УДК 004.4:004.045
DOI: 10.31471/1993-9965-2024-1(56)-92-100

КОМПОНЕНТНИЙ ПІДХІД ДО ПЕРЕДАВАННЯ СИГНАЛІВ ТА ОБРОБЛЕННЯ РЕЗУЛЬТАТІВ ПРИ УЛЬТРАЗВУКОВОМУ КОНТРОЛІ

І. З. Лютак*, З. П. Лютак

ІФНТУНГ; 76019, м. Івано-Франківськ, вул. Карпатська, 15,
e-mail: ihorlt@gmail.com

В роботі розглядається виклик, пов'язаний з різноманітністю пристроїв для ультразвукового контролю та необхідністю уніфікації отриманих даних для їх аналізу в єдиному форматі, що спростить процес контролю та дозволить розробляти нові методи аналізу. Автори підкреслюють значення відслідковуваності до оригінального джерела даних для глибшого розуміння отриманої інформації та її достовірності. Основна ціль дослідження полягає у розробці програмного забезпечення та алгоритму для ефективної обробки та аналізу даних, отриманих з ультразвукового контролю. Представлений огляд різноманітних досліджень та розробок у галузі ультразвукового контролю та використання машинного навчання для аналізу даних. Описано декілька підходів і технологій, зокрема, використання моделі на основі машинного навчання для прогнозування міцності з'єднань у 3D-друкованому бетоні, а також методики оцінки якості покриттів, нанесених адитивними методами. Автори глибоко занурюються у теоретичні основи свого підходу до аналізу даних, використовуючи концепції кінцевих автоматів стану (КАС) та конструктивної взаємодії моделі (КВМ). Кінцеві автомати стану описуються як кортежі, що включають множину станів, алфавіт, функції переходів, початковий стан, та множину приймаючих станів, дозволяючи моделювати поведінку систем через послідовності переходів між станами. Детально описуються поняття служб та їх фрагментів у контексті програмного забезпечення, де кожен компонент виконує певну функцію або обробляє повідомлення. Важливим аспектом є моделювання трасування виконання системи за допомогою часових послідовних діаграм сполучень, що дозволяє деталізувати взаємодію між компонентами. Описується метод створення автоматизованих дерев префіксів (АДП) для кожного фрагмента служби, що уможливлює детальне моделювання поведінки системи на основі виконаного коду програми. Значна увага приділяється аналізу часових анотацій у станових машинах, з особливим фокусом на визначенні та інтерпретації змін часової поведінки. Через специфіку роботи з часовими даними, дослідження висвітлює виклики, пов'язані з шумом, ненормальністю розподілів та малими розмірами вибірки, що є типовими для аналізу продуктивності реальних систем. Детально описуються інтеграція та функціональність користувацького інтерфейсу Mids+Time, який об'єднує часово-анотовані автомати різниці, евристики виявлення змін, та візуалізації в єдиному інтерфейсі. Користувацький інтерфейс Mids+Time забезпечує інтерактивні візуалізації, включаючи гістограми, контрольні карти, та графіки зміщення часових даних, які дозволяють користувачам ефективно аналізувати та порівнювати продуктивність різних компонентів програмного забезпечення.

Ключові слова: програмне забезпечення, компонентний підхід, ультразвук, моделювання.

The paper addresses the challenge associated with the diversity of devices for ultrasonic testing and the necessity to unify the obtained data for analysis in a single format, which simplifies the control process and allows for the development of new analysis methods. The authors emphasize the importance of traceability to the original data source for a deeper understanding of the information received and its reliability. The primary goal of the research is described as the development of software and algorithms for the effective processing and analysis of data obtained from ultrasonic testing. The paper provides a review of various studies and developments in the field of ultrasonic testing and the use of machine learning for data analysis. Several approaches and technologies are described, including the use of a machine learning model to predict the strength of joints in 3D-printed concrete and methods for assessing the quality of coatings applied by additive methods. The authors delve deeply into the theor-?tical foundations of their approach to data analysis, using concepts of finite state automata (FSA) and constructive interaction models (CIM). Finite state automata are described as tuples that include a set of states, an alphabet, transition functions, an initial state, and a set of accepting states, allowing the modeling of system behavior through sequences of transitions between states. The concepts of services and their fragments are detailed in the context of software, where each component performs a specific function or processes messages. An important aspect is the modeling of system execution tracing using timed sequence diagrams, which allows for detailed interaction between components. The method of creating automated prefix trees (APT) for each service fragment is described, enabling detailed modeling of system behavior based on the executed program code. Significant attention is given to the analysis of timing annotations in state machines, with a particular focus on defining and interpreting changes in temporal behavior. Due to the specifics of working with time data, the research highlights challenges

related to noise, non-normality of distributions, and small sample sizes, which are typical for analyzing the performance of real systems. The integration and functionality of the Mids+Time user interface are described in detail, combining timed automata, change detection heuristics, and visualizations in a single interface. The Mids+Time user interface provides interactive visualizations, including histograms, control charts, and time-shift graphs, which allow users to effectively analyze and compare the performance of different software components.

Keywords: software, component-based approach, ultrasound, modeling.

Вступ

Компонентний підхід до передачі результатів ультразвукового контролю відіграє ключову роль у забезпеченні точності та уніфікації даних, що є критичними для підтримання безпеки та довговічності інженерних споруд. Сучасні технології та інженерні методи дозволяють швидко та точно аналізувати стан металевих конструкцій, що є важливим для планування та виконання ремонтних робіт. Вдосконалення методів передачі та обробки даних за допомогою компонентного підходу набуває особливої актуальності в сучасному будівництві та інженерії. У контексті передачі результатів ультразвукового контролю виникає потреба в уніфікації даних та забезпеченні їхньої точності. Програмне забезпечення, здатне об'єднувати та аналізувати дані в єдиному форматі, спрощує процес контролю та дозволяє розробляти більш точні методи аналізу технічного стану металоконструкцій. Під час аналізу даних важливо враховувати особливості кожного приладу, алгоритми та методики обробки, що впливають на кінцевий результат. Відслідковуваність до оригінального джерела даних сприяє глибшому розумінню отриманої інформації та збільшує її достовірність. Специфіка роботи з часовими даними в ультразвуковому контролі металоконструкцій пов'язана з викликами, які включають роботу з шумом, ненормальністю розподілів та малими розмірами вибірки. Шум може суттєво впливати на точність отриманих результатів, що вимагає використання ефективних методів фільтрації та очищення даних. Ненормальність розподілів та малі розміри вибірок додають складності в статистичному аналізі даних, що потребує застосування спеціалізованих алгоритмів для коректного інтерпретування результатів. Ці фактори ускладнюють процес аналізу продуктивності реальних систем ультразвукового контролю, вимагаючи застосування передових методик та технологій для забезпечення точності та надійності отриманих даних. Компонентно-орієнтована інженерія програмного забезпечення є добре вивченою методикою для управління складністю великомасштабних програмних систем, яка використовується в широкому спектрі доменів. Основна ідея цього підходу полягає у розділенні системи на повторно використовувані компоненти, які спілкуються через слабко зв'язані інтерфейси. Такий підхід дозволяє кожному компоненту бути розробленим, протестованим та підтриманим автономно, що забезпечує модульність, продуктивність та здатність до взаємодії. Використання цього підходу у сфері ультразвукового контролю металоконструкцій дозволяє створювати гнучкі та ефективні системи, здатні забезпечити високу точність і достовірність переданих даних.

Мета роботи

Мета цієї роботи полягає в розробці програмного забезпечення та алгоритму, що здатні ефективно передавати, обробляти та аналізувати дані, отримані в результаті ультразвукового контролю металевих конструкцій. У сучасному світі, де безпека та надійність металоконструкцій є пріоритетними, необхідно мати засоби для швидкого та точного аналізу їх стану. Наша ціль – надати фахівцям інструментарій, який дозволить не тільки виявляти потенційні проблеми на ранніх стадіях, але й прогнозувати майбутній стан конструкцій, опираючись на якісний аналіз даних ультразвукового контролю.

Аналіз сучасних закордонних і вітчизняних досліджень і публікацій

Дослідження [1] показує підхід до аналізу результатів контролю в у процесі прискореного будівництва мостів, де застосовуються 3D-друкований бетон та структурна модернізація, щоб зменшити час будівництва на місці. У праці [1] запропоновано модель на основі машинного навчання для прогнозування міцності з'єднання між шарами на основі ультразвукових характеристик. Ультразвукові тести є вхідними параметрами, а результати випробувань на зсув – цілями для моделі машинного навчання. На ультразвукові сигнали накладалась варіаційна розкладка режиму, і з кожного сигналу видобувались три внутрішні функції режиму, що дало 21 незалежну характеристику. Науковцями було запропоновано модель XGBoost [1], яка отримала значення R-квадрату 95,5% з чотирьох характеристик як вхідних даних. Розроблену модель використовують також у прогнозуванні міцності з'єднання на основі ультразву-

кового тестування. В праці [2] дослідження стосуються неруйнівної оцінки покриттів, нанесених адитивним методом за допомогою ультразвукових вимірювань поверхневих хвиль для оцінки якості покриттів, які були нанесені на підкладки за допомогою процесу адитивного виробництва зі спрямованим енергетичним депонуванням. Було виявлено, що швидкість поверхневої хвилі та амплітуда розсіяного зворотного сигналу ультразвукових хвиль корелювали із тріщинами та вигином покриття, тому було використано три різні частоти для збудження поверхневих хвиль, оскільки кожна частота має різну глибину проникнення. Отримані сигнали були використані для аналізу металоконструкцій шляхом порівняння швидкості поверхневої хвилі та амплітуди розсіяного ультразвукового зворотного сигналу. Виявлено, що амплітуда розсіяного сигналу поверхневих хвиль корелювала із щільністю тріщин та вигином. За останні роки значно розширилось використання машинного навчання (МН) для автоматизованого аналізу даних неруйнівного випробування. Особливий інтерес представляє використання МН для аналізу даних щодо внутрішньої перевірки зварних з'єднань. Так, в праці [3] проведено оцінку поточних можливостей МН та автоматизованого аналізу даних для ультразвукового контролю, а також виявлення будь-яких прогалин або недоліків у сучасних технологіях МН, що застосовуються для автоматизованого аналізу даних результатів контролю. Дослідники [3] виявили можливості, обмеження та потенційні прогалини у виборі як характеристик, так і даних для оптимізації моделей МН. В праці [4] досліджувались складні компонентно-орієнтованої системи та часовий розподіл дій в компонентах, які можуть поширювати зміни в комунікації і сигналах через кілька компонентів, що часто призводить до негативних наслідків не там, де вони виникли, а в зовсім іншому місці. Оскільки на даний час основний акцент при розробці інтерференції моделей був зроблений на функціональних вимогах до компонентно-орієнтованої програмної системи, нефункціональні аспекти, такі як часові параметри, отримували менше уваги. Ці параметри є основою для аналізу результатів ультразвукового контролю. Науковці розробили підхід щодо включення інформації про часові параметри, доступну в слідах виконання, до моделей системної поведінки. При порівнянні отриманих даних було виявлено значущі зміни в часовій поведінці, що дало змогу запобігати регресії продуктивності.

Висвітлення невирішених раніше частин загальної проблеми

У сфері ультразвукового контролю металевих конструкцій з'являється потреба у кількісній оцінці та валідації ефективності методів МН, що включає необхідність створення спільних наборів даних для проведення об'єктивних тестів і порівнянь. Також не завжди чітко вказано, яким чином необхідно налаштовувати моделі МН для оптимізації їх продуктивності в рамках конкретних задач аналізу даних. Це може включати вибір структури моделі, налаштування. Важливим аспектом є також питання демонстрації результатів від алгоритмів МН. На даний момент не вистачає методик, які б дозволяли забезпечити відтворюваність результатів, проведення адекватної верифікації та валідації. Особливо важливим є питання компонентної архітектури побудови аналізу. Враховуючи особливості компонентно-орієнтованої інженерії програмного забезпечення, виникає необхідність у розробленні модульних систем аналізу даних, які б могли ефективно взаємодіяти між собою, гарантуючи при цьому надійність і точність результатів. Існує ряд завдань, на які існуючі дослідження лише частково надають відповіді, включаючи оптимальний вибір характеристик, роботу із незбалансованими даними для ефективного використання та інтерпретації методів МН в рамках ультразвукового контролю.

Формулювання цілей статті

В роботі ми ставимо ціль щодо побудови моделі аналізу даних на основі компонентного підходу. Ця модель дозволить інтегрувати різноманітні сигнали з ультразвукових пристроїв, які використовують сучасні методи ультразвукового контролю. Одними з ключових компонентів моделі будуть сигнали з різних ультразвукових пристроїв. Це дозволить отримати більш деталізовану інформацію про стан металевих конструкцій, комбінуючи дані з різних джерел. Частиною моделі будуть компоненти, засновані на машинному навчанні. Ці компоненти допоможуть у виділенні ключової інформації, уніфікації даних з різних сигналів, а також в прогнозуванні можливих дефектів або проблем в металевих конструкціях. Модель повинна працювати в реальному часі, що є критично важливим для ультразвукового контролю. Очікується, що реалізація компонентного підходу допоможе створити ефективну основу для системи контролю якості металевих конструкцій за допомогою ультразвукових методів, яка буде відзначатися високою точністю та надійністю.

ISSN 1993–9965 print
ISSN 2415–3524 online
Науковий вісник ІФНТУНГ
2024. № 1(56)

Висвітлення основного матеріалу дослідження

Розглянемо теоретичні основи підходу, що включають кінцеві автомати стану (КАС) та конструктивну взаємодію моделі (КВМ). Кінцеві автомати стану можемо описати такими термінами, як кортеж [4]:

$$A=\left(Q,\Sigma,\delta,q_0,F\right), \qquad (1)$$

де Q – кінцеві автомати (або машини станів);

Σ – множина символів, що є "алфавітом";

$\delta: Q\times\Sigma\to Q$ – часткова функція переходу;

$q_0 \in Q$ – початковий стан;

$F \subseteq Q$ – множина приймаючих станів.

Автомат приймає мову, що складається з набору послідовностей символів $L(A)$, якщо від початкового стану можливо перейти до приймаючого стану, слідуючи визначеним переходам. Мова моделі автомата має наближати множину журналів, які може виробляти система.

Розглянемо конструктивну взаємодію моделі. Завданням КВМ є відтворення автоматів із вибірки вихідних слів. Існує багато підходів до цього завдання, які можна поділити на дві категорії: 1) активні методи навчання, які постійно звертаються до системи, над якою ведеться навчання, щоб задати питання; 2) пасивні методи навчання, які виводять автомати на основі позитивних прикладів без додаткових запитів до КВМ. Активні методи навчання важко впровадити, адже жива система повинна відповідати на питання, а це може бути проблематичним з точки зору масштабованості. Результати, засновані на певних евристиках, можуть не бути загальними. Це пасивний метод навчання, який активно використовує знання домену, зокрема стосовно використаної архітектури програмного забезпечення. Ця модель створена з метою відображення ментальних моделей авторів програмного забезпечення. Програмне забезпечення, що складається з кількох компонентів, кожен з яких надає набір внутрішніх або зовнішніх служб. Служба може бути розділена на декілька частин, які називаються фрагментами служб. Наприклад, якщо служба визначає зворотний виклик після асинхронної дії, то ця функція і зворотний виклик відносяться до однієї і тієї ж служби, але вони розглядаються як окремі фрагменти служб. Кожен фрагмент служби може бути процедурою, яка обробляє внутрішнє або зовнішнє повідомлення. Припускається, що компоненти виконують лише один фрагмент служби одночасно і фрагменти служби не можуть бути перерваними. Вхідні дані КВМ представляють собою трасу виконання системи, яка містить часові події, кожна з яких відповідає повідомленню між фрагментами служб. Події повинні йти парами: подія "Вхід" та подія "Вихід". Ці траси виконання моделюються за допомогою часових послідовних діаграм сполучень (ЧПДС). КВМ використовують ЧПДС як вхідні дані.

Метод для моделювання систем на основі виконаного коду програми

На рис. 1 показано, як працює КВМ. Все починається з блоку трасування виконання системи, що представляє виконання програми. Це трасування розбивається на окремі для кожного компонента системи і називається трасуванням компонентів. Подальші трасування розбиваються на фрагменти, які відповідають окремим частинам виконання служб в системі. Для кожного службового фрагмента створюється автомат, відомий як автоматизоване дерево префіксів (АДП). Це детермінований кінцевий автомат, що має унікальний стан прийняття для кожного слова в його мові. Згодом, автомати об'єднуються в більші структури. Наприклад, усі початкові стани автоматів службових фрагментів об'єднуються в один, що призводить до створення автомата у вигляді "квітки", де

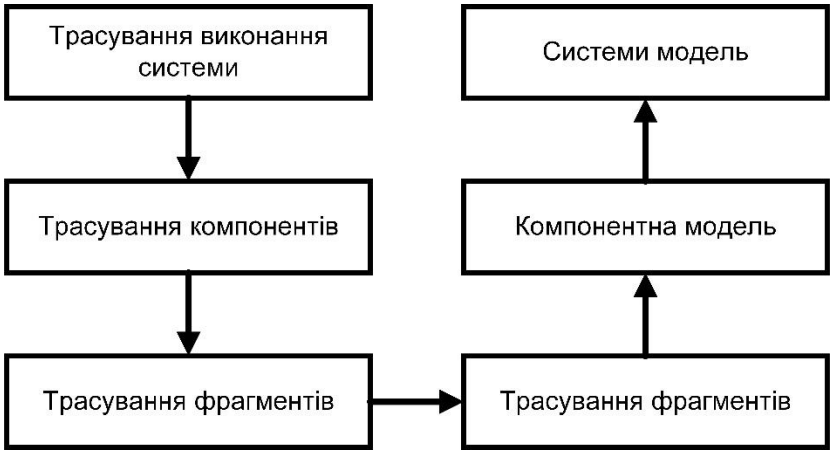

Рисунок 1 – Діаграма конструктивної взаємодії компонентів моделі

центральний стан є станом очікування. Автомати компонентів можуть бути об'єднані в одну системну модель за допомогою асинхронної композиції. Це дозволяє моделювати асинхронну комунікацію між компонентами. Існують певні правила або обмеження для моделей, створених за допомогою КВМ. Наприклад, для кожної події " Вхід" повинна існувати відповідна подія " Вихід". Ці обмеження допомагають забезпечити правильність і надійність моделей.

Порівняння станових машин

Основна ідея полягає в тому, що коли у нас є автомати (це специфічні структури даних) для різних версій програмного забезпечення, нам потрібен спосіб знаходити і виділяти різницю між ними. Метод, описаний в [5], допомагає виділити структурні відмінності між двома автоматами. Вони створили алгоритм під назвою LTSDiff, який виконує таке порівняння. LTSDiff працює так: спочатку алгоритм розраховує подібність між парами станів обох автоматів, зважаючи на їх переходи. Після цього найбільш схожі стани помічаються як "орієнтири". Процес продовжується так, що алгоритм крок за кроком додає до списку найбільш схожі стани, поки всі не будуть обрані. Якщо є стани або переходи в одному автоматі, яких немає в іншому, вони вважаються структурними відмінностями. Ці відмінності можна виділити за допомогою різниці автоматів та позначати кольорами, де додані елементи зелені, а видалені – червоні. Розроблене програмне забезпечення мовою програмування Java під назвою MIDs, яке використовує узагальнену версію LTSDiff, відому як gLTSDiff і є у загальному доступі.

Метод порівняння, який працює на шести рівнях. Цей підхід [4] фокусується на порівнянні моделей між собою, зокрема на розрізненні структурних відмінностей. Різниця в автоматах показує структурні відмінності на найнижчому рівні навчених моделей, починаючи від окремих переходів. При аналізі змін між версіями програмного забезпечення може бути корисним спочатку отримати загальний огляд системи. MIDs використовує цей підхід для порівняння різних наборів моделей, які містять моделі сутностей (у цьому контексті - фрагменти послуг). Існує шість рівнів деталізації, на яких можна порівнювати моделі:

Рівень 1 (Model Set Variants): перевіряє, чи приймають набори моделей однакову мову, тобто мають ідентичну поведінку.

Рівень 2 (Model Set Variant Relations): показує відносини мовної інклюзії між наборами моделей.

Рівень 3 (Model Set Variant Differences): вказує на кількість різних сутностей для кожної пари наборів моделей.

Рівень 4 (Model Variants): перевіряє, чи приймають моделі сутностей однакову мову.

Рівень 5 (Model Variant Relations): показує відносини мовної інклюзії між моделями.

Рівень 6 (Model Variant Differences): показує структурну різницю між моделями сутностей, як це було описано раніше.

Обмеження в дослідженні: їх роботи рівні 1-3 менш релевантні, оскільки вони обмежуються лише двома наборами моделей. Таке обмеження було введено для зменшення складності реалізації і тому, що при роботі з даними про час можна інтуїтивно описувати зміни як "швидше" або "повільніше".

Методика дослідження

Розглянемо випадки, коли для порівняння доступні пари КАС з різними часовими анотаціями. Процес комбінування двох автоматів у єдиний автомат різниці за допомогою імплементації алгоритму gLTSDiff, потребує злиття відповідних станів та переходів. Цей процес вимагає узгодження відповідних станів та переходів двох автоматів, а потім їх об'єднання в один. Нам необхідно змінити алгоритм, щоб правильно обробляти комбінацію переходів, зокрема нашу функцію анотації часу *T*. Ми визначаємо, що переходи автомату різниці двох КАС мають бути анотовані впорядкованою парою можливо порожніх часових зразків, де перший елемент відповідає часовим значенням вихідного трейсу T_L, а другий — часовим значенням зміненого трейсу T_R. Символи переходів у цих автоматах відповідають повідомленням, які обмінюються між фрагментами сервісу. Мова, яку приймає навчений автомат, є наближенням набору вихідних трейсів, які могли б бути згенеровані системою.

Для аналізу часової поведінки системи ми асоціюємо послідовність абсолютних часових міток з кожним переходом автомата, що відповідають часам, коли відбулася подія. Співвідношення $T : Q \times \Sigma \to \Re$ зіставляє переходи з кінцевою послідовністю невід'ємних часових міток у хронологічному (тобто зростаючому) порядку. Ці послідовності називаються часовими зразками *s*. Через спосіб побудови автоматів з вхідного трейсу, подія, представлена переходом, відбулася принаймні один раз, що відображається у факті, що часові зразки повинні бути непорожніми.

ISSN 1993–9965 print ISSN 2415–3524 online
Науковий вісник ІФНТУНГ 2024. № 1(56)

Розглянемо, для прикладу, випадок (рис. 2), де

$$Q=\{s_0,s_1\},\ \Sigma=\{a,b\},$$
$$\delta(s_0,a)=s_1,\ \delta(s_1,b)=s_2,$$
$$q_0=s_0,\ F=\{s_0\},\quad (2)$$
$$T(s_0,a)=[1,3],\ T(s_1,b)=[2,4],$$

де a, b – події; $s_1 \ldots s_n$ – часові зразки.

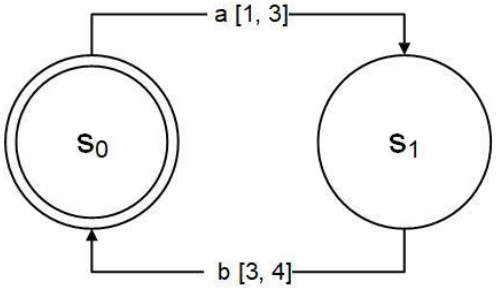

Рисунок 2 – Приклад роботи КАС

З попереднім кроком ми отримали модель системи, що складається з одного анотацією часу до КАС для кожного фрагмента сервісу. Оброблювані нами трейси містять абсолютні значення часу в наносекундах з моменту початку епохи Unix (00:00:00 UTC 1 січня 1970 р.). Ці значення передаються до моделей, тому кожен перехід анотований послідовністю міток часу Unix з точністю до наносекунди.

Це подання є корисним для деяких аналізів, наприклад, для обчислення тривалості між двома довільними подіями, такими як початок і кінець фрагменту сигналу. Однак для інших завдань, зокрема на рівні фрагментів сигналів, це подання може бути некорисним або навіть недоліком. Інженерам необхідно оцінювати продуктивність системи, і для цього тривалості набагато корисніші, ніж дискретні точки в часі. Строго кажучи, мітки часу Unix представляють тривалості. Однак тривалості з більш пізніми початковими точками набагато легше інтерпретувати та порівнювати: наприклад, час, що минув з моменту запуску виконання системи, або час, що минув з моменту останнього виклику функції. Менші числа також легші для читання, ніж повні мітки часу Unix з точністю до наносекунди (наприклад, 845032054749200).

Важливим також є виявлення змін часової поведінки та регресій за часом, але ми не даємо чіткого визначення того, що таке значна зміна часової поведінки. Зробити це важко, оскільки момент, коли два набори часових зразків слід вважати однаковими або значно різними, є динамічним та суб'єктивним. Він змінюється від системи до системи, від інженера до інженера, та від випадку до випадку використання. Те, що є великою затримкою в одному компоненті, може бути ігнороване як шум – у іншому. Тому автоматичні підходи ніколи не зможуть повністю замінити оцінку часових даних людиною. Коффін писав з цього приводу, що прості графіки та статистичні підсумки, хоча й не є самоціллю, часто вказують, які моделі, тести та інтервали найбільш ясно відповідатимуть на запитання експериментатора" [6]. З цієї причини ми намагаємось застосовувати евристики для виявлення значних змін, також надаємо інженерам корисні візуалізації для ручного огляду. Наші евристики є параметризованими. Це означає, що вони містять такі параметри, як фіксовані пороги значущості, які можуть і повинні бути налаштовані користувачем. Для нашої оцінки параметри були налаштовані відповідно до зворотного зв'язку інженерів компанії, але вони можуть бути змінені для відповідності іншим випадкам використання без значних додаткових зусиль.

Крім виклику іноді нечіткого визначення значних змін часової поведінки, нам потрібно впоратися з викликами, які також виникають у класичному оцінюванні продуктивності та оцінюванні:

- шум. Шум зазвичай неминучий при вимірюванні часових даних. Методи виявлення повинні бути стійкими, так що незначні відхилення у даних ідеально не повинні змінювати прогнози евристик;
- ненормальність. Багато статистичних методів, наприклад стандартний двосторонній *t*-тест Стьюдента чи подібний тест на розмах Тьюкі, вимагають, щоб розподіли були нормальними. Це припущення часто не виконується для даних з огляду продуктивності;
- малі розміри вибірки. Ця проблема специфічна для нашої програми та посилює дві попередні проблеми. У системах, де продуктивність програмних блоків можна вимірювати окремо, як це робиться у бенчмарк-фреймворках, таких як BenchmarkDotNet2 або Criterion3, можна легко вплинути на кількість вимірювань. Повторюваність є ключовою вимогою для надійних вимірювань продуктивності. Ми ж працюємо з реальними трейсами, які захоплюють реальні робочі навантаження, і не маємо контролю над розмірами вибірки. Внаслідок цього ми часто маємо справу з малими розмірами вибірки. Мала вибірка часто відповідає експоненційному розподілу, де багато фрагментів сервісів викликаються лише кілька разів (1-100), а декілька фрагментів сервісів викликаються багато разів (>1000). Ми спостерігали подібні

Таблиця 1 – Трасування системи

Трасування 1		Трасування 2		Трасування 3	
Мітка часу компоненту	Функція	Мітка часу компоненту	Функція	Мітка часу компоненту	Функція
1	вхід *f*	61	вхід *f*	61	вхід *f*
2	вихід *g*	63	вхід *g*	62	вхід *h*
3	вихід *g*	64	вихід *g*	64	вихід *h*
4	вихід *f*	65	вихід *f*	65	вихід *f*

розподіли кількості викликів у непромислових трейсах.

Алгоритм та результати його роботи

Маючи трасування виконання системи *w* з часовими мітками *t* для КАС, додаймо часові анотації до КАС, щоб отримати КАС в часі (КАСЧ), просто моделюючи переходи автомата по одному і додаючи часові мітки подій при здійсненні переходів (рис. 3).

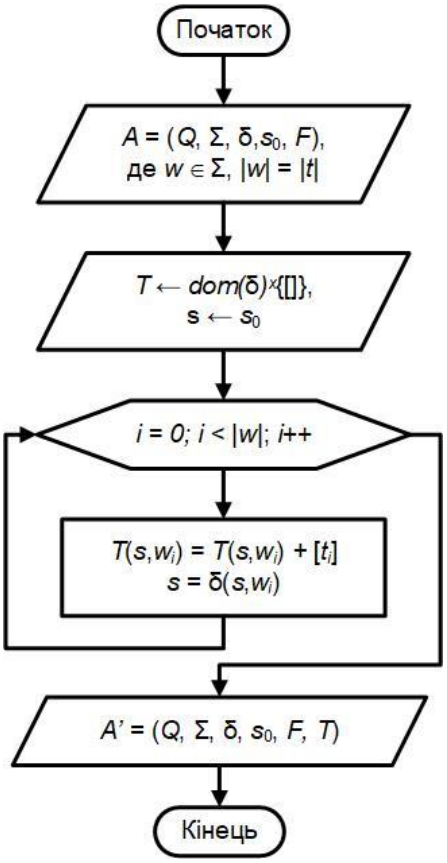

Рисунок 3 – Алгоритм додавання часових анотацій *T* до КАС

На першому етапі алгоритму вхідними даними є вибрана система для трасування. Другим кроком є ініціалізація системи з нульовими часовими послідовностями. На третьому етапі алгоритму відбувається процес анотації часовими мітками на кожному часовому переході.

Для демонстрації роботи алгоритму було використано три мінімальні трасування:

- трасування 1, що включає події, пов'язані з викликами функцій *f* та *g* з відповідними часовими мітками;
- трасування 2, що функціонально еквівалентне трасуванню 1, але подія входу у виклик *g* затримана на одну одиницю часу;
- трасування 3, де виклики *g* замінені на виклики *h* і подія виходу з *h* затримана на одну одиницю часу порівняно з виходом з *g* у трасуванні 1.

Додаючи часові мітки до відповідних переходів, отримуємо КАСЧ, який точно відображає тимчасові аспекти подій. Це дозволяє не лише зберігати інформацію про послідовність подій, але й забезпечує високу точність у відображенні часових характеристик, що є критичним для аналізу продуктивності систем ультразвукового контролю металоконструкцій. У разі монотонного зростання часових міток, можна відновити оригінальне трасування з КАСЧ, завжди вибираючи перехід з мінімально наступною часовою міткою.

Робота інтерфейсу. У інтерфейсі усі елементи: часово-анотовані автомати різниці, евристики виявлення, візуалізації, об'єднуються в єдиний користувацький інтерфейс у Mids+Time. Оскільки ми не можемо показати внутрішні дані ASML, використовуємо відкрите програмне забезпечення cURL для генерації трейсів. Відтак використовуємо інструментовану версію cURL для створення запитів до google.com, один раз використовуючи WiFi-з'єднання, а інший раз — мобільне з'єднання, отримуючи таким чином трейси з різними часовими параметрами. Оскільки cURL сам по собі не є компонентно-орієнтованим програмним забезпеченням у класичному розумінні: ми визначаємо різні блоки компіляції як різні компоненти, виклики функцій між блоками компіляції як комунікацію і функції – як фрагменти сервісу. Найчастіше викликана функція на рис. 4.

ISSN 1993–9965 print
ISSN 2415–3524 online
Науковий вісник ІФНТУНГ
2024. № 1(56)

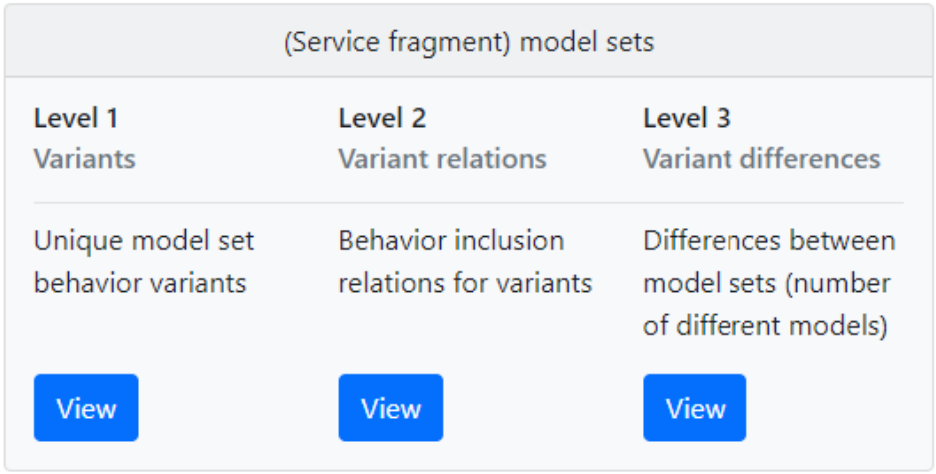

Рисунок 4 – Головне меню MIDS+Time

Наш аналіз трейсів cURL за допомогою Mids+Time показав різні збільшення часу виконання функцій, зокрема у компонентах, пов'язаних з виконанням мережевих запитів та опитуванням, що відображає очікуване збільшення часу відгуку мережі через те, що мобільне з'єднання повільніше, ніж WiFi-з'єднання. Вивід Mids генерується у формі інтерактивної HTML-сторінки, де користувач може окремо доступитись до шести рівнів деталізації з головного меню (рис. 4). Для Mids+Time ми додали сьомий вигляд, сторінку з оглядом часових показників. Вона містить деяку загальну інформацію вгорі: гістограми для розподілу кількості викликів фрагментів сервісу та розподілу часу виконання фрагментів сервісу, а також список 5 найчастіше викликаних фрагментів сервісу та 5 фрагментів сервісу з найтривалішим часом виконання. Вони призначені для надання користувачу негайного огляду будь-яких значних змін. Для наших прикладних трейсів cURL можемо побачити, що час виконання – у зміненому трейсі: багато фрагментів сервісу мають збільшений час виконання. Зміни у списках топ-5 виділені звичайною червоною та зеленою схемою кольорів. Нижче відображається список кожного компонента системи. Він може бути розгорнутий, щоб відкрити підсписок кожного фрагмента сервісу всередині цього компонента. Ці підсписки сортуються за магнітудою змін їхнього часу виконання: фрагменти сервісу, де час між двома вхідними трейсами змінився найбільше, будуть показані зверху. Якщо зразки часу виконання вважаються за значні відмінності, назва фрагмента сервісу буде показана пурпуровим кольором. Кожен елемент у списку фрагментів сервісу містить невеликий попередній перегляд автомата різниці фрагмента сервісу, а також гістограму, контрольну карту та графік зміщення часових даних виконання. Деталізація фрагменту сервісу відкриває таблицю з розміром вибірки (тобто частотою викликів), мінімумом, 25-м перцентилем, медіаною, 75-м перцентилем та максимальним часом виконання для лівого та правого трейсів відповідно. Якщо часові зразки виконання фрагмента сервісу виявлені як значно відмінні нашими евристиками, запис у списку буде виділений пурпуровою межею та пурпуровим заголовком.

Висновки

Запропоновано методологію для аналізу часової поведінки компонентно-орієнтованих програмних систем. Вона базується на системних моделях Mids, інструменту для виведення моделей з трейсів виконання та порівняння системних моделей різних версій програмного забезпечення. Ми витягуємо дані про час з трейсів виконання системи та додаємо їх до моделей систем на основі автоматів, анотуючи переходи послідовностями часових міток. На основі цих даних можна розрахувати різні метрики, такі як час виконання функції. Також розробляємо метод виявлення змін часової поведінки, який поєднує різні прості евристики для обробки шуму, ненормальності та малих розмірів вибірки. Більше того, ми показуємо, як дані про час можуть бути інтуїтивно візуалізовані за допомогою різних видів графіків і шляхом розширення концепції автомата різниці Mids та його користувацького інтерфейсу. Оцінюємо наш метод на реальних трейсах літографічних машин ASML, знаходячи його здатним допомагати інженерам у виявленні реальних різниць у часовій поведінці простим і ефективним способом. Не зважаючи на те, що ми не були знайомі з програмними системами та вихідним кодом,

який генерував ці трейси, нам вдалося отримати базове розуміння системи та знайти та проаналізувати регресії продуктивності, які раніше були невідомі фахівцям. За допомогою нашого доказу концепції застосування Mids+Time до відкритого програмного забезпечення cURL показуємо, що наша методологія може бути застосовна до компонентно-орієнтованого програмного забезпечення. Обидва експерименти з трейсами ASML і трейсом cURL демонструють, що можливості виявлення Mids+Time не обмежуються тільки часовими різницями, в сигналах. Розроблений підхід є цікавим при аналізі ультразвукових сигналів, що передаються через розподілену систему вимірювання лініями зв'язку із різними типами фізичного рівня реалізації передавання інформації.

Література / References

1. Khademi P., et al. Time-Frequency Analysis of Ultrasonic Signals for Quality Assessment of Concrete. SSRN. 4397220. URL: https://ssrn.com/abstract=4397220

2. Smoqi Z., et al. Ultrasonic nondestructive evaluation of additively manufactured wear coatings. *NDT & E International*. 2023. Vol. 133. P. 102754.

3. Sun H., Ramuhalli P., Jacob R. E. Machine learning for ultrasonic nondestructive examination of welding defects: A systematic review. *Ultrasonics*. 2023. Vol. 127. P. 106854.

4. Hilbig A. Analysing timing behavior of component-based software systems : Master's Thesis. 2023.

5. Walkinshaw N., Bogdanov K. Automated comparison of state-based software models in terms of their language and structure. *ACM Transactions on Software Engineering and Methodology (TOSEM)*. 2013. Vol. 22, No 2. P. 1–37.

6. Coffin M., Saltzman M. J. Statistical analysis of computational tests of algorithms and heuristics. *INFORMS Journal on Computing*. 2000. Vol. 12, No. 1. P. 24–44.