УДК 004.2:004.414
DOI https://doi.org/10.32782/2521-6643-2025-2-70.20

Симонов Д. І., доктор філософії, науковий співробітник, завідувач лабораторії проблем прикладної інформатики Інституту кібернетики ім. В.М. Глушкова Національної академії наук України
ORCID: 0000-0002-6648-4736

Деменко І. О., аспірант Інституту кібернетики ім. В.М. Глушкова Національної академії наук України
ORCID: 0009-0008-2683-6523

ФОРМАЛІЗАЦІЯ ПРОЦЕСІВ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ СКЛАДНИХ ПРОГРАМНИХ СИСТЕМ НА ОСНОВІ МУЛЬТИАГЕНТНОГО МОДЕЛЮВАННЯ

Дослідження присвячено формалізації процесів автоматизованого тестування складних програмних систем, що функціонують у динамічних та стохастичних середовищах. Показано, що традиційні методи тестування, побудовані на фіксованих сценаріях, не забезпечують достатнього рівня адаптивності й масштабованості при зростанні складності сучасних програмних продуктів. Обґрунтовано доцільність застосування мультиагентного підходу як інструмента побудови інтелектуальних систем тестування, здатних до самоорганізації, навчання та оптимізації процесу виявлення помилок.

У дослідженні здійснено системно-теоретичний аналіз процесу тестування як динамічної системи, що складається з множини автономних агентів, які взаємодіють між собою та з тестовим середовищем. Запропоновано узагальнену формальну модель мультиагентного процесу тестування, у якій визначено основні структурні елементи – множину агентів, середовище їхньої взаємодії, простір тестових сценаріїв, політики поведінки та функцію оцінювання результатів. Такий підхід дає змогу розглядати процес тестування як еволюційну систему, у якій агенти змінюють власні стратегії на основі отриманого досвіду та зворотного зв'язку, поступово підвищуючи ефективність і повноту перевірки програмного забезпечення.

Розроблено графову та стохастичну модель простору тестових сценаріїв, яка враховує ймовірність виявлення дефектів, вартість і тривалість тестів, ступінь покриття та інші характеристики ефективності. Визначено систему метрик, що охоплює як класичні показники продуктивності тестування, так і специфічні для мультиагентних систем параметри – кооперативність, конфліктність і швидкість збіжності політик агентів. Описано алгоритми перевірки коректності моделі, які забезпечують оцінювання її узгодженості, досяжності станів, стійкості та адекватності в умовах стохастичних збурень.

Отримані результати створюють теоретичну основу для побудови інтелектуальних мультиагентних систем автоматизованого тестування, здатних до адаптації та самоорганізації. Запропонований підхід сприяє підвищенню ефективності тестового процесу, скороченню витрат часу й ресурсів, а також підвищенню рівня надійності програмного забезпечення. Подальші дослідження спрямовуються на розвиток моделей навчання агентів і впровадження отриманих рішень у сучасні CI/CD-процеси.

Ключові слова: *автоматизоване тестування; мультиагентне моделювання; інтелектуальні системи; графова модель; метрики ефективності; адаптивне тестування; штучний інтелект.*

Symonov D. I., Demenko I. O. Formalization of automated testing processes of complex software systems based on multiagent modeling

The study is devoted to the formalization of automated testing processes for complex software systems operating in dynamic and stochastic environments. It is shown that traditional testing methods based on fixed test scenarios do not ensure sufficient adaptability and scalability as the complexity of modern software increases. The research substantiates the feasibility of applying a multi-agent approach as an effective tool for building intelligent testing systems capable of self-organization, learning, and optimization of defect detection processes.

The study performs a system-theoretic analysis of the testing process as a dynamic system composed of multiple autonomous agents interacting with each other and with the testing environment. A generalized formal model of the multi-agent testing process is proposed, which defines the main structural elements: a set of agents, the environment of their interaction, the test scenario space, behavioral policies, and the evaluation function of the obtained results. This approach allows considering testing as an evolutionary process in which agents adapt their strategies based on accumulated experience and feedback, gradually improving the efficiency and completeness of software verification.

A graph-based and stochastic model of the test scenario space is developed, taking into account the probability of defect detection, testing cost and duration, coverage degree, and other efficiency attributes. A system of metrics is defined, encompassing both classical measures of testing performance and multi-agent specific parameters such as cooperativity, conflict rate, and convergence speed of agent policies. Algorithms for model correctness verification are described, providing consistency, reachability, stability, and adequacy assessment under stochastic disturbances.

The obtained results form a theoretical foundation for developing intelligent multi-agent systems of automated testing capable of adaptation and self-organization. The proposed approach enhances the efficiency of the testing process, reduces time and resource costs, and increases software reliability. Future research will focus on developing agent learning mechanisms and integrating the proposed model into CI/CD processes and continuous quality control systems.

Key words: *automated testing; multi-agent modeling; intelligent systems; graph model; efficiency metrics; adaptive testing; artificial intelligence.*

Постановка проблеми. Сучасні програмні системи характеризуються високим рівнем складності, розподіленістю компонентів, динамічною зміною станів та значною кількістю можливих сценаріїв взаємодії. За таких умов класичні методи автоматизованого тестування, орієнтовані на фіксовані набори тестів або жорстко визначені правила покриття, втрачають ефективність. Виникає потреба у побудові інтелектуальних тестувальних систем, здатних до самонавчання, адаптації та оптимізації процесів виявлення помилок у реальному часі.

Існуючі підходи до автоматизованого тестування, зокрема model-based, behavior-driven, data-driven та AI-driven testing, забезпечують лише часткове вирішення окремих завдань – таких як генерація тестових даних, пріоритизація сценаріїв або аналіз результатів. Проте вони не охоплюють проблему узагальненого формального опису процесу тестування як цілісної динамічної системи, у якій одночасно діють множини взаємопов'язаних суб'єктів (агентів), що взаємодіють із зовнішнім середовищем та між собою.

Формалізація процесів тестування у вигляді мультиагентної системи дозволяє подати його як сукупність автономних агентів – тестувальників, системних спостерігачів, генераторів сценаріїв, що діють у спільному середовищі та мають власні цілі, політики поведінки й механізми обміну інформацією. Такий підхід забезпечує динамічне узгодження дій між агентами, колективне прийняття рішень і поступове вдосконалення стратегії тестування завдяки зворотному зв'язку.

Наукова проблема полягає у відсутності універсальної формальної моделі, яка б описувала процеси автоматизованого тестування складних програмних систем як багатокомпонентну, стохастичну та еволюційну систему. Необхідно сформувати теоретичні засади, що дозволяють представити простір тестових сценаріїв, визначити динаміку їх еволюції та формалізувати взаємодію агентів у межах інтелектуальної системи тестування.

Таким чином, актуальним є завдання побудови системно-теоретичної та математичної моделі процесів автоматизованого тестування на основі мультиагентного моделювання, що створює основу для подальшої розробки алгоритмів самонавчання, оптимізації та оцінювання ефективності інтелектуальних тестових систем.

Аналіз останніх досліджень і публікацій. Автоматизоване тестування програмних систем протягом останніх десятиліть стало одним із ключових напрямів забезпечення якості програмного забезпечення. Класичні підходи, такі як data-driven testing, keyword-driven testing, model-based testing, дозволили значно зменшити трудомісткість тестового процесу, проте залишаються переважно декларативними, тобто залежать від попередньо визначених шаблонів і сценаріїв. Унаслідок цього вони мають обмежену здатність до адаптації у складних, розподілених та динамічно змінюваних середовищах.

З появою технологій штучного інтелекту (ШІ) розпочався перехід до інтелектуального тестування (AI-driven testing), у межах якого алгоритми машинного навчання використовуються для автоматичного генерування тестових сценаріїв, прогнозування дефектів, оптимізації покриття та аналізу результатів. Роботи [1–4] демонструють можливість використання методів класифікації, нейронних мереж і reinforcement learning для підвищення ефективності вибору тестів і виявлення аномалій у поведінці системи. Проте більшість підходів орієнтовані на локальні задачі, зокрема на генерацію тестів або прогнозування дефектів, і не забезпечують цілісного уявлення про тестовий процес як про систему з багаторівневою взаємодією.

Водночас в останні роки зростає інтерес до агентно-орієнтованих методів моделювання, що застосовуються у різних галузях – від оптимізації транспортних потоків до управління виробничими системами [5–7]. Агентне моделювання дозволяє відобразити складні колективні процеси, де автономні елементи взаємодіють за певними правилами, адаптуються до змін середовища та здатні до самоорганізації. Цей підхід виявився ефективним також у сфері тестування, де агенти можуть виконувати ролі тестувальників, моніторів, координаторів або генераторів сценаріїв [8–10].

У дослідженнях [11, 12] запропоновано використання мультиагентних архітектур для динамічного розподілу тестових завдань, формування тестових даних і вибору стратегій перевірки. Такі системи характеризуються підвищеною масштабованістю та здатністю до самооптимізації. Водночас більшість робіт зосереджені на архітектурних аспектах реалізації агентів і не приділяють достатньої уваги формалізації процесу тестування як системи взаємодіючих динамічних об'єктів з власними політиками поведінки.

Актуальними залишаються питання побудови математичних моделей простору тестових сценаріїв, визначення метрик складності й надійності тестових алгоритмів, а також перевірки коректності моделей тестування, які враховують стохастичні впливи та фактори невизначеності. Існують підходи, що використовують графові та ймовірнісні методи для моделювання процесу тестування, проте вони не охоплюють інтерактивну взаємодію між агентами, які приймають рішення на основі колективного досвіду.

Отже, незважаючи на значний прогрес у сфері автоматизації тестування та впровадження інтелектуальних методів, відсутня узагальнена формальна модель, яка б поєднувала системно-теоретичний підхід, агентну взаємодію та стохастичну природу процесів тестування. Саме ця прогалина зумовлює необхідність подальших досліджень у напрямі формалізації процесів автоматизованого тестування складних програмних систем на основі мультиагентного моделювання.

Метою статті є розроблення системно-теоретичних і математичних основ формалізації процесів автоматизованого тестування складних програмних систем із використанням мультиагентного моделювання.

Для досягнення цієї мети передбачається:

- представити процес автоматизованого тестування як динамічну систему, що складається з множини взаємодіючих агентів із власними стратегіями поведінки та інформаційними політиками;
- сформулювати математичну модель простору тестових сценаріїв на основі графових і стохастичних підходів;
- визначити метрики складності, надійності та ефективності агентної взаємодії в процесі тестування;
- розробити підходи до перевірки коректності та узгодженості формалізованої моделі.

Реалізація поставленої мети створює теоретичні передумови для побудови інтелектуальних систем автоматизованого тестування, здатних до самоорганізації, адаптації та навчання в умовах змінного середовища.

Виклад основного матеріалу. Системно-теоретичні передумови формалізації процесів тестування. Процес тестування програмних систем може розглядатися як складна динамічна система, що функціонує у відкритому середовищі, реагує на зовнішні впливи та змінює власні параметри у часі. Відповідно до системно-теоретичного підходу [5], будь-яку таку систему можна подати у вигляді:

$$\mathfrak{S} = S, U, Y, \tag{1}$$

де S – множина внутрішніх станів системи, U – множина вхідних впливів (тестових дій), Y – множина вихідних результатів (реакцій системи).

У випадку автоматизованого тестування елементи цієї трійки описують поточний стан програмного продукту, тестові стимули, що застосовуються до нього, та результати у вигляді поведінкових або логічних відповідей системи.

Складність сучасних програмних систем полягає у великій кількості компонентів, які взаємодіють асинхронно, мають власні правила обміну даними та можуть змінювати стан під впливом зовнішніх подій. Тестування таких систем неможливо ефективно реалізувати в межах централізованої моделі управління – потрібна децентралізована структура, здатна до самоорганізації та автономного прийняття рішень.

У межах системного підходу природним є представлення процесу тестування як мультиагентної системи (МАС), у якій кожен агент виконує певну роль у спільному середовищі:

- агенти-генератори тестів – формують нові тестові сценарії або варіанти вхідних даних на основі покриття та попередніх результатів;
- агенти-виконавці – здійснюють запуск тестів, спостереження за станом системи та збір метрик;
- агенти-аналізатори – інтерпретують результати, виявляють невідповідності, оновлюють політики тестування;
- агенти-координатори – узгоджують дії інших агентів, забезпечуючи оптимальний розподіл ресурсів та пріоритизацію сценаріїв.

Функціонування мультиагентної тестової системи відбувається в умовах інформаційної невизначеності та стохастичних впливів, що зумовлює потребу у врахуванні таких властивостей, як:

- адаптивність – здатність агентів змінювати стратегії поведінки залежно від результатів тестів;
- самоорганізація – формування колективної стратегії тестування без централізованого управління;
- еволюційність – накопичення досвіду агентами у вигляді оновлення політик або параметрів моделей;
- стійкість – здатність системи зберігати працездатність у випадку збурень чи відмов окремих агентів.

Математично взаємодію між агентами можна описати як сукупність функціональних залежностей:

$$S_{t+1} = f_i\left(S_t, u_{i,}(t), m_i(t), E_t, \eta_i(t)\right), \tag{2}$$

де S_t – стан середовища у момент часу t; $u_{i,}(t)$ – дія агента i; $m_i(t)$ – повідомлення, отримані від інших агентів; E_t – зовнішній контекст (стан тестованої системи, зовнішні події); $\eta_i(t)$ – стохастична компонента, що відображає непередбачувані флуктуації.

Такий опис дозволяє перейти від інтуїтивного уявлення процесу тестування до його формалізованої моделі, придатної для математичного аналізу, моделювання й подальшої автоматизації процесів прийняття рішень у тестуванні.

Формалізація процесів тестування. Автоматизоване тестування складних програмних систем потребує чіткого формалізованого підходу, який дозволяє описати не лише окремі дії тестувальника, а й взаємодію множини агентів у динамічному середовищі, а також стохастичний характер тестового процесу. Така формалізація створює основу для математично обґрунтованого прийняття рішень, забезпечує адаптивність та дозволяє враховувати взаємозалежності дій агентів і стан середовища.

Для формалізації процесу розглянемо систему, яка складається з множини агентів $A=\{a_1,a_2,\ldots,a_n\}$, де кожен агент a_1 характеризується власним станом $x_1(t)\in X_i$, множиною можливих дій U_i, функцією переходів f_i та множиною повідомлень M_j, які він може надсилати або отримувати. Сукупний стан системи в момент часу t подається вектором $S(t)=(x_1(t),x_2(t),\ldots,X_n(t))$.

Динаміка стану агента описується рівнянням:

$$x_i(t+1)=f_i(x_i(t),u_{i,}(t),m_i(t),E_t,\eta_i(t)), \tag{3}$$

де E_t – стан тестованої системи, а $\eta_i(t)$ – випадковий збурюючий вплив, що моделює невизначеність або похибку вимірювань.

Таким чином, кожен агент одночасно реагує на внутрішній стан, зовнішнє середовище та повідомлення інших агентів, що відображає складність реального процесу тестування.

Оскільки агенти змінюють свої стани у часі та взаємодіють між собою, для опису структури тестового процесу доцільно застосувати графову модель:

$$G=V,E, \tag{4}$$

де $V=\{\nu_1,\nu_2,\ldots,\nu_m\}$ – множина можливих станів системи, а $E\subseteq V\times V$ – множина переходів, що відповідають виконанню конкретних тестових дій.

Кожному ребру $e_{ij}=(\nu_i,\nu_j)$ відповідає тестовий сценарій T_{ij} з атрибутами:

- мовірність виявлення дефекту p_{ij};
- вартість виконання c_{ij};
- час виконання t_{ij};
- ступінь покриття cov_{ij}.

На основі цих атрибутів визначається функція ефективності тестування:

$$F(T_{ij})=\alpha_1 p_{ij}+\alpha_2 cov_{ij}-\alpha_3 c_{ij}-\alpha_4 t_{ij}, \tag{5}$$

де коефіцієнти α_k відображають важливість кожного фактора. Завдання оптимізації полягає у максимізації цього функціонала:

$$F^*=\max_{T_{ij}} F(T_{ij}), \tag{6}$$

що дозволяє обирати найбільш ефективні сценарії з урахуванням ресурсних обмежень.

Опис графової моделі природно веде до розуміння того, що вибір сценарію залежить не тільки від його атрибутів, а й від поведінки агентів, яка змінюється стохастично.

Взаємодію агентів тестової системи можна подати у вигляді марковського процесу або стохастичної гри, де стан агента змінюється залежно від результатів попередніх дій:

$$P(x_i(t+1))=S_j\,|\,x_i(t)=s_k,u_i(t)=p_{kj}(i), \tag{7}$$

де $p_{kj}(i)$ – елемент матриці переходів, що відображає адаптацію політики агента.

Таким чином, процес тестування розглядається як послідовність випадкових подій, де поведінка агентів еволюціонує на основі зворотного зв'язку від середовища. Це дозволяє формувати адаптивні стратегії тестування, в яких агенти поступово оптимізують свої дії залежно від результатів попередніх тестів.

Поєднуючи опис агентів, графову модель сценаріїв та стохастичну поведінку, мультиагентний процес тестування можна представити як:

$$M=A,E,G,P,R, \tag{8}$$

де: A – множина агентів, E – середовище, у якому агенти функціонують, G – граф тестових сценаріїв, P – множина політик агентів, R – функція винагороди, що визначає критерії успішності (наприклад, кількість знайдених дефектів або коефіцієнт покриття).

Ця модель дозволяє системно аналізувати ефективність тестування, визначати метрики складності та надійності системи, а також формалізувати критерії адаптивності та оптимізації процесу.

Метрики та критерії оцінки ефективності мультиагентного тестування. Оцінювання якості та ефективності процесів автоматизованого тестування є ключовим завданням при побудові формальної моделі мультиагентної системи. У системно-теоретичному контексті метрики виступають як зовнішні функціонали, що відображають ступінь досягнення цілей тестування, а також характеризують поведінку, узгодженість і стабільність взаємодії агентів.

Для систематизації підходів до оцінки ефективності метрики мультиагентного тестування можна поділити на три основні групи:

1. метрики покриття та продуктивності тестування;
2. метрики взаємодії та координації агентів;
3. метрики складності та надійності тестового процесу.

Метрики покриття та продуктивності тестування. Найпоширенішим показником є ступінь покриття (COV), що визначається як відношення кількості протестованих елементів системи до їх загальної кількості:

$$COV = \frac{N_{tested}}{N_{total}} \times 100\%, \tag{9}$$

де N_{tested} – кількість протестованих елементів, а N_{total} – загальна кількість елементів системи.

Ступінь покриття є фундаментальною метрикою, оскільки він дозволяє оцінити, наскільки повно тестовий набір охоплює функціональні та структурні компоненти системи. Високий показник покриття свідчить про те, що тестування охоплює більшу частину функціоналу та ймовірність пропуску дефектів зменшується.

Додатково ефективність тестування можна оцінювати за допомогою mutation score (MS), який відображає здатність тестів виявляти помилки. Цей показник визначається як відсоток «знищених» (виявлених) мутацій від загальної кількості штучно створених мутацій у коді:

$$MS = \frac{N_{killed}}{N_{mutants}} \times 100\%, \tag{10}$$

де N_{killed} – кількість виявлених помилок, а $N_{mutants}$ – загальна кількість мутацій.

Метрика MS дозволяє оцінити чутливість тестів до дефектів, тобто наскільки добре тестовий набір виявляє потенційні помилки, що не були явно передбачені у вимогах.

Таким чином, метрики покриття забезпечують оцінку результативності тестів на рівні всієї системи. Проте у контексті мультиагентного тестування важливо не лише знати, що тестовані елементи охоплені, а й оцінювати взаємодію агентів, тобто ступінь їх координації та узгодженості дій під час виконання тестових сценаріїв. Такий підхід дозволяє виявити неефективності у колективній поведінці агентів і забезпечити більш точне прогнозування ефективності адаптивного тестового процесу.

Метрики взаємодії агентів. У мультиагентних системах ефективність тестування визначається не лише результативністю окремих тестів, а й координацією та узгодженістю дій агентів. Колективна поведінка агентів має вирішальне значення, оскільки від того, наскільки узгоджено виконуються тестові сценарії, залежить швидкість виявлення дефектів, економія ресурсів і загальна стабільність системи.

Нехай $u_i(t)$ – дія агента i у момент часу t, а $\pi_i(t)$ – його стратегія (політика дій). Міра кооперативності агентів може бути подана як коефіцієнт кореляції між діями:

$$K_{coop} = \frac{1}{N(N-1)} \sum_{i \neq j} corr(u_i, u_j), \tag{11}$$

де $corr(u_i, u_j)$ – коефіцієнт кореляції між діями агентів i та j. Високе значення K_{coop} свідчить про узгодженість дій та колективну оптимізацію тестового процесу. Така оцінка дозволяє визначити, наскільки агентна взаємодія ефективно спрямована на досягнення спільної мети тестування, зменшуючи дублювання дій і витрати ресурсів.

Коефіцієнт конфліктності визначається як:

$$K_{conf} = 1 - K_{coop}, \tag{12}$$

що дозволяє оцінити ступінь суперечливості дій між агентами. Високий K_{conf} може сигналізувати про дублювання тестів або неефективне використання ресурсів, що підкреслює необхідність коригування стратегій агентів для підвищення узгодженості.

Ще однією важливою характеристикою є швидкість збіжності політик агентів, яка оцінює стабільність процесу самоорганізації та навчання в мультиагентній системі

$$V_{conv} = \frac{1}{N} \sum_{i=1}^{N} \frac{\| \pi_i(t+1) - \pi_i(t) \|}{\| \pi_i(t) \|}, \tag{13}$$

Низьке значення V_{conv} свідчить про стабільність стратегій агентів і ефективне узгодження дій у часі, тоді як високі значення можуть вказувати на нестабільність колективної поведінки та необхідність додаткової оптимізації алгоритмів координації.

Таким чином, метрики взаємодії агентів дозволяють оцінити узгодженість, конфліктність і стабільність колективної поведінки, а разом із метриками покриття – ефективність співпраці агентів і якість мультиагентної системи.

Метрики складності та надійності тестового процесу. Складність процесу тестування можна розглядати як функцію кількості можливих станів системи, тестових сценаріїв та взаємодій між ними. Для графової моделі $G = V, E$ складність тестового простору визначається як:

$$C_{graph} = |V| + |E| + \sum_{v_i \in V} \deg(v_i), \tag{14}$$

де $\deg(v_i)$ – ступінь вершини v_i, що відображає кількість можливих переходів (тестів), що виходять із даного стану.

Збільшення значення C_{graph} свідчить про зростання структурної складності, що вказує на більшу кількість можливих взаємодій, варіантів розвитку тестового процесу та потенційно вищі витрати на його повне покриття. Такий підхід дозволяє систематично оцінювати складність тестового простору й визначати, які ділянки системи потребують більш ретельного контролю.

Надійність тестового процесу характеризує здатність мультиагентної системи автоматизованого тестування правильно виявляти дефекти навіть за наявності стохастичних збурень і невизначеностей. Формально, вона визначається як:

$$R(t) = 1 - P_{err}(t), \tag{15}$$

де $P_{err}(t)$ – ймовірність помилкової або неповної ідентифікації дефекту у момент часу t. Високе значення $R(t)$ означає, що система демонструє стабільність у виявленні помилок і є надійною у практичному застосуванні.

Застосування цих метрик дає змогу не лише оцінювати ефективність тестування з точки зору покриття та кооперації агентів, але й аналізувати структурну складність процесу, виявляти вузькі місця та потенційні ризики. Крім того, вони створюють основу для побудови адаптивних стратегій управління мультиагентними системами тестування, підвищення їхньої стійкості та оптимізації ресурсів у складних програмних середовищах.

Алгоритми перевірки коректності формалізованої моделі. Перевірка коректності формалізованої моделі процесів автоматизованого тестування є важливим етапом її валідації. Під коректністю розуміють узгодженість між структурними, поведінковими та стохастичними аспектами моделі, а також відповідність її математичного опису реальним властивостям тестового процесу. Для цього застосовують аналітичні, алгоритмічні та симуляційні методи, що забезпечують перевірку від логічної структури до поведінки системи у часі.

На початковому етапі виконується структурна перевірка моделі тестування, поданої у вигляді графа сценаріїв. Її мета – виявити циклічні залежності без умов завершення, непов'язані підграфи та суперечливі переходи. Для цього використовується алгоритм обходу графа в глибину (DFS) із перевіркою досяжності всіх вершин.

На рис. 1 подано фрагмент графової моделі тестового простору, що демонструє результати структурного аналізу. Червоним позначено цикл deadlock, який утворює замкнену послідовність без умов завершення, а сірим – ізольований вузол F, недосяжний для інших станів системи. Такий аналіз дозволяє виявити логічні помилки структури моделі до виконання стохастичного аналізу.

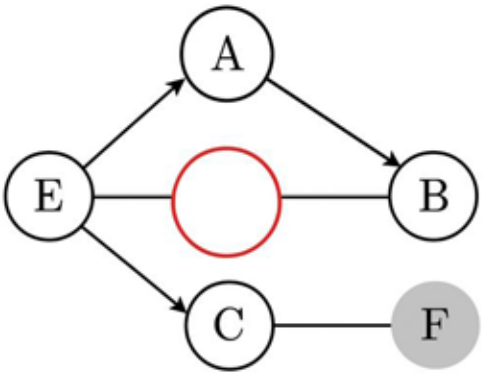

Рис. 1. Фрагмент графової моделі тестового простору з виявленням циклу deadlock та ізольованого вузла

Наступним кроком є оцінка повноти тестового простору шляхом аналізу досяжності станів. Така перевірка визначає, чи забезпечує модель можливість переходу в усі передбачені стани системи. Для практичної

ілюстрації проведено тестування невеликого модуля системи керування замовленнями, де агенти перевіряють сценарії створення, редагування та видалення об’єкта.

На рис. 2 подано граф станів процесу тестування модуля керування замовленнями. Вузол New відповідає створенню замовлення, Assigned – його обробці, Pending (виділений сірим) – стану очікування, Complete – завершенню, а Canceled – скасуванню замовлення. Аналіз досяжності показав, що всі стани покриваються тестами, крім неактивного стану Order_Canceled, який спочатку не мав вхідного переходу. Після додавання зв’язку між станами модель стала повною.

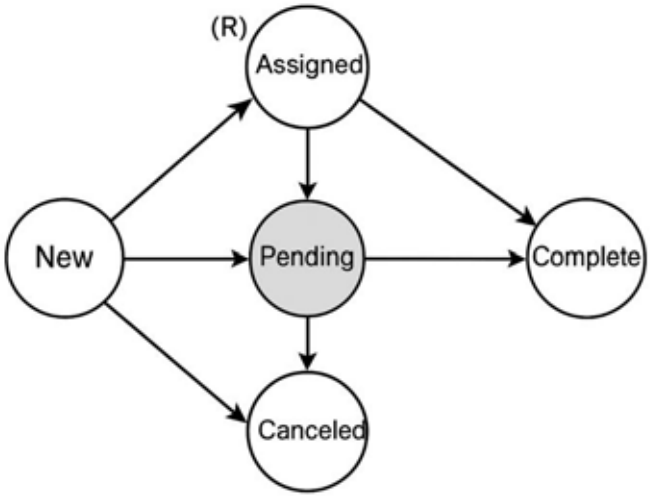

Рис. 2. Граф станів процесу тестування модуля керування замовленнями з позначенням непокритих станів

Варто пам’ятати, що у мультиагентній системі поведінка кожного агента змінюється під впливом результатів попередніх тестів. Тому важливою характеристикою є стійкість – здатність системи повертатися до стабільного стану після зовнішніх збурень, таких як поява дефектів або невизначеність у даних.

На рис. 3 подано графік збіжності політик агентів у процесі симуляційного тестування. По осі X відкладено кількість ітерацій, а по осі Y – рівень узгодженості політик. Залежність має сигмоїдну форму: на початку спостерігається швидке зростання узгодженості, яке поступово переходить у стаціонарний режим після приблизно 12–15 ітерацій. Це свідчить про збіжність поведінкових політик агентів до спільної оптимальної стратегії, що підтверджує стійкість системи та коректність динамічного компонента моделі.

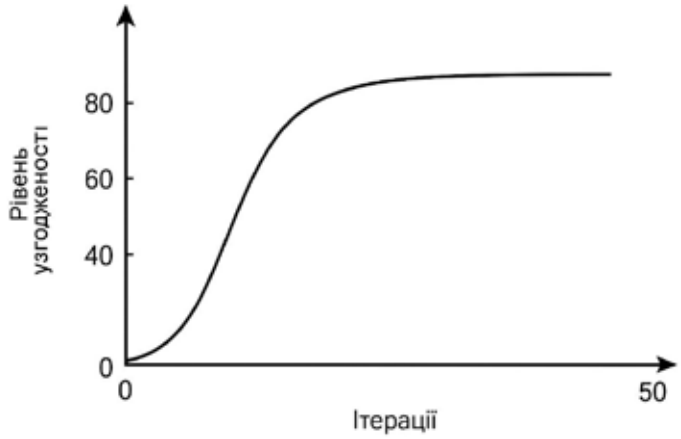

Рис. 3. Графік збіжності політик агентів у процесі симуляційного тестування

Завершальний етап передбачає імітаційне тестування моделі з використанням агентного середовища JADE. Було змодельовано 100 тестових запусків для трьох типів агентів: генератора тестів, аналізатора результатів і координатора. Середня точність виявлення дефектів склала 94 %, а середній час збіжності системи до стабільного режиму – 4,2 с. Порівняння результатів симуляції з реальними сценаріями підтвердило узгодженість моделі та її придатність для практичного застосування.

У табл. 1 наведено зведені результати імітаційного тестування для основних агентів моделі, включно з кількістю виконаних дій, середньою точністю та часом досягнення стабільного стану.

Таблиця 1

Результати імітаційного тестування агентної моделі у середовищі JADE

Тип агента	Кількість запусків	Середня точність, %	Час збіжності, с	Основна функція
Генератор тестів	100	92	3.8	Формування тестових сценаріїв
Аналізатор результатів	100	95	4.4	Оцінка результатів і виявлення дефектів
Координатор	100	96	4.3	Узгодження дій агентів і прийняття рішень
Середнє значення	–	94	4.2	–

Узагальнюючи, оновлена методика перевірки передбачає три взаємопов'язані рівні контролю – структурний, поведінковий та імітаційний. Їх поєднання забезпечує цілісну оцінку коректності формалізованої моделі, дозволяє виявити слабкі місця у взаємодії агентів і підтверджує її адекватність до реальних процесів тестування програмних систем.

Висновки. У роботі сформульовано теоретичні основи формалізації процесів автоматизованого тестування складних програмних систем на базі мультиагентного моделювання. Проведений аналіз існуючих підходів до автоматизації тестування засвідчив обмеження традиційних методів та підтвердив доцільність використання агентно-орієнтованої парадигми, що дозволяє відобразити динаміку, адаптивність і колективну поведінку компонентів тестової системи.

Запропоновано формальну модель мультиагентного процесу тестування, яка поєднує графове та стохастичне представлення простору тестових сценаріїв, визначено основні метрики ефективності, складності та надійності, а також окреслено алгоритмічні засади перевірки коректності моделі. Результати дослідження створюють теоретичне підґрунтя для розроблення інтелектуальних систем тестування, здатних до самоорганізації, навчання та адаптації у змінних програмних середовищах.

Подальші дослідження спрямовуватимуться на уточнення математичних моделей взаємодії агентів, розроблення механізмів навчання з підкріпленням і проведення експериментальної перевірки ефективності запропонованих рішень у реальних умовах програмної інженерії.

Список використаних джерел:

1. Helmy M., Sobhy O., ElHusseiny F. AI-Driven Testing: Unleashing Autonomous Systems for Superior Software Quality Using Generative AI. International Telecommunications Conference (ITC-Egypt). 2024. DOI: https://doi.org/10.1109/ITC-Egypt61547.2024.10620598
2. Kesavan E. AI-Driven and Autonomous Testing. International Scientific Journal of Engineering and Management. 2024. DOI: https://doi.org/10.55041/isjem01651
3. Nama P. Integrating AI in testing automation: Enhancing test coverage and predictive analysis for improved software quality. World Journal of Advanced Engineering Technology and Sciences. 2024. DOI: https://doi.org/10.30574/wjaets.2024.13.1.0486
4. Doddapaneni J. AI Test Design and Script Generator: Enhancing Software Testing through AI-driven Automation. International Journal of Multidisciplinary Research and Growth Evaluation. 2025. DOI: https://doi.org/10.54660/.ijmrge.2025.6.1.1942-1943
5. Fawaz A., Mougharbel I., Al-Haddad K., Kanaan H. Y. Energy routing protocols for energy Internet: A review on multi-agent systems, metaheuristics, and artificial intelligence approaches. IEEE Access. 2025. 19 p.
6. Симонов Д. І., Заїка Б. Ю., Симонов Є. Д. Мультивихідні регресійні моделі для управління багатокомпонентними динамічними системами. Таврійський науковий вісник. Серія: Технічні науки, (6). 2024. с. 106–119.
7. Симонов Д. І. Ідентифікація та контроль хаотичних процесів у складних технічних системах. *Науковий вісник Ужгородського університету*. Серія «Математика і інформатика», 46(1). 2025. с. 273–284.
8. Kovacevic J., Radujko U., Djukic M., Novkovic T. Smart Multi-Agent Framework for Automated Audio Testing. Elektronika ir Elektrotechnika. 2023. DOI: https://doi.org/10.5755/j02.eie.33222
9. Nunez A., Islam N. T., Jha S., Najafirad P. AutoSafeCoder: A Multi-Agent Framework for Securing LLM Code Generation through Static Analysis and Fuzz Testing. arXiv.org. 2024. DOI: https://doi.org/10.48550/arXiv.2409.10737
10. Nooyens R., Bardakci T., Beyazit M., Demeyer S. Test Amplification for REST APIs via Single and Multi-Agent LLM Systems. International Conference on Testing Software and Systems. 2025. DOI: https://doi.org/10.48550/arXiv.2504.08113

11. Liu C., Gu Z., Wu G., Zhang Y., Wei J., Xie T. Temac: Multi-Agent Collaboration for Automated Web GUI Testing. arXiv.org. 2025. DOI: https://doi.org/10.48550/arXiv.2506.00520
12. Kong H., Hu D., Ge J., Li L., Li T., Wu B. VulnBot: Autonomous Penetration Testing for A Multi-Agent Collaborative Framework. arXiv.org. 2025. DOI: https://doi.org/10.48550/arXiv.2501.13411
13. Hall V. A. Coding with ChatGPT. Birmingham: Packt Publishing Ltd, 2024. 304 p.
14. Bluck A. S. Practical Java Programming with ChatGPT. Delhi: Orange Education Pvt Limited, 2023. 428 p.
15. Bodungen C. ChatGPT for Cybersecurity Cookbook. Birmingham: Packt Publishing Ltd, 2024. 376 p.
16. Herszfang H. P., Henstock P. V. Supercharged Coding with GenAI. Birmingham: Packt Publishing Ltd, 2025. 460 p.

References:

1. Helmy, M., Sobhy, O., & ElHusseiny, F. (2024). AI-Driven Testing: Unleashing autonomous systems for superior software quality using generative AI. International Telecommunications Conference (ITC-Egypt). https://doi.org/10.1109/ITC-Egypt61547.2024.10620598
2. Kesavan, E. (2024). AI-Driven and autonomous testing. International Scientific *Journal of Engineering and Management*. https://doi.org/10.55041/isjem01651
3. Nama, P. (2024). Integrating AI in testing automation: Enhancing test coverage and predictive analysis for improved software quality. World Journal of Advanced Engineering Technology and Sciences. https://doi.org/10.30574/wjaets.2024.13.1.0486
4. Doddapaneni, J. (2025). AI test design and script generator: Enhancing software testing through AI-driven automation. International Journal of Multidisciplinary Research and Growth Evaluation. https://doi.org/10.54660/.ijmrge.2025.6.1.1942-1943
5. Fawaz, A., Mougharbel, I., Al-Haddad, K., & Kanaan, H. Y. (2025). Energy routing protocols for energy Internet: A review on multi-agent systems, metaheuristics, and artificial intelligence approaches. IEEE Access.
6. Symonov, D. I., Zaika, B. Y., & Symonov, Y. D. (2024). Multi-output regression models for controlling multi-component dynamic systems. *Tavriyskyi Scientific Bulletin*. Series: Technical Sciences, (6), 106–119.
7. Symonov, D. I. (2025). Identification and control of chaotic processes in complex technical systems. *Scientific Bulletin of Uzhhorod University*. Series “Mathematics and Informatics”, 46(1), 273–284.
8. Kovacevic, J., Radujko, U., Djukic, M., & Novkovic, T. (2023). Smart multi-agent framework for automated audio testing. Elektronika ir Elektrotechnika. https://doi.org/10.5755/j02.eie.33222
9. Nunez, A., Islam, N. T., Jha, S., & Najafirad, P. (2024). AutoSafeCoder: A multi-agent framework for securing LLM code generation through static analysis and fuzz testing. arXiv. https://doi.org/10.48550/arXiv.2409.10737
10. Nooyens, R., Bardakci, T., Beyazit, M., & Demeyer, S. (2025). Test amplification for REST APIs via single and multi-agent LLM systems. International Conference on Testing Software and Systems. https://doi.org/10.48550/arXiv.2504.08113
11. Liu, C., Gu, Z., Wu, G., Zhang, Y., Wei, J., & Xie, T. (2025). Temac: Multi-agent collaboration for automated web GUI testing. arXiv. https://doi.org/10.48550/arXiv.2506.00520
12. Kong, H., Hu, D., Ge, J., Li, L., Li, T., & Wu, B. (2025). VulnBot: Autonomous penetration testing for a multi-agent collaborative framework. arXiv. https://doi.org/10.48550/arXiv.2501.13411
13. Hall, V. A. (2024). Coding with ChatGPT. Birmingham, UK: Packt Publishing Ltd.
14. Bluck, A. S. (2023). Practical Java programming with ChatGPT. Delhi, India: Orange Education Pvt Limited.
15. Bodungen, C. (2024). ChatGPT for cybersecurity cookbook. Birmingham, UK: Packt Publishing Ltd.
16. Herszfang, H. P., & Henstock, P. V. (2025). Supercharged coding with GenAI. Birmingham, UK: Packt Publishing Ltd.

Дата надходження статті: 27.10.2025
Дата прийняття статті: 17.11.2025
Опубліковано: 30.12.2025