.app/.MainActivity: +4s248ms
```

Різні автори [6, 10, 11] використовували тривалість запуску Activity як прямий індикатор старіння програмного забезпечення, що може відчути користувач. У [11] застосунки, які зазнавали робочого навантаження, перезавантажували кожні 60 секунд для того, щоб забезпечити регулярне збирання даних про тривалість запуску Activity протягом всього часу виконання експерименту, а також щоб уникнути кешування Activity системою Android і запобігти накопиченню помилок (таких як витоки пам'яті) всередині користувацьких додатків, оскільки зосереджували увагу на старінні програмного забезпечення в середині самої операційної системи Android.

Android-застосування зазвичай мають тільки одну головну Activity, тому частота запусків Activity під час реального використання мобільного пристрою може бути низькою. Зокрема, у випадку крос-платформових фреймворків та додатків майже вся логіка та користувацький інтерфейс реалізується в межах одного Android Activity екрана. Потрібно також враховувати різні випадки використання програм. Наприклад, інколи користувач регулярно і велику кількість часу використовує тільки один додаток, не переходячи на інший. Тому використання індикатора тривалості запуску Android Activity і необхідність у перезапуску додатків для збирання інформації під час

виконання тестів накладають певні обмеження на можливість коректного оцінювання наявності старіння у системі в реальних варіантах використання мобільного девайсу та поведінки користувача. Тривалість запуску Activity не дає змоги простежити старіння в користувацьких додатках.

Формулювання цілі статті

Мета цієї роботи полягає у тому, щоб розглянути нові метрики графічного інтерфейсу користувача в контексті старіння програмного забезпечення, які були б ефективними та які можна виміряти упродовж життєвого циклу користувацьких застосувань.

Для досягнення поставленої мети виконано такі завдання:

- Розроблення фреймворку для виконання стресових тестів для мобільних додатків Android, перетворення зібраних даних про стан системи під час виконання тестів на часові ряди для відповідних наборів метрик.
- Аналіз ефективності таких метрик графічного інтерфейсу користувача мобільної системи, як тривалість відображення кадрів та кількість “зіпсованих” кадрів. Порівняння їх з метрикою тривалості запуску Android Activity.
- Аналіз кореляцій між виміряними метриками, обґрунтування доцільності використання нових метрик та необхідність їх подальших досліджень у контексті старіння мобільного програмного забезпечення.

Виклад основного матеріалу

Розроблено фреймворк, який реалізує шестиетапну методологію моніторингу і тестування, що описана вище під час аналізу останніх досліджень та публікацій та складається із трьох частин:

1) модуль виконання стресових тестів та збирання інформації про стан системи;
2) модуль перетворення системних метрик на часові ряди;
3) модуль аналізу часових рядів.

Для роботи модуля стресових тестів та збирання інформації використано персональний комп’ютер та мобільний пристрій, з’єднання між якими відбувалось через USB інтерфейс та утиліту командного рядка *adb* [14] для обміну командами та збирання даних з мобільного пристрою.

Визначення стратегії моніторингу

Етап визначення стратегії моніторингу полягає у виборі системних метрик та індикаторів, компонентів та застосувань для дослідження, а також технічних засобів для вимірювань вибраних метрик.

Для плавної роботи додатків потрібно забезпечити оптимальну кількість кадрів за секунду (60 fps) та мінімізувати кількість кадрів, які були пропущені чи затримані. Тому вимірювали нові індикатори продуктивності інтерфейсу користувача: тривалість відображення кадрів (Frame Draw Time) та кількість “зіпсованих” кадрів (Janky Frames).

Для порівняння та оцінювання нових метрик також збирали інформацію про тривалість запуску Android Activity та використання пам’яті, зокрема індикатор PSS, що описано вище під час аналізу останніх досліджень та публікацій.

Для отримання системних даних використано системний засіб Android *dumpsys* [15], що виконується на мобільному пристрої та дає змогу зібрати інформацію про системні сервіси. Агреговану статистику про відображення кадрів для кожного додатка отримували за допомогою команди *gfxinfo* та вказаної назви пакета цього додатка як параметра. Для одержання детальних даних про час і тривалість відображення кадрів використовували команду *framestats*. Приклад повної команди для збирання інформації про кадри додатка інтернет-браузера:

```
adb shell dumpsys gfxinfo com.android.chrome framestats
```

Ця команда повертає багато даних, серед яких необхідно виділити чотири важливих записи, необхідні для цього дослідження: Total Frames Rendered, Janky Frames, INTENDED_VSYNC та FRAME_COMPLETED.

Тривалість відображення кадру (Frame Time) – це різниця між FRAME_COMPLETED і INTENDED_VSYNC. Необхідно, щоб ця тривалість відображення кадру не перевищувала 16 мс, щоб забезпечити частоту оновлення кадрів понад 60 кадрів за секунду.

Кількість "зіпсованих" кадрів (Janky Frames) – це пропущені чи затримані кадри, які не були відображені. Відношення Janky Frames до Total Frames Rendered вказує на частку пропущених чи затриманих кадрів.

Сформовано набір із п'яти додатків для виконання стресового тестування та збирання інформації на основі припущення, що цей набір часто використовує середньостатистичний користувач мобільного пристрою: com.google.android.youtube, com.android.contacts, com.android.chrome, com.android.camera, com.google.android.apps.photos.

Визначення алгоритму генерації робочого навантаження

Реалізація алгоритму генерації робочого навантаження полягає у моделюванні використання мобільного пристрою користувачем.

Використано застосунок командного рядка *monkey* [16], який виконується на емуляторі чи мобільному пристрої та генерує псевдовипадковий потік таких подій, як натискання на кнопки, доторкання до дисплею, жести, а також різні події рівня системи. Приклад команди *monkey*:

```
adb shell monkey [options] <event-count>
```

Щоб припинити виконання додатків, кожні 30 секунд використовували засіб командного рядка Android *am* (Activity Manager) та його команду *force-stop*.

Планування виконання стресових тестів

Загальний план і опис виконаних тестів подано в табл. 1. У цій роботі виконано вісім стресових тестів. Для порівняння і аналізу ефективності метрик тривалості запуску Android Activity та відображення кадрів виконано дві групи тестів по чотири тестові запуски в кожній. Чотири тести виконано з примусовим перезавантаженням усіх додатків кожні 30 секунд, а інші чотири тести без примусового перезавантаження. Робоче навантаження генерувалось протягом всього часу виконання тесту.

Таблиця 1

Загальний план виконання стресових стестів

		Тести з примусовими перезапусками додатків	Тести без примусових перезапусків додатків
Кількість тестів		4	4
Тривалість виконання тесту		2,5–3 години	
Інтервал запису метрик		20–30 секунд	
Користувацькі додатки		com.google.android.youtube com.android.contacts com.android.chrome com.android.camera com.google.android.apps.photos	
Вимірювані метрики	користувацького інтерфейсу	Тривалість відображення Android Activity Тривалість відображення кадрів Кількість "зіпсованих" кадрів	
	системні	PSS Total RAM, Used RAM, Free RAM	
Версія Android		7.0	
Модель мобільного пристрою		Xiaomi Redmi Note 4, 3 Gb RAM, 2018	

Аналіз результатів

Для аналізу зібраних під час стресового тестування даних виконано їх перетворення на часові ряди. Отримані експериментальні дані із різних джерел перетворено на таблиці csv формату, а потім об'єднано в один часовий ряд з 30-секундним інтервалом. Метрики тривалості запуску Android Activity, тривалості відображення кадрів та PSS – це середні арифметичні значення всіх записів у 30-секундному тестовому інтервалі. Кількість "зіпсованих" кадрів у часовому ряді – це сума всіх значень 30-секундного інтервалу часового ряду, що відображає загальну кількість для всіх процесів у цьому часовому інтервалі.

Застосовуючи метод лінійної регресії для метрик тривалості запуску Android Activity, тривалості відображення кадрів та кількості "зіпсованих" кадрів, можна робити висновки про продуктивність системи та виявляти тренди старіння програмного забезпечення. Метрику тривалості запуску Android Activity перевірено і використано в попередніх роботах, тому нові метрики порівнювали із нею. Припустимо, що тривалість відображення кадрів чи кількість "зіпсованих" кадрів можна розглядати як метрики старіння, якщо їх знак коефіцієнта лінійної регресії збігається зі знаком коефіцієнта лінійної регресії тривалості запуску Android Activity. Але також можливе припущення про те, що метрики кадрів можуть бути репрезентативнішими і такими, що показують тренд старіння у тих випадках, коли метрику тривалості відображення Android Activity неможливо використати.

Для визначення кореляцій між окремими індикаторами, зокрема для визначення впливу використання пам'яті на метрики кадрів, застосовано коефіцієнт кореляції Спірмена.

Ефективність метрик відображення кадрів

Примусовий перезапуск додатків (рис. 1) дає змогу робити регулярні записи про тривалість відображення Android Activity так само, як і про тривалість відображення кадрів. Однак такий підхід до виконання експерименту унеможливлює перевірку реальніших випадків використання користувацьких додатків, наприклад, коли користувач постійно працює із одним додатком, чи цей додаток реалізовано за допомогою тільки однієї Android Activity. Результати експерименту (рис. 2) показують, що метрика тривалості відображення кадру може надати деталізованішу картину процесу старіння без необхідності примусового перезапуску додатків. Метрика тривалості запуску Android Activity в тестах без перезапусків містить приблизно в чотири рази менше записів, ніж метрика тривалості відображення кадру.

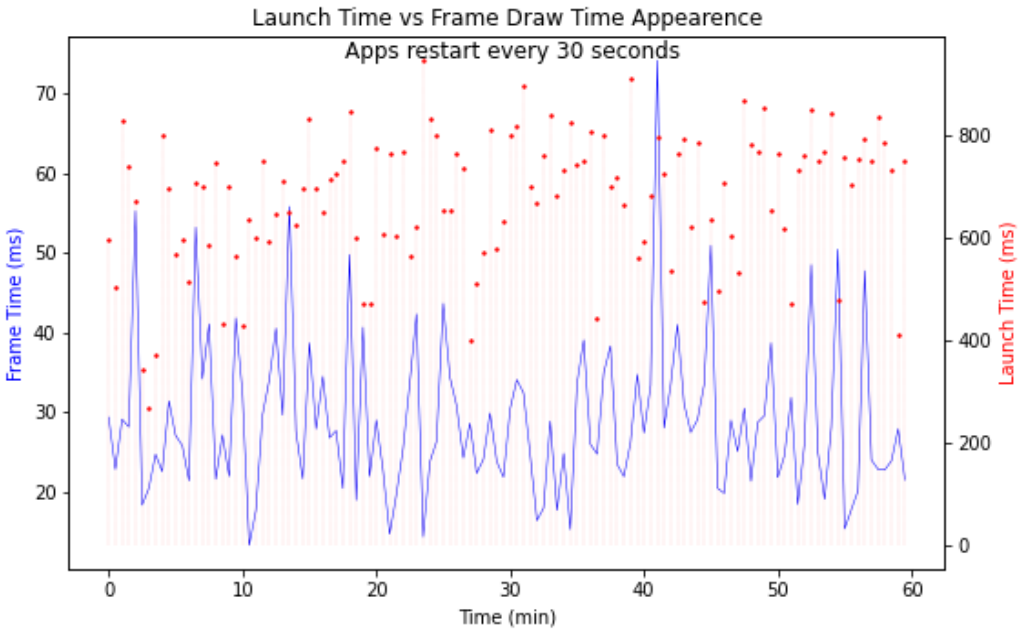

Рис. 1. Порівняння метрики тривалості запуску Android Activity та тривалості відображення кадрів, коли додатки перезапускаються кожні 30 с

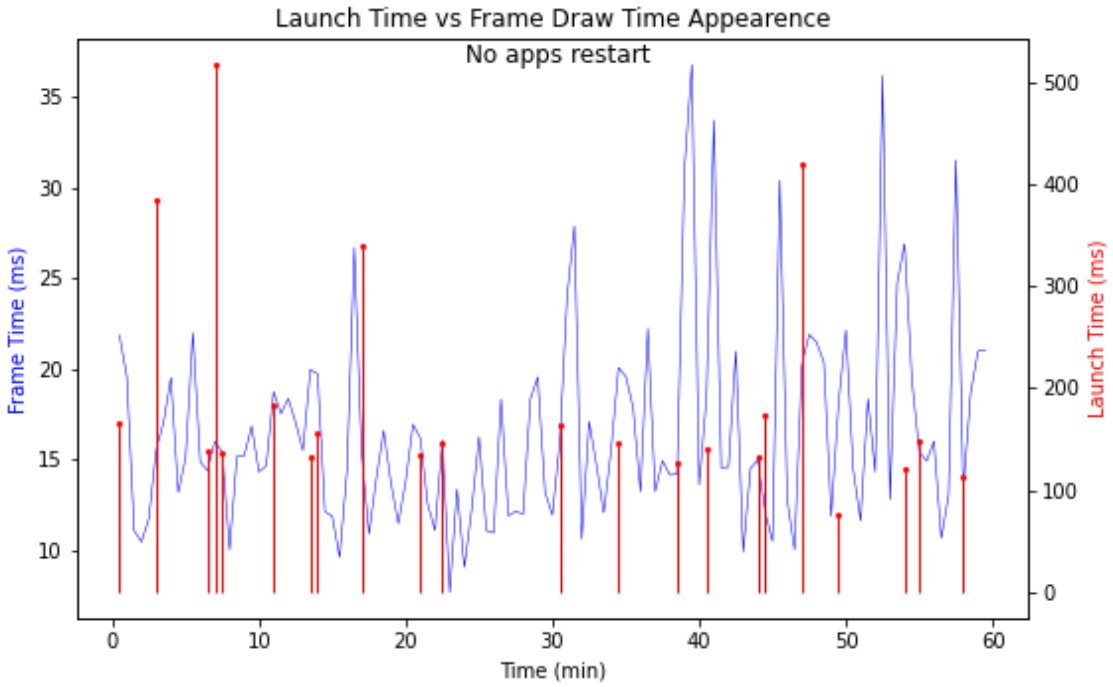

Рис. 2. Порівняння записів метрик тривалості запуску Android Activity та тривалості відображення кадрів без перезапуску додатків

Для виявлення трендів старіння застосовано метод лінійної регресії, який використовує часові ряди відповідних метрик. На рис. 3 подано лінійну регресію тривалості відображення кадрів для другого експерименту із перезапуском додатків. Результати експерименту показують, що тривалість відображення кадрів вже перевищує порогове значення 16 мс, яке б забезпечувало 60 кадрів за секунду. Враховуючи наявність тренду до збільшення тривалості відображення кадру, можна було б застосувати методи прогнозування для визначення часу, коли тривалість відображення кадрів буде критично високою і зможе спричинити відмову старіння. На рис. 4 наведено лінійну регресію для метрики “зіпсованих” кадрів для першого експерименту без перезапуску додатків. Цю метрику можна застосувати як для визначення наявності старіння, так і для детальнішого вивчення старіння, розглядаючи причини виникнення пропущених чи затриманих кадрів.

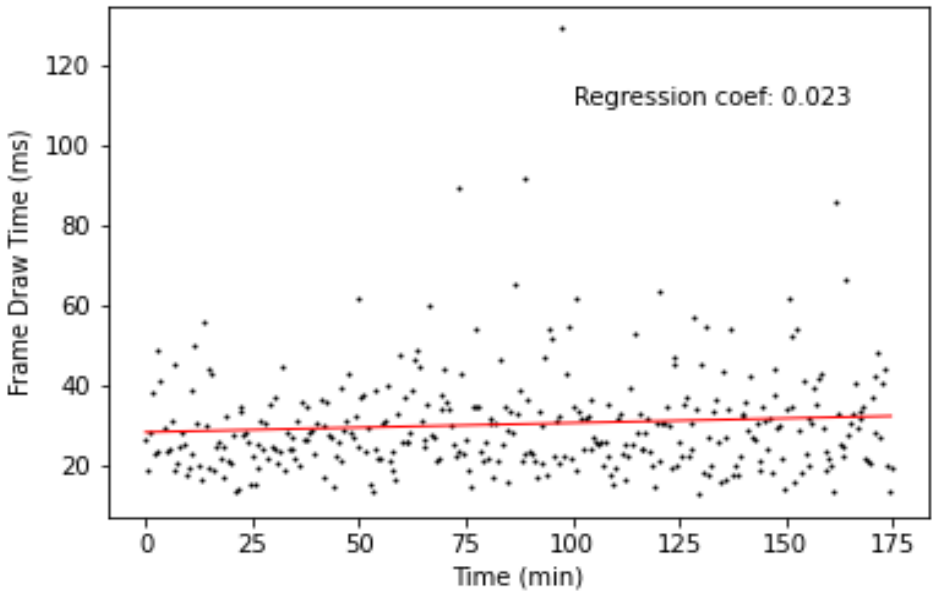

Рис. 3. Приклад лінійної регресії для метрики тривалості відображення кадрів

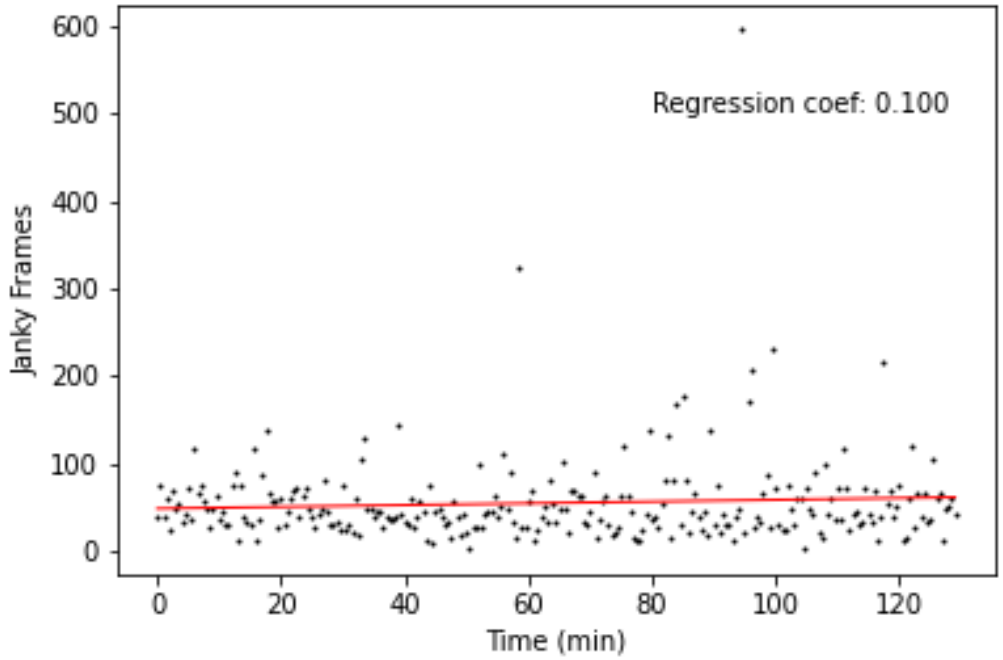

Рис. 4. Приклад лінійної регресії для метрики "зіпсованих" кадрів

У табл. 2 подано коефіцієнти лінійної регресії для кожної з виміряних метрик у кожному експерименті. Знак "+" вказує, що спостерігається старіння в експериментальних даних, а знак "–" – що старіння не спостерігається. У п'яти із восьми тестів тренди метрик тривалості запуску Android Activity та тривалості відображення кадрів збігаються. В чотирьох із восьми тестів тренди метрик тривалості запуску Android Activity та кількості "зіпсованих" кадрів однакові. В чотирьох із восьми тестів тренди метрик тривалості відображення кадрів та "зіпсованих" кадрів збігаються. В усіх випадках сумарний PSS та середній PSS (рис. 5) відображають стабільний тренд старіння.

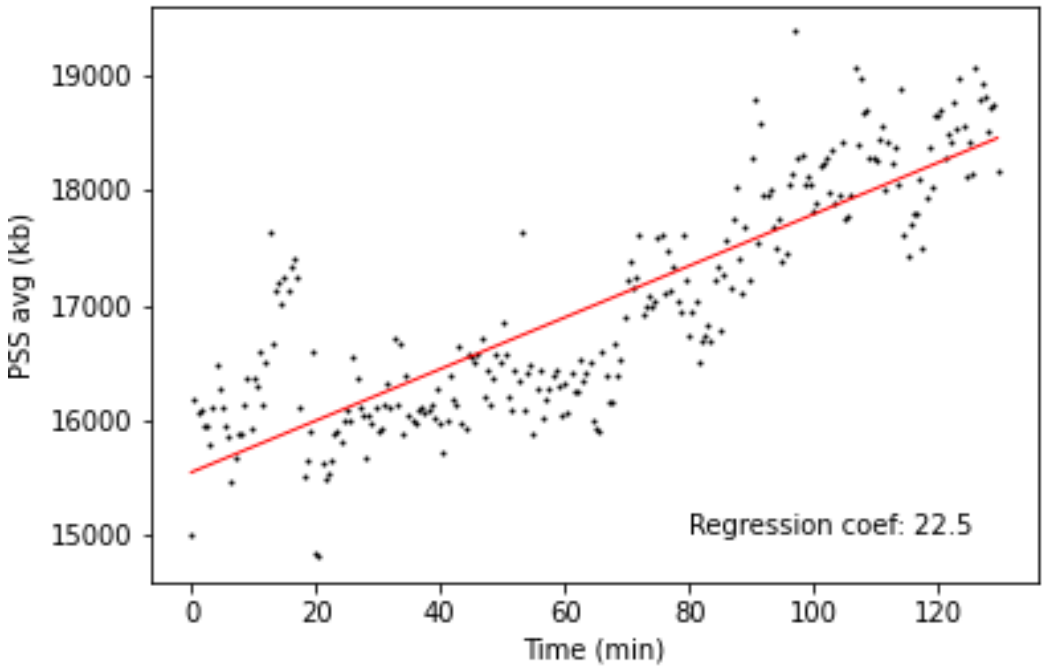

Рис. 5. Приклад лінійної регресії для метрики середнього PSS

Таблиця 2

Коефіцієнти лінійної регресії для виміряних метрик

Метрики	Тести з примусовими перезапусками додатків				Тести без примусових перезапусків додатків			
	1	2	3	4	1	2	3	4
Тривалість запуску Android Activity	+0,03	+0,038	***–0,275***	***–0,049***	***–0,022***	+0,106	***–0,664***	+0,565
Тривалість відображення кадрів	***–0,02***	+0,023	***–0,026***	+0,009	+0,009	+0,004	***–0,01***	+0,015
Кількість “зіпсованих” кадрів	+0,1	***–0,039***	***–0,005***	+0,005	+0,1	+0,018	***–0,065***	***–0,008***
Середній PSS	+4,2	+6,2	+8,2	+5,5	+22,5	+17,8	+17,3	+22,5
Сумарний PSS	+640,3	+531,8	+853,6	+576,9	+3410,6	+1938,5	+2889,4	+1493,0

Ефективні метрики користувацького інтерфейсу повинні забезпечувати достатньою кількістю даних для різних варіантів використання. Практичне значення має варіант тривалого використання одноекранного додатка, коли неможливо отримати достатньо інформації про тривалість відображення Android Activity. Наприклад, Android планшет можуть часто використовувати працівники підприємств, які тривалий час працюють із певним спеціалізованим додатком. У такому разі використання метрик кадрів для визначення наявності старіння може бути доречним, тому потрібні подальші дослідження. Розглянуті метрики також можуть застосовуватися в комбінації, щоб забезпечити можливість виявлення старіння програмного забезпечення в різних варіантах використання користувачем мобільного пристрою та додатків.

Також варто зауважити, що метрику тривалості відображення кадру можна застосувати для визначення стану старіння системи та для підрахунку часу до відмови через старіння [6]. Відмовою через старіння у цьому випадку можна вважати той стан системи, коли тривалість відображення кадру істотно перевищує 16 мс, наприклад, понад 48 мс. Отже, можна виділити три стани системи в контексті старіння: система в молодому стані (до 16 мс); система в стані переходу до старіння (16–48 мс); система в стані відмови через старіння (понад 48 мс). Своєю чергою, такий поділ на стани та визначення порогових значень може бути застосований для побудови математичних моделей на основі ланцюгів Маркова чи для обчислення часу до відмови за допомогою регресійних методів, зокрема методу лінійної регресії.

Кореляції між метриками

Для розглянутих метрик не виявлено сильних кореляцій (табл. 3). Тому зосереджено увагу на кореляціях між метриками кадрів та метриками сумарного та середнього PSS. Існує незначна негативна кореляція (–0,3±0,05) між “зіпсованими” кадрами та сумарним PSS. Можна припустити: це спричинено тим, що збільшення обсягу використовуваної пам’яті процесами зменшує кількість помилок кадрів через недостатність пам’яті. Потрібно продовжити дослідження окремих користувацьких додатків та системних процесів для визначення кореляцій між метриками кадрів та використання пам’яті.

Таблиця 3

Коефіцієнти кореляції Спірмена між виміряними метриками та трендами PSS

Вимірювані метрики	Тести з примусовими перезапусками додатків				Тести без примусових перезапусків додатків			
	1	*2*	*3*	*4*	*1*	*2*	*3*	*4*
Тривалість відображення Android Activity (Total/Avg PSS)	–0,03; 0,06;	–0,001; 0,04;	–0,06; –0,06;	–0,03; –0,01;	**0,1;** –0,05;	0,09; 0,07;	–0,05; **–0,11;**	–0,05; –0,07;
Тривалість відображення кадрів (Total/Avg PSS)	–0,03; **–0,11;**	**0,18;** 0,04;	0,01; **–0,16**	–0,04; –0,09;	0,09; 0,04;	0,03; 0,08;	–0,01; –0,08;	**0,11;** –0,05;
Кількість “зіпсованих” кадрів (Total/Avg PSS)	**–0,23;** 0,01;	**–0,3;** –0,03;	**–0,33;** 0,03;	**–0,2;** 0,05;	**–0,13;** –0,04;	**–0,26;** 0,09;	**–0,14;** **–0,11;**	**–0,27;** 0,05

Висновки

У цій роботі реалізовано фреймворк для виконання стресового тестування мобільних додатків для операційної системи Android для того, щоб ідентифікувати та вивчати метрики і чинники старіння.

Запропоновано та експериментально перевірено дві метрики графічного інтерфейсу користувача: тривалість відображення кадрів та кількість “зіпсованих” кадрів. Перевага обох метрик у тому, що вони забезпечують можливість будувати неперервні часові ряди, що можуть бути корисними в більшості тестових та реальних сценаріїв, зокрема для визначення старіння у користувацьких додатках. Обидві метрики можна використовувати в комбінації з метрикою тривалості відображення Android Activity та іншими системними метриками для прогнозування старіння за допомогою регресійних моделей чи моделей на основі ланцюгів Маркова.

У майбутніх роботах заплановано перевірити інші чинники, такі як нативні та крос-платформові додатки, одноекранні та багатоекранні додатки, довготермінові стрес-тести одного додатка чи набору додатків. Також необхідно продовжити дослідження кореляцій у контексті процесу старіння між “зіпсованими” кадрами, тривалістю відображення кадрів, використанням пам’яті процесами та іншими системними метриками для того, щоб визначити причини та умови виникнення старіння.

Список літератури

1. Dohi, T., & Trivedi, K., & Avritzer, A. (2020). Handbook of software aging and rejuvenation. World Scientific Publishing Co Pte Ltd, 424. Retrieved from https://doi.org/10.1142/11673

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

3. 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.

4. Polovko, A. M., & Gurov, S. V. (2008). Fundamentals of Reliability Theory. St. Petersburg: BHV-Petersburg, 704.

5. Bao, Y., & Sun, X., & Trivedi, K. S. (2005). A workload-based analysis of software aging and rejuvenation. *IEEE Transactions on Reliability*, 54 (3), 541–548. Retrieved from https://doi.org/10.1109/TR.2005.853442

6. Cotroneo, D., & Simone, L. D., & Natella, R., & Pietrantuono, R., & Russo, S. (2019). A Configurable Software Aging Detection and Rejuvenation Agent for Android. *11th Intl Workshop on Software Aging and Rejuvenation (WoSAR).* Retrieved from https://doi.org/10.1109/ISSREW.2019.00078

7. Яковина, В. С., & Угриновський, Б. В. (2019). Старіння програмного забезпечення в контексті його надійності: огляд проблематики. *Науковий вісник НЛТУ України*, *29*(5), 123–128. Retrieved from https://doi.org/10.15421/40290525

8. Яковина, В. С., & Угриновський, Б. В. (2020). Старіння програмного забезпечення мобільних додатків: аналіз проблематики. *Науковий вісник НЛТУ України*, *30*(2), 107–112. Retrieved from https://doi.org/10.36930/40300219

9. International Data Corporation. (n. d.). Smartphone OS market share. Retrieved from http://www.idc.com/promo/smartphone-market-share/os

10. Araujo, J., & Alves, V., & Oliveira, D., & Dias, P., & Silva, B., Maciel, P., (2013). An Investigative Approach to Software Aging in Android Applications. *IEEE International Conference on Systems, Man, and Cybernetics.* Retrieved from https://doi.org/10.1109/SMC.2013.213

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

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

13. Wu, H., & Wolter, K. (2015). Software aging in mobile devices: Partial computation offloading as a solution. *IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW),* 125–131. Retrieved from https://doi.org/10.1109/ISSREW.2015.7392057

14. Android Developers. (n. d.). Logcat command-line tool. Retrieved from https://developer.android.com/studio/command-line/logcat.

15. Android Developers. (n. d.). Android Debug Bridge. Retrieved from https://developer.android.com/studio/command-line/adb

16. Android Developers. (n. d.). Dumpsys – Android Developers. Retrieved from https://developer.android.com/studio/command-line/dumpsys

17. Android Developers. (n. d.). UI/Application Exerciser Monkey. Retrieved from https://developer.android.com/studio/test/monkey

References

1. Dohi, T., & Trivedi, K., & Avritzer, A. (2020). Handbook of software aging and rejuvenation. World Scientific Publishing Co Pte Ltd, 424. https://doi.org/10.1142/11673

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

3. 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.

4. Polovko, A.M., & Gurov, S.V. (2008). Fundamentals of Reliability Theory. St.Petersburg: BHV-Petersburg, 704.

5. Bao, Y., & Sun, X., & Trivedi, K. S. (2005). A workload-based analysis of software aging and rejuvenation. IEEE Transactions on Reliability. 54 (3), 541–548. https://doi.org/10.1109/TR.2005.853442

6. Cotroneo, D., & Simone, L. D., & Natella, R., & Pietrantuono, R., & Russo, S. (2019). A Configurable Software Aging Detection and Rejuvenation Agent for Android. 11th Intl Workshop on Software Aging and Rejuvenation (WoSAR). https://doi.org/10.1109/ISSREW.2019.00078

7. 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

8. Yakovyna, V. S., & Uhrynovskyi, B. V. (2020). Software aging in the context of reliability: a review of the issue. Scientific Bulletin of UNFU, 30(2), 107-112. https://doi.org/10.36930/40300219

9. International Data Corporation. (n. d.). Smartphone OS market share. http://www.idc.com/promo/smartphone-market-share/os

10. Araujo, J., & Alves, V., & Oliveira, D., & Dias, P., & Silva, B., Maciel, P., (2013). An Investigative Approach to Software Aging in Android Applications. IEEE International Conference on Systems, Man, and Cybernetics. https://doi.org/10.1109/SMC.2013.213

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

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

13. Wu, H., & Wolter, K. (2015). Software aging in mobile devices: Partial computation offloading as a solution. IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 125–131. https://doi.org/10.1109/ISSREW.2015.7392057

14. Android Developers. (n. d.). Logcat command-line tool. https://developer.android.com/studio/command-line/logcat.

15. Android Developers. (n. d.). Android Debug Bridge. https://developer.android.com/studio/command-line/adb

16. Android Developers. (n. d.). Dumpsys - Android Developers. https://developer.android.com/studio/command-line/dumpsys

17. Android Developers. (n. d.). UI/Application Exerciser Monkey. https://developer.android.com/studio/test/monkey

USER-PERCEIVED RESPONSE METRICS IN ANDROID OS FOR SOFTWARE AGING DETECTION

Vitaliy Yakovyna[1], Bohdan Uhrynovskyi[2]

Lviv Polytechnic National University,
[1] ORCID: 0000-0003-0133-8591, email: vitaliy.s.yakovyna@lpnu.ua
[2] ORCID: 0000-0002-4356-192X, email: bohdan.v.uhrynovskyi@lpnu.ua

Mobile systems and devices including Android are vulnerable to the effects of software aging which are manifested in performance degradation during long run-time. It is important to identify efficient system and user interface metrics for detecting and counteracting the software aging effects. The aging metrics used in researches of the Android operating system do not take into account the aging processes in user applications. Therefore, this paper discusses two new graphical user interface metrics that allow to track performance degradation and user applications response time: Frame Draw Time and Janky Frames (dropped or delayed frames). Test framework was implemented to perform stress testing of mobile applications in the Android operating system, to collect system state data during stress test performing and to map obtained raw data into time series. Calculated time series are used for further analysis and study of system and graphical user interface metrics. The considered metrics have been compared to the previously used Android Activity Launch Time metric and RAM usage metrics. Practical results have shown that Frame Draw Time and Janks Frames metrics provide data, which can be useful in most scenarios of mobile application using. Therefore, it is proposed to use the two new metrics in combination with other previously used metrics to detect aging trends in the system state and to study the phenomenon of software aging in general. It is noted that the Frame Draw Time metric value can be mapped to states with determined thresholds for transition between these states. These states and thresholds provide the possibility of developing mathematical models based on Markov chains or forecasting the time to aging-failure using regression methods. The need of further study of the correlations between Frame Draw Time metric, Janky Frames metric and metrics of memory usage by different system processes has been identified. Thus, the expediency of using the proposed metrics in future studies of the aging phenomenon in the Android operating system is substantiated, in particular, the effectiveness of the proposed metrics could be checked for different mobile use cases and for different types of mobile applications.

Key words: software aging; software reliability; aging metrics; mobile system; Android OS.