МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

шифр «**Мета**»

Наукова робота на тему:

МЕТОДИ ІНТЕЛЕКТУАЛЬНОЇ ПІДТРИМКИ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ПРОГРАМНИХ ЗАСОБІВ

2020

ЗМІСТ

стор.

ВСТУП

За останні десятиліття спостерігається бурхливе зростання інформаційних технологій у всіх сферах життя людського суспільства, зокрема в предметній області розробки програмного забезпечення. Зростання обчислювальної потужності, а також розвиток технологій зберігання даних привели до того, що сучасні програмні продукти поєднують величезну кількість функцій і значно збільшилися в розмірах. При цьому, можна відзначити експоненціальне зростання складності всіх класів програмного забезпечення.

Крім зростання складності програмних комплексів потрібно відзначити тенденцію на скорочення тривалості циклу розробки. застосування гнучкою (agile) замість каскадної (waterfall) моделі розробки зажадало підвищення ступеня автоматизації тестування для збереження якості розроблюваного програмного забезпечення [2]. Це призвело до народження нового класу систем - систем автоматизованого тестування [3]. У міру подальшого зростання складності програмних комплексів росте складність систем автоматизації тестування. Внаслідок цього збільшується кількість відмов, викликаних збоями системи автоматизації тестування. В результаті центр докладання зусиль зміщується від розробки програмного продукту і тестів для нього в сторону підтримки системи автоматизованого тестування.

На даний момент підтримки автоматизованого тестування приділяється недостатньо часу в дослідженнях. В кінцевому підсумку, це призводить до підвищених вимог до команди підтримки автоматизованого тестування і неефективного витрачання ресурсів (людино-годин і машинного часу). Застосування системи інтелектуальної підтримки автоматизованого тестування, яка візьме на себе ряд функцій, раніше виконуваних персоналом, дозволить підвищити як якість тестування, так і ефективність використання ресурсів.

Так як створення програмного забезпечення - одна з високотехнологічних спеціалізацій України в міжнародному поділі праці [12, 13], то розробка нового класу систем інтелектуальної підтримки автоматизованого тестування програмних комплексів (СІП АТПК), є подобою САПР, дозволяє отримати перевагу в конкурентній боротьбі на даному сегменті ринку.

Наукова проблема роботи: забезпечення автоматизованого тестування і супровід інструментаріїв підтримки для підвищення ефективності та якості тестування.

Мета і завдання дослідження. Мета дослідження – підвищення ефективності процесів підтримки автоматизованого тестування програмного забезпечення (ПЗ) за рахунок впровадження інтелектуального модуля підтримки роботи системи автоматизації тестування (САТ) програмного забезпечення. Дана мета досягається вирішенням наступних задач:

1. Аналіз стану робіт в області автоматизації процесу тестування ПЗ і виявлення недоліків існуючих підходів до побудови систем автоматизованого тестування.

2. Модифікація типової архітектури систем автоматизації тестування шляхом впровадження модуля інтелектуальної підтримки.

3. Розробка модифікованої продукційної моделі інформаційного забезпечення (ІЗ) інтелектуальної системи для її адаптації до предметної області автоматизованого тестування ПЗ.

4. Вироблення принципів підтримки інтелектуального модуля на протязі усього життєвого циклу даного модуля.

5. Розробка інструментарію для реалізації інтелектуальної підтримки автоматизованого тестування.

Об'єкт дослідження - система автоматизованого тестування програмних комплексів, які виконуються на гетерогенних обчислювальних системах.

Предмет дослідження - інструментарій підтримки процесу автоматизованого тестування програмних комплексів.

Область застосування. Методи і підходи, розроблені в рамках даної роботи, застосовні для модифікації існуючих або розроблюваних систем автоматизованого тестування. Результати аналізу предметної області можуть бути використані на різних етапах проектування систем автоматизації розробки ПЗ для обґрунтування
технічних рішень.

Наукова новизна:

1. Запропоновано модифіковану архітектуру системи автоматизованого тестування ПЗ, призначена для побудови даного класу систем і дозволяє автоматизувати значну частину операцій з підтримки роботи подібних систем і відрізняється від інших наявністю модуля інтелектуальної підтримки.

2. Розроблено модифіковану продукційну модель і спосіб її побудови. Дана модель, а також спосіб її побудови призначені для опису процесів в предметній області функціонування СІП АТПК, дозволяють формалізувати інформаційне забезпечення роботи даного класу АС та відрізняються від аналогів орієнтацією на підтримку всіх етапів ЖЦ даного класу систем.

Практична цінність роботи полягає в наступному:

1. Запропонована модифікована архітектура систем автоматизованого тестування, що відрізняється від існуючих наявністю модуля інтелектуальної підтримки, побудованому з використанням модифікованої продукційної моделі.

2. Сформульовані принципи підтримки продукційної бази знань, орієнтовані на її підтримку на всіх етапах життєвого циклу.

3. Розроблена структура інтелектуального модуля і інтерфейси взаємодії з іншими елементами системи автоматизованого тестування, що дозволило створити інструментарій – основу інтелектуального модуля підтримки автоматизованого тестування.

Основні результати:

1. Модифікована архітектура системи автоматизованого тестування, що відрізняється від існуючих наявністю модуля інтелектуальної підтримки, побудованому з використанням продукційної моделі.

2. Модифікована продукційна модель опису процесів в предметній області функціонування СІП АТПК і спосіб її побудови.

3. Принципи підтримки продукційної бази баз знань в даній предметній області на всіх етапах життєвого циклу.

Апробація роботи.

Структура і обсяг роботи. складається зі вступу, основної частини, що містить 4 розділи, висновків і списку використаних джерел. Загальний обсяг роботи - 84 сторінки. Робота містить 20 рисунків та 6 таблиць. Список використаної літератури включає 47 бібліографічних джерела.

1 ДОСЛІДЖЕННЯ ТА АНАЛІЗ ТЕНДЕНЦІЙ РОЗВИТКУ СИСТЕМ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ

Розвиток індустрії розробки програмного забезпечення відбувалося спільно з розвитком підходів до тестування ПЗ. При цьому розвивалися як методи тестування, так і погляди на тестування в цілому.

Одним з основних факторів, що визначають результат діяльності є цілепокладання. За час існування тестування відбулося кілька змін у визначенні цілей тестування, що значно вплинуло на розвиток даної галузі:

1. На початковому етапі метою тестування був доказ працездатності програмного забезпечення. Тестуванням займався розробник і тестування було лише частиною процесу налагодження. Процес тестування будувався як демонстрація того, що продукт коректно виконує закладені функції. Недоліком такого підходу є те, що в реальному житті оточення продукту не ідеально (можливе введення помилкових даних, необхідні ресурси можуть виявитися недоступними, характеристики оточення можуть не відповідати очікуванням). Однак, поведінка товару не перевірялося для подібних випадків. Як наслідок, працездатність програмного комплексу не гарантувалася в нештатних ситуаціях.

2. У міру розвитку індустрії розробки програмного забезпечення та відокремлення ролей: розробник, інженер по тестуванню, відбулися зміни в цілях тестування. На даному етапі на перший план виходить пошук вразливостей, при якому за мету ставиться пошук усіх можливих помилок в програмному забезпеченні. Даний підхід несе в собі ряд суперечностей:

- процес тестування при такому підході нескінченний, так як виправлення однієї помилки може породити нові. Не існує критерію припинення тестування;

- кількість відомих помилок в продукті мало говорить про стан продукту. Відсутність відомих помилок може говорити, як про високу якість продукту, так і про низьку якість тестування;

- протиставлення розробника програмного забезпечення, метою якого була мінімізація помилок в програмі, і інженера з тестування, орієнтованого на пошук помилок, нерідко призводило до конфліктів між відділами розробки та тестування. Конфлікти негативно позначалися на загальному процесі розробки та на якість фінального продукту;

- справжньою метою розробки є отримання якісного продукту, а не пошук недоліків в ньому. Дотримання цього підходу суперечить спільній меті процесу розробки програмного забезпечення.

Рішення даних протиріч зумовило перехід на третій етап. На сучасному етапі розвитку тестування програмного забезпечення метою є забезпечення якості продукту. Це багатогранний процес, керуючий якістю продукту, який має на увазі формалізацію всіх етапів розробки, тестування, випуску і експлуатації.

Зміна цілей тестування міняли методи тестування і підходи до тестування в цілому. На початкових етапах метою було доказ працездатності програми. При цьому логічним є бажання досягти повного покриття коду, так як в такому випадку досягалася гарантія працездатності. Але в міру зростання складності програмного забезпечення почали проявлятися недоліки такого підходу:

1. Кількість ресурсів, необхідних для збільшення покриття на 1% росте експоненціально в міру наближення до 100%. І досягнення 100% покриття неможливо при розумних витратах.

2. Не тільки досягнення повного покриття, але і його вимір, для складного програмного забезпечення є нетривіальним завданням:

- спрощені форми (покриття по функціям або рядках коду) не дають адекватної оцінки, так як не враховують всі можливі шляхи виконання;

- покриття по шляхах виконання або покриття за умовами для складних програмних комплексів вимагають покриття величезної кількості випадків.

Частину з них неможливо покрити в принципі через особливості оточення. При цьому визначення загальної кількості та виділення значущих саме по собі є нетривіальним завданням. Написання тестів для кожного з шляхів виконання або умов настільки трудомістке завдання, що ніколи не робилося подібних спроб для складних програмних комплексів, навіть в таких критичних областях як програмне забезпечення для космосу або атомних об'єктів, найчастіше при оцінці покриття враховуються очікувані ситуації. При цьому в міру зростання гетерогенності оточення і швидкості його зміни неможливо передбачити всі можливі ситуації. В результаті досягнення повного покриття в мінливому середовищі неможливо в принципі.

Таким чином відбувся відхід від максимізації покриття як засобу досягнення заданої якості продукту. Покриття тестами стає одним з параметрів при оцінці поточної якості продукту. Проаналізувавши тенденції в організації тестування, можна виділити найбільш значущі:

1. Тестування програмного забезпечення розглядається як одним із заходів забезпечення заданої якості продукту.

2. Зростання складності розроблюваних продуктів призвів до зростання кількості тестів необхідних для перевірки якості продукту і зростання ступеня автоматизації процесу тестування програмного забезпечення.

3. Загальна тенденція до скорочення часу циклу розробки підвищує вимоги до надійності систем автоматизованого тестування.

Більшість з сформульованих вимог відтворюється в кожній з відомих авторам САТПК. Це дозволяє формалізувати типову архітектуру даного класу АС, яка представлена на рис. 1.

На рис. 1. представлена загальна архітектура системи автоматизованого тестування, що складається:

1. Парку тестових машин, під яким слід розуміти всі пристрої, використовувані для тестування програмного забезпечення. це можуть бути

серверні платформи, настільні комп'ютери, мобільні системи, планшети, смартфони. Крім того, можуть використовуватися системи віртуалізації. Наприклад, Android емулятори або віртуальні машини VMWare. Парк тестових машин може перебувати в одному місці, так і бути географічно розподіленим.

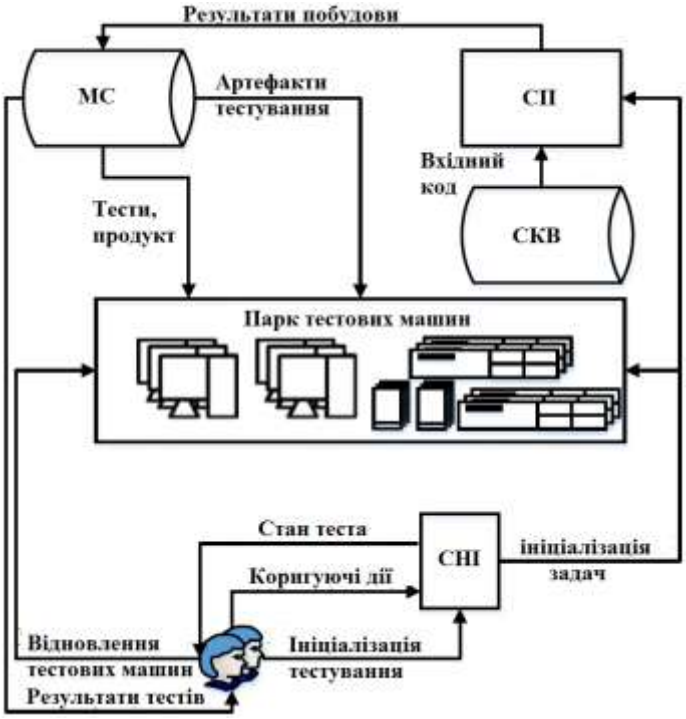

Рисунок 1 - Типова архітектура систем автоматизованого тестування

2. Системи розподілу завдань, в якості якої використовується одна з систем безперервної інтеграції. Крім розподілу тестових завдань вона дозволяє централізовано адмініструвати тестові машини. Дані підсистеми побудовані на клієнт серверній архітектурі. При цьому є один сервер, і на кожній тестовій машині працює клієнтський додаток, який обробляє запити від сервера і здійснює запуск завдань на конкретній тестовій машині.

3. Системи побудови (СП), що складаються з декількох серверів, для побудови тестового програмного забезпечення і тестів для всіх цільових платформ.

4. Системи контролю версій вихідного коду (СКВ), містить вхідний код програмних засобів, що використовуються для тестування, в тому числі: тести, розроблені для тестування продукту; система емуляції оточення -

дозволяє підготувати тестову машину для виконання тестів. Зазвичай це включає в себе установку або емуляцію установки: програм, від яких залежить тестоване програмне забезпечення, програм, від яких залежать тести, самого тестованого програмного забезпечення; програми, що забезпечують тестування.

5. Мережеве сховище (МС) служить для зберігання програм, необхідних для забезпечення тестування, скомпільовані версії тестів та тестового програмного забезпечення, а також артефактів тестування.

6. Інженерів по тестуванню і системних адміністраторів, які забезпечують роботу системи.

При цьому важливо розуміти життєвий цикл тестового завдання в рамках даної системи автоматизованого тестування. В ідеальному випадку кожен тест проходить ряд етапів послідовно, як показано на рис. 2.

Рисунок 2 - Ідеальний життєвий цикл тестового завдання

Але як показує практика, на кожному з етапів можливі відмови, які призводять до необхідності повернення на попередні етапи і повторення ряду кроків. В результаті життєвий цикл набуває вигляду, вказаний на рис. 3.

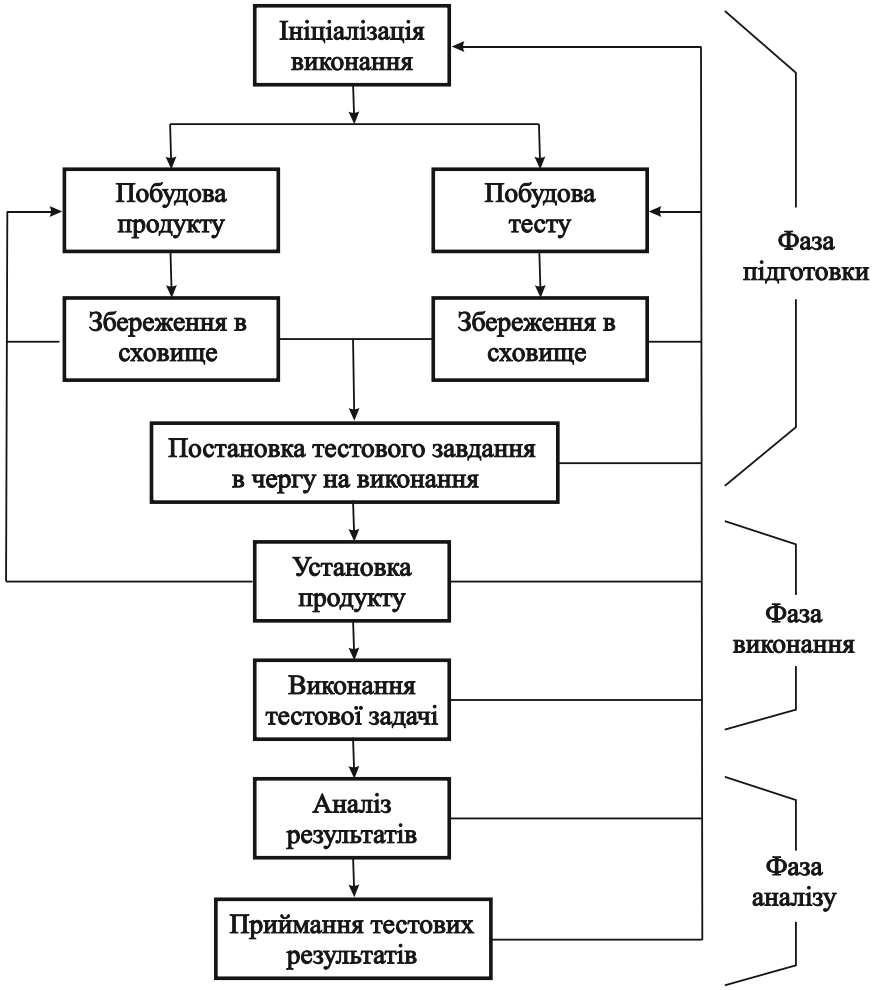

Рисунок 3 - Реальний життєвий цикл тестового завдання

Контроль процесу автоматизованого тестування і управління ним здійснюється інженерами по тестуванню. З урахуванням ітеративності життєвого циклу одиничного тесту, великої кількості тестів, а також обмеженого часу, який відводиться на тестування, інженери по тестуванню змушені витрачати чимало часу на процес підтримки автоматизованого тестування. При цьому вони виконують наступні дії:

1. Контроль за станом парку тестових пристроїв, що включає в себе регулярний контроль їх стану і відновлення при необхідності їх працездатності.

2. Первинний аналіз звітів про тестування і сортування помилок тестування відповідно до причини відмови: проблеми оточення (відмови програмного і апаратного забезпечення, не пов'язаного з тестами або тестуючим продуктом); відмови, пов'язані з помилками в тесті; відмови, пов'язані з помилками в тестованому продукті.

З урахуванням безперервності процесу тестування, необхідна організація позмінної роботи, щоб уникнути втрати машинного часу із-за різних проблем з оточенням. Крім того, аналіз зареєстрованих відмов на прикладі одного з проектів показав, що близько 20% помилок, що виникають під час автоматизованого тестування, є помилковими відмовами і вимагають виправлення тестів / оточення або перезапуску тестів. З урахуванням кількості тестів, виконуваних в рамках тестування, трудомісткість підтримки автоматизованого тестування стає значною. Альтернативою цьому може стати передача частини функцій від персоналу до самої системи автоматизації. На даний момент всі однозначно визначені ланцюжки дій автоматизовані. Подальша автоматизація вимагає впровадження модуля прийняття рішення для адекватного аналізу результатів і введення корекційних дії в автоматичному режимі.

Подальше підвищення ступеня автоматизації процесу тестування програмних комплексів неможливо без інтелектуалізації системи автоматизованого тестування. До інтелектуальних систем (ІС) відносяться технічні або програмні системи, в функціонал яких входить вирішення завдань для конкретної предметної області, для яких заздалегідь невідомий алгоритм рішення [18]. У структурі ІС виділяють інтелектуальний інтерфейс, машину логічного висновку і базу знань. База знань є основою ІС, визначає її приналежність до конкретної предметної області та можливість вирішувати поставлені перед ІС завдання.

Підвищення ступеня автоматизації простежується і в системах САПР [14,27]. Подібні системи зароджувалися як системи що розширюють аналітичні

можливості людини. У міру зростання обчислювальних потужностей і ступеня автоматизації, в окремих галузях, стало можливим автоматичне застосування рішення, виробленого САПР. Зокрема, в предметній області автоматизованого тестування можливо діагностування до 92% відмов у автоматичному режимі. При цьому для 80% з них можуть бути автоматично застосовані коригувальні дії. При цьому пропонована система є подальшим розвитком класичних САПР в предметній області автоматизованого тестування.

В даний час існує досить велика кількість програмних оболонок побудови продукційних експертних систем (CLIPS, DROOLS, Jess). Таким чином немає необхідності реалізовувати процедуру логічного висновку, натомість можна використовувати існуюче середовище розробки. Більшість цих середовищ використовує RETE алгоритм. Пропоновані модифікації продукційного правила не впливають на працездатність RETE алгоритму, так як передбачувані модифікації вводяться в вигляді поля факту і обробляються як додаткові умови.

Таким чином, може бути використана будь-яка з існуючих програм-оболонок, побудованих на базі даного алгоритму. Як середовище розробки можуть бути використані CLIPS, DROOLS або Jess.

2 ПОСТАНОВКА ЗАДАЧІ

В результаті дослідження було виявлено протиріччя, виражене в тому, що складність розроблюваних програмних комплексів зростає, в той час як тривалість циклу розробки скорочується. Вивчення типової архітектури систем автоматизації тестування підкреслило, що існуючі підходи потребують значних трудовитрат для досягнення необхідної якості тестованого продукту. Це обумовлює актуальність і перспективність даного дослідження.

Порівняння властивостей різних моделей представлення даних, а також параметрів предметної області автоматизованого тестування дозволило вибрати продукційну модель, як найбільш ефективну для побудови модуля інтелектуальної підтримки автоматизованого тестування.

Аналіз властивостей продукційних систем дозволив виявити ряд проблем, які потребують вирішення, для ефективного використання продукційної бази знань в досліджуваній предметній області: формальні конфлікти; вибір альтернативних рішень; складність представлення великої кількості параметрів об'єктів реального світу в продукційних системах; складності масштабування продукційних баз знань.

Сформульовано науково-практичні завдання дослідження, найважливішими з яких є: модифікація продукційної моделі інтелектуальної системи до досліджуваної предметної області; вироблення принципів підтримки інтелектуально модуля на всіх етапах його життєвого циклу; розробка інструментаріїв для реалізації запропонованого підходу.

3 ПРОДУКЦІЙНА МОДЕЛЬ ОПИСУ ПРОЦЕСІВ ПРЕДМЕТНОЇ ОБЛАСТІ ФУНКЦІОНУВАННЯ СІП АТПК

Типова архітектура систем автоматизованого тестування вимагає значної участі з боку інженерів по тестування і адміністраторів. Функція підтримки цілком і повністю покладається на людину. Даний підхід має ряд недоліків:

1. Для якісного аналізу стану тестів необхідно аналізувати великий обсяг даних (мільйони тестових завдань і десятки мегабайт файлів звітів). Людина погано пристосована для подібної роботи.

2. Контроль за станом системи потрібно проводити якомога частіше, щоб максимально швидко виявляти проблему і виправляти її до того, як вона вплине на велику кількість тестових завдань. З урахуванням великого обсягу даних і цілодобового режиму тестування, необхідно організовувати позмінну роботу великої кількості інженерів.

3. Рутинний характер роботи важкий для людини і демотивує інженерів, які беруть участь в підтримці процесу тестування.

4. В характері роботи закладено протиріччя: виявлення проблем не вимагає високої кваліфікації; виправлення виявлених проблем, вимагає широких знань і високої кваліфікації.

У зв'язку з цим персонал, який здійснює підтримку, розбитий на дві команди: виявлення проблем і спроба вирішення по шаблонах покладається на команду безпосередньої підтримки, що складається з менш кваліфікованих фахівців і працюють позмінно; рішення проблем, не вирішених на першому етапі, вирішують більш кваліфіковані фахівці.

В результаті потрібна велика команда різних фахівців для підтримки системи. Зважаючи на характер роботи існує велика текучка і як наслідок, високі витрати на навчання. При цьому характер робіт по підтримці процесу тестування ідеально підходить для автоматизації: обробка великих обсягів даних, велика

кількість повторень, безперервна цілодобова робота, наявність готових сценаріїв обробки помилок.

В рамках даної роботи пропонується впровадити інтелектуальну систему, що виконує функції команди безпосередньої підтримки, для підвищення ефективності роботи САТПК. В результаті відбудеться перехід до використання системи, зазначеної на рис. 4, в якій інтелектуальний модуль автоматизує ряд рутинних операцій, які раніше вимагали участі персоналу.

Модифікована система автоматизованого тестування (рис. 4) складається:

1. Парку тестових машин, під яким слід розуміти всі пристрої, використовувані для тестування програмного забезпечення. Це можуть бути серверні платформи, настільні комп'ютери, мобільні системи, планшети, смартфони.

2. Системи розподілу завдань, в якості якої використовується одна з систем безперервної інтеграції (СБІ).

3. Системи побудови (СП), що складається з декількох серверів, для побудови тестованого програмного забезпечення і тестів для всіх цільових платформ.

4. Системи контролю версій вихідного коду (СКВ), яка містить вхідний код програмних засобів, що використовуються для тестування, в тому числі: тести, розроблені для тестування продукту; система емуляції оточення - дозволяє підготувати тестову машину для виконання тестів.

5. Мережеве сховище (МС), яке служить для зберігання програм, необхідних для забезпечення тестування, скомпільовані версії тестів та тестового програмного забезпечення, а також артефактів тестування.

6. Тестових інженерів і системних адміністраторів, які забезпечують роботу системи.

7. Модуля системи інтелектуальної підтримки (СІП), який бере на себе частину функцій, раніше виконуваних персоналом.

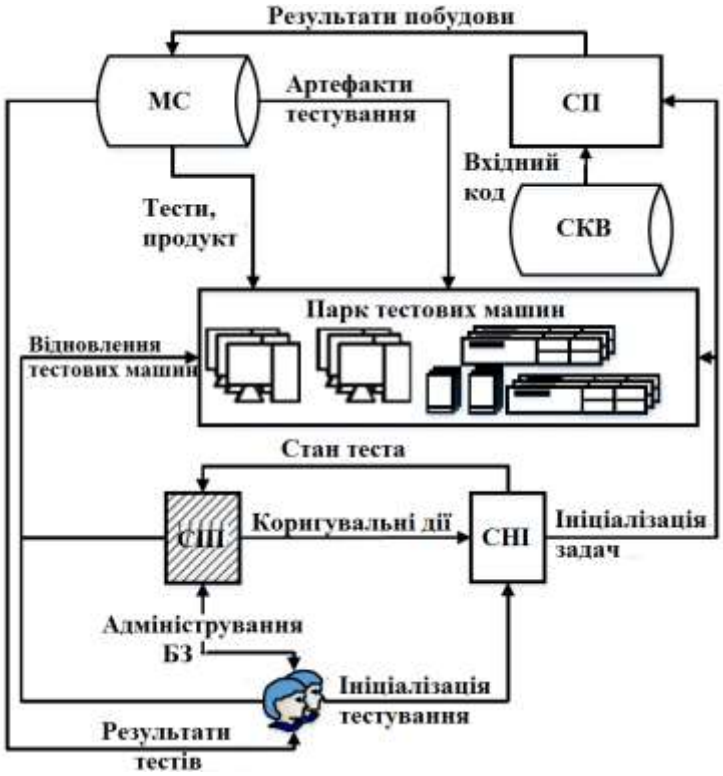

Рисунок 4 - Модифікована архітектура систем автоматизованого тестування

Нова архітектура, за рахунок автоматизації прийняття рішень, дозволяє автоматизувати які раніше не були автоматизовані операції і знизити навантаження на персонал, що дозволить інженерам з тестування сфокусуватися на покращенні якості тестування за рахунок поліпшення самих тестів і збільшення тестового покриття [21].

Продукційна модель опису процесів предметної області функціонування СІП АТПК. Продукційна модель бази знань є гнучким інструментом, що дозволяє ефективно будувати експертні системи для різних областей. Особливості цієї моделі, також як і особливості предметної області, вимагають модифікація загальної моделі для заданої предметної області.

Базовий формат правил продукційної моделі має такий вигляд.

$$(k)\ ЯКЩО\ A_i\ і\ A_j\ ТО\ B_1 \qquad (1)$$

де: k - ідентифікатор правила, A - множина допустимих умов, B - множина допустимих дій.

У процесі модифікації продукційної моделі до предметної області автоматизованого тестування була проаналізована статистика виконання тестів в системі автоматизованого тестування. В результаті були виділені:

- множина об'єктів в предметної області (дана множина відображається в базі знань через стан об'єктів) – збірка продукту, збірка тестів, тестові пристрої, тестова задача і т. д .;

- множина можливих станів об'єктів предметної області відображається на множині умов (*A*) - для тестового завдання це: не запущений, в черзі на виконання, виконується, завершив виконання;

- множина операцій, що призводять об'єкти в певний стан, відображається на множині допустимих дій - для тестової завдання це: перезапуск, перезапуск на іншому тестовому пристрої, відправка повідомлення розробнику тесту і т.д.

Поряд з перевагами, у продукційних систем є ряд недоліків. До недоліків продукційної моделі можна віднести наявність конфліктів двох типів [39,40]:

1. Формальні конфлікти, які виникають, коли в процесі життєвого циклу в базі знань виникають кілька правил з однаковими лівими частинами, але різними правими.

2. Конфлікти, що виникають при інтеграції проектів високого рівня складності, коли існує кілька різних підходів до вирішення однієї і тієї ж задачі.

Обидва типи конфліктів вносять неоднозначність в процес прийняття рішення. У разі конфліктів першого типу можна ввести в розгляд додатковий параметр об'єкта, тим самим диференціюючи два стани і правила. Але в реальних системах не завжди можна отримати достатню деталізацію стану об'єкта. В результаті одні й ті ж «симптоми» можуть описувати різні «реальні» стани об'єкта і вимагати різні дії. Тому неможливо гарантувати відсутність формальних конфліктів в базі знань. Незважаючи на те що виникнення конфліктів альтернативних рішень малоймовірно, тому що в даному випадку предметна область досить обмежена, при цьому за рахунок розбиття всієї предметної області

на фрагменти, відповідні типові завдання, досягається зменшення ймовірності виникнення подібних конфліктів, але виключити їх виникнення не можна.

Таким чином, наявність конфліктів обох видів є невід'ємним властивістю продукційної моделі баз знань. А так як конфлікти можуть виникати протягом усього життєвого циклу виробничої системи, що обумовлено тим, що очікується поява нових знань про предметну область в процесі роботи СІП, то потрібно формалізувати й автоматизувати засоби вирішення подібних ситуацій.

В обох випадках виникає більше одного правила з однаковими лівими частинами. Необхідно розробити механізм пріоритезації подібних правил і виконання одного з них. Можуть бути різні підходи до порядку перебору:

- застосовувати випадкове правило з використанням генератора псевдовипадкових чисел - найбільш простий спосіб з точки зору реалізації, але найменш ефективний, так як не передбачуваний і не дозволяє будувати оптимальні сценарії;

- застосовувати останнє правило дозволило досягти результату - такий підхід добре працює для двох правил і показує незадовільні результати з точки зору оптимальності сценарію при більшій кількості альтернативних правил;

- застосовувати правила відповідно до наперед визначеним пріоритетом - добре працює в статичному оточенні, але швидко змінюється середовище вимагає постійної зміни пріоритетів для збереження оптимальності сценаріїв;

- застосовувати правила відповідно до частоти їх успішного спрацьовування за попередній період - найбільш гнучка і добре формалізується система пріоритетів.

У магістерській роботі пропонується зменшити навантаження на експертну систему за рахунок скорочення числа застосовуваних правил [20,21]. Для цього вводиться два рівня пріоритетів. Експертний коефіцієнт $\left(M_f\right)$ - задається адміністратором БЗ, на етапі наповнення бази знань на основі попереднього досвіду використання САТПК. Так як кількість правил в кожному фрагменті бази

знань не перевищує тридцяти, адміністратор бази знань може поставити пріоритети вручну.

$$M_f = X \tag{2}$$

Даний коефіцієнт використовується, коли кілька правил мають однаковий пріоритет. Це особливо актуально на початковому етапі роботи СІП, коли мало даних для використання статистичних критеріїв. Але і в процесі життєвого циклу подібних систем можуть виникати ситуації, коли кілька правил мають однакову частоту застосування.

Крім того, пропонується ввести статистичний коефіцієнт, обчислюється як частота активації правила що призвела до успішного результату.

$$M_h = C / T \tag{3}$$

де: C - кількість виконань правила за минулий період, що приводило до успішного результату; T - тривалість періоду підрахунку частоти.

Вибір оптимальної тривалості періоду важливо, так як в змінному середовищі найбільш часто зустрічаються проблеми що також змінюються. На підставі аналізу статистики тривалість періоду обрана 7 днів.

При цьому потрібно скоротити число кроків для досягнення бажаного результату. Це означає що правило, яке миттєво призводить до бажаного результату має викликатися в першу чергу. Правило, яке вимагає додаткових перевірок для досягнення результату менш пріоритетно. Для автоматичної оцінки правил вводиться додатковий коефіцієнт, що описує кількість правил, викликаних після поточного до досягнення результату. В результаті даний пріоритет набуває вигляду:

$$M_h = F / T \cdot \sum 1 / N \tag{4}$$

де: N - кількість правил, активованих після поточного до досягнення мети за попередній період.

Кількість пріоритетів може бути розширено для більш ефективного управління БЗ. В результаті загальний формат продукційного правила набуде вигляду:

$$(k)\ ЯКЩО\ \left(A_i \ldots A_j\right)\ i\ \left(M_f \ldots M_h\right)\ ТО\ B_1 \qquad (5)$$

де: k - ідентифікатор правила; A - множина допустимих умов; B - множина допустимих дій; M - множина допустимих пріоритетів, які застосовуються до правил.

З метою виключення конфліктів при визначенні пріоритету функції пріоритетів ранжуються адміністратором БЗ. В даному випадку M_h має більш високий пріоритет, в порівнянні з M_f.

Предметна область системи автоматизованого тестування є достатньо складною. Велика кількість параметрів і станів може привести до складнощів в побудові і підтримці сценаріїв у базі знань. Можливі два шляхи збору даних для роботи сценарію:

1. Збір всіх даних, які можуть знадобитися при роботі сценарію. Первинний збір даних перед стартом сценарію може зайняти значний час, за цей час, або буде простоювати тестовий пристрій, або можуть виникнути відмови наступних тестових завдань. В обох випадках відбувається значна втрата машинного часу, а також часу персоналу, що витрачається на перезапуск тестових задач. При цьому велика частина зібраних даних не буде використана, так як в оптимальний сценаріїй обходиться мінімальною частина гілок дерева сценарію.

2. Збір даних по мірі обходу дерева сценарію, породжує ряд протиріч: з одного боку, збір даних - повільна операція через можливих перерв і очікування доступності ресурсів, це призводить до необхідності паралельного застосування сценаріїв; але розпаралелювання застосування сценаріїв з урахуванням тривалості отримання даних для накопичення фактів для

кожного вузла сценарію призводить до того, що за час роботи сценарію змінюється стани досліджуваних об'єктів за рахунок роботи інших сценаріїв.

В результаті, для оптимальної роботи інтелектуального модуля потрібно скоротити час підготовки бази фактів за рахунок фрагментації предметної області, при цьому виникає два рівня сценаріїв:

- мета-сценарій, який вибирає, до якого фрагменту предметної області відноситься проблема на підставі файлів звіту. При граничному стані можуть бути ініційовані сценарії для декількох предметних областей.

- сценарії для фрагментів предметної області відповідно до типових завдань (наприклад, проблема програмного оточення або мережева проблема), який збирає необхідний обсяг даних і формує базу фактів перед запуском сценарію.

Даний підхід дозволяє: скоротити число правил, аналізованих в процесі роботи в межах одного сценарію; спростити підтримку бази правил, так як при модифікації сценарію необхідно враховувати менший обсяг правил; розподілити підтримку різних сценаріїв між різними адміністраторами бази знань.

В результаті, шаблон продукційного правила набуде вигляду:

$$(frag)\ (k)\ ЯКЩО\ \left(A_i \ldots A_j\right)\ i\ \left(M_f \ldots M_h\right)\ ТО\ B_1 \qquad (6)$$

де: *frag* - ідентифікатор фрагмента предметної області, до якої належить дане правило.

Ще одним недоліком продукційної моделі бази знань є складність застосування в реальних проектах через занадто складну структури реальних об'єктів. В результаті для роботи правил потрібно кілька параметрів, які необхідно отримати до моменту активації правила. Для вирішення цієї проблеми пропонується використання апарату дескрипторів, що означає концептуальний перехід від продукційної моделі до фреймо-продукційної або ситуаційної моделі [18,26]. Дескриптор містить посилання на паспорт ситуації, який надає інформаційне забезпечення процедур подій. У ньому виділяються наступні цілі

(межі представлення ситуації): ідентифікатор факту - ID; множина вхідних параметрів - *A*; множина вихідних параметрів - *B*.

Наприклад, при перевірці мережевої доступності тестової машини паспорт ситуації прийме вигляд, показаний на рис. 5.

В результаті узагальнене правило набуде вигляду:

$$(frag)\ (k)\ ЯКЩО\ \left(D.A_i \ldots D.A_j\right)\ i\ \left(D.M_f \ldots D.M_h\right)\ TO\ D.B_1 \qquad (7)$$

де: *frag* - ідентифікатор фрагмента предметної області, до якої належить дане правило; *k* - ідентифікатор правила; *D* – множина дескрипторів ситуацій; *M* - множина допустимих пріоритетів, застосовуваних до правил; *A* - множина допустимих умов; *B* - множина допустимих дій.

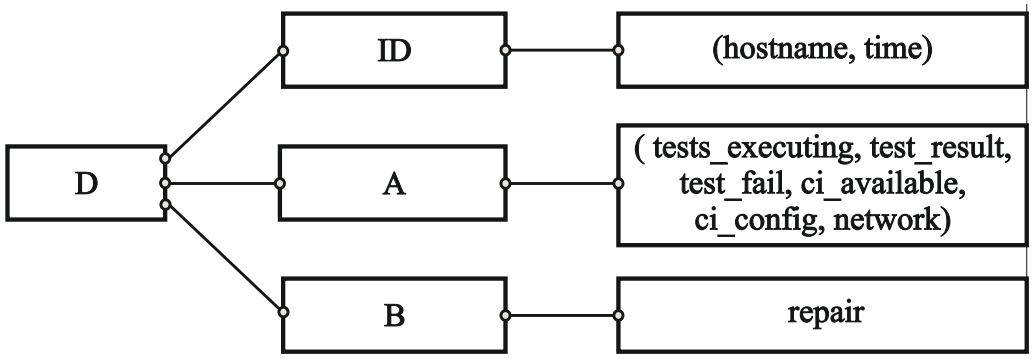

Рисунок 5 - Модель паспорта ситуації

Таким чином, запропонована модифікована продукційна модель опису процесів в предметній області функціонування СІП АТПК і спосіб її побудови, які дозволяють формалізувати інформаційне забезпечення роботи даного класу АС базою знань і відрізняється від аналогів орієнтацією на підтримку всіх етапів ЖЦ даного класу систем.

Принципи побудови та підтримки бази знань.

Розробка і впровадження системи є необхідною, але не є достатньою умовою успішного її застосування. Необхідно враховувати весь життєвий цикл комплексу та розробити принципи його підтримки на всіх етапах ЖЦ. У процесі життєвого циклу САТПК змінюється оточення, що призводить до зміни в предметній області СІП. Наприклад, введення нових апаратних платформ в якості

тестових комп'ютерів призводить до росту помилок, викликаних апаратними збоями і, як наслідок, до необхідності змінити сценарії.

Крім того, можуть виникнути раніше невідомі помилки або зникнути старі ситуації, що вимагають обробки системою інтелектуальної підтримки. В результаті, додадуться або видаляться відповідні вузли в деяких сценаріях. Будь-яка зміна сценарію несе ризик для його цілісності і ефективності.

У роботі сформульовані принципи підтримки системи бази знань, які дозволяють вирішувати проблеми надмірності і зв'язності сценаріїв на всіх етапах життєвого циклу, і відрізняються від інших орієнтованістю на запобігання виникнення дефектних або неефективних сценаріїв[20,21]:

1. Принцип унікальності результатів для всіх вузлів в дереві сценарію, допускається одноразове використання кожного допустимого результату в рамках одного сценарію, дозволяє гарантувати відсутність дублюючих шляхів / циклів в сценарії.

$$\left\{ s_{вих.} \mid s_{i\,вих.} = s_{j\,вих.} \;\; i \;\; i \neq j \right\} = \varnothing \tag{8}$$

2. Принцип логістичної зв'язності, що полягає в тому, що кожен вузол в сценарії повинен бути доступний з кореня, дерева сценарію. Це дозволяє гарантувати цілісність БЗ.

$$S = \left\{ s \mid зв'язаний\ з\ коренем\ (s) \right\} \tag{9}$$

3. Принцип логістичної оптимальності вимагає скорочувати шлях пройдений в сценарії до досягнення бажаного результату.

$$P_{лог.опт.} = \min\left(\sum path(F) \right) \tag{10}$$

4. Принцип статистичної оптимальності передбачає, що вузли в сценарії, які частіше приводили до бажаного результату, з більшою ймовірністю призведуть до нього і в майбутньому.

$$P_{стат.опт.} = \min\left(path\left(S_{\max(N_S)} \right) \right) \tag{11}$$

Модифікація БЗ за рахунок додавання / видалення / зміни правил може привести до наступних проблем сценаріїв:

1. Виникнення дублюючих шляхів, які призводять до неоднозначності в сценаріях аналізу і, як наслідок, до складнощів підтримки і аналізу фінальних сценаріїв.

2. Незв'язаність дерев - недоступні вузли та гілки можуть виникнути при зміні існуючих правил, коли вузол або ціла гілка дерева сценарію не може бути досягнута через втрату зв'язку з коренем дерева.

3. Виникнення циклічних шляхів, які можуть привести до зациклення процедури аналізу і блокування експертної підсистеми.

Принцип логістичної оптимальності говорить, що оптимальність сценарію можна визначити в залежності від кількості вузлів, пройдених в сценарії, до досягнення бажаного стану. При цьому найбільш «ефективні» вузли повинні проходитися в першу чергу.

Вузол вважається ефективним якщо в результаті проходу через нього був досягнутий бажаний стан. При цьому «ефективність» вузла вище, якщо до досягнення бажаного результату було пройдено меншу кількість кроків.

В результаті, для скорочення кількості кроків, необхідних для однозначного визначення стану досліджуваного об'єкта пропонується використовувати принцип частотного пріоритету. При цьому обчислюється частота проходу через вузол на основі даних про попередні запуски тестування. Розрахунок розбивається на три етапи:

1. Кожному листу в дереві аналізу, задається пріоритет, який вираховується як кількість успішних активацій даного правила до загальної кількості активацій, на основі даних про попередні запуски тестування.

2. Будується таблиця сумісності подій, результатів і станів.

3. Кожній події, що має один або кілька результатів (проміжна інформація про стан об'єкта), ймовірність задається як сума ймовірностей піддерева.

Дерево будується, починаючи з найбільш ймовірних подій і далі до найменш вірогідних подій. Таким чином максимізують ймовірність виявлення помилки за мінімальну кількість кроків. За умови рівності ймовірностей, пріоритет віддається стану, що описує апаратні проблеми.

За час життя САТПК може знадобитися зміна сценарію в зв'язках з:

1. Зміною «ефективності» вузлів подій (в результаті змінами інфраструктури або парку тестових машин, статистика відмов може змінюватися).

2. Виділенням нової інформації про предметну область (виявлення нових джерел відмов).

3. Появою нових об'єктів контролю (введення нових підсистем).

4. Ситуаціями, коли проблема не відтворюється протягом тривалого періоду часу (виправили причину виникнення відмов).

Модифікація БЗ за рахунок додавання / видалення / зміни правил (в зв'язку з виділенням нової інформації про предметну область) може привести до різних проблем сценаріїв:

1. Виникнення дублюючих шляхів, які призводять до неоднозначності в сценаріях аналізу і як наслідок до складнощів підтримки і аналізу фінальних сценаріїв.

2. Незв'язані дерева - недоступні вузли та гілки можуть виникнути при зміні існуючих правил, коли вузол або ціла гілка дерева сценарію не може бути досягнута через втрату зв'язку з коренем дерева.

3. Виникнення циклічних шляхів, які можуть привести до зациклення процедури аналізу і блокування експертної підсистеми.

В результаті потрібні процедури верифікації та корекції БЗ. На першому етапі автоматизуються процедури верифікації. Процес корекції виконується адміністратором БЗ.

Для виключення виникнення проблем першого типу пропонується дотримуватися принципу унікальності результатів, а саме, контролювати кількість

однакових результатів в дереві сценарію. Всі результати повинні бути унікальні в рамках розглянутого набору правил, що описують фрагмент предметної області. Тим самим вирішується і третя проблема, так як цикл передбачає, що в один вузол можна прийти з двох вихідних вузлів.

Як верифікатор проблем другого типу використовується принцип логістичної зв'язності. В рамках даного підходу перевіряються всі можливі шляхи з кожного вузла. При цьому проводиться процедура зворотного виведення. Перевірка вважається успішно пройденою, якщо вдалося досягти корінь дерева.

Крім перевірки правил коректності на рівні правил, потрібно переконатися в доступності даних, необхідних для кожного правила. Це особливо актуально в разі процедури додавання нових правил. Так як дані доступні через дескриптор в паспорті ситуації досить перевірити, що в паспорті ситуації присутнє відповідне поле. Крім того, в процесі зміни правил може виникнути ситуація, коли поле в паспорті ситуації не використовується жодним правилом в БЗ. Такі поля потрібно видаляти з метою скорочення розміру факту.

Можна відзначити орієнтацію принципів на підтримку системи інтелектуальної підтримки автоматизованого тестування на всіх етапах життєвого циклу.

На етапі проектування в архітектуру системи закладаються елементи, що забезпечують реалізацію запропонованих принципів (система пріоритетів, засіб роботи зі знаннями, інструментарій модифікації системи). На етапі розробки відбувається не тільки побудова інструментарію, а й первинне заповнення бази знань. На цьому етапі адміністратори бази знань користуються принципами статистичної оптимальності, логістичної зв'язності і принципом унікальності результатів для формування коректних і оптимальних сценаріїв, з урахуванням попереднього досвіду. На етапі верифікації бази знань інтелектуальний модулю працює в дорадчому режимі: відбувається аналіз даних зібраних за попередній період і інформація про коректуючі дії надається

адміністратору БЗ. Адміністратор порівнює коригувальну дія, вироблену в автоматичному режимі, і то що було зроблено щоб виправити ситуацію, перевіряючи роботу СІП. При цьому відбувається накопичення даних для частотного пріоритету і підвищення ефективності сценарію. На цьому і наступних етапах життєвого циклу СІП САТПК може знадобитися модифікація сценаріїв (виявлена помилка, змінилися зовнішні умови, що призвели до появи нового або виключенню старого типу проблем). Принцип унікальності результатів і логістичний принцип дозволяють гарантувати коректність сценаріїв при їх модифікації. Фрагментація бази знань полегшує аналіз сценаріїв і прискорює їх обробку. Система пріоритетів дозволяє підвищити ефективність сценаріїв з точки зору витрачених обчислювальних ресурсів.

На етапі виведення з експлуатації (наприклад, при переході на використання іншої системи автоматизованого тестування) можливе створення генератора, що перетворює базу знань в людино - орієнтовані документи, що дозволить зберегти накопичені знання про предметну область для подальшого використання.

У процесі вивчення предметної області були виявлені ряд закономірностей і особливостей які обумовлюють необхідність впровадження інтелектуального модуля підтримки систем автоматизованого тестування. Для реалізації подібного модуля пропонується:

1. Модифікувати типову архітектуру системи автоматизованого тестування додаванням інтелектуального модуля підтримки.

2. Застосувати модифікований формат продукційного правила для більш ефективної реалізації сценаріїв в заданій предметній області та нівелювання недоліків властивих продукційним системам.

3. Використовувати розроблені принципи підтримки бази знань інтелектуального модуля на віх етапах життєвого циклу, зокрема: логістичний критерій, як засіб для перевірки зв'язності сценаріїв; принцип унікальності

результатів, що гарантує коректність сценаріїв на всіх етапах життєвого циклу; логістичний критерій оптимальності; статистичний критерій оптимальності дерева сценарію.

4. Застосувати модель «конструктор» в якості методу побудови та підтримки інтелектуального модуля.

В результаті, запропонована модифікована продукційна модель опису процесів предметної області функціонування СІП АТПК і спосіб її побудови, які дозволяють формалізувати інформаційне забезпечення роботи даного класу АС базою знань і відрізняються від аналогів орієнтацією на підтримку всіх етапів життєвого циклу даного класу систем. Крім того, сформульовані принципи підтримки системи баз знань, які дозволяють вирішувати проблеми надмірності, цілісності БЗ, що відрізняються від інших орієнтованістю на запобігання виникненню дефектних або неоптимальні сценаріїв [21].

4 МЕТОДИКА ФОРМАЛІЗАЦІЇ ЗНАНЬ ПРО ПРЕДМЕТНУ ОБЛАСТЬ

При побудові експертної системи необхідно виробити методи формалізації знань про предметну область і відображення їх в базу знань. На даний момент не існує єдиного підходу до вирішення подібного завдання. Пропонується використовувати модель «конструктор», введену в [30]. В рамках такої моделі, проектування здійснюється відповідно зі стратегією «зверху - вниз». При цьому вводиться 4 рівня проектування (рис. 6).

Рівень типових задач описує набір відокремлених підзадач, які потребують вирішення для досягнення глобальної мети. Рівень типових сценаріїв описує множину сценаріїв, що дозволяють вирішувати типові завдання. Рівень подій описує об'єкти управління та їх параметри, необхідні роботи сценаріїв. Рівень причинно-наслідкових зв'язків містить набір продукційних правил, які є основою для вищих рівнів. В результаті проектування інтелектуального модуля розбивається на наступні етапи:

1. В якості першого кроку будується множина типових задач предметної області. Сукупність елементів типових задач утворює клас, якому в абстрактній моделі відповідає родовий об'єкт «Типова задача».

2. Далі між елементами класу встановлюються горизонтальні, вертикальні і причинно-наслідкові зв'язки.

3. Наступним кроком є побудова для кожної типової задачі множини типових сценаріїв її рішення. Для цього проводиться декомпозиція предметної області на елементи породжуючого сценарії множини подій. При цьому кожна типова задача розглядається окремо, що дозволяє спростити аналіз і формалізацію. Перетини на рівні типових сценаріїв або типових подій штучно прибираються за рахунок дублювання об'єктів перетину.

4. На заключному етапі створюються правила в продукційній базі знань правил, відповідних типовим сценаріям.

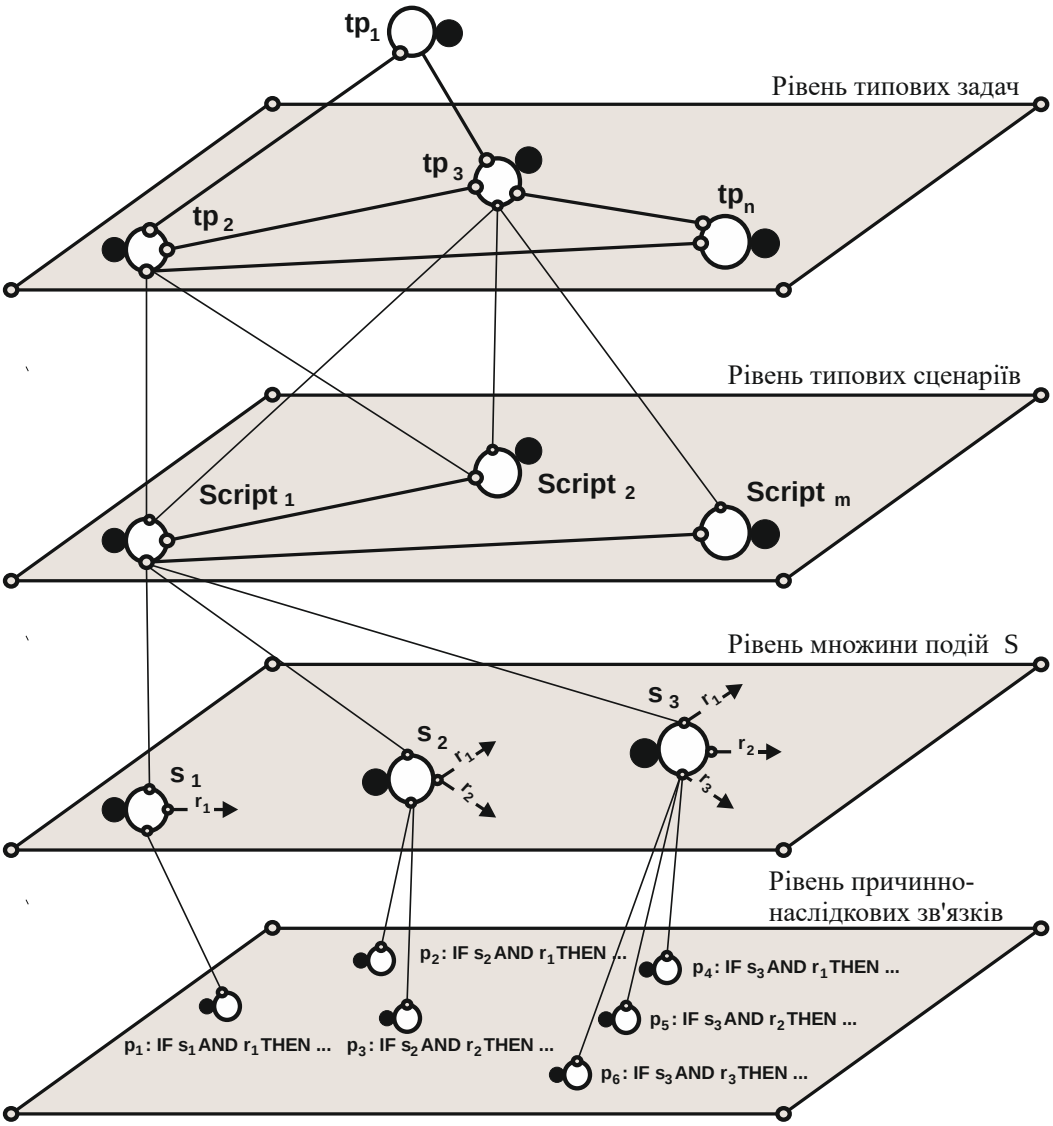

Рисунок 6 - Модель проектування

Графічна інтерпретація об'єктного представлення елементів, що породжують сценарії множини подій зображено на рис. 7.

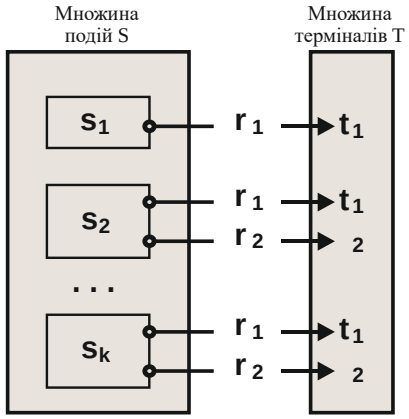

Рисунок 7 - Графічна інтерпретація об'єктного представлення елементів

На концептуальному рівні представимо елементи, що породжують сценарії множини подій в формі вершин графічного об'єкта.

Причинно-наслідкові зв'язки події сценарію, в графічному об'єкті представлені виходящим із вершини ребром. Ребрам графічного об'єкта відповідає ідентифікатор $\{r_1, r_2, ...\}$. Кожне ребро закінчується терміналом, який на рис.3.2 представлений стрілкою. Сукупність терміналів графічного об'єкта визначає результат події, який описаний відповідним причинно-наслідковим зв'язком.

Конструювання сценаріїв здійснюється заповненням терміналів ребер. В термінали вводяться вершини подій. Заповненням терміналів здійснюється формування структури подій, охоплених причинно наслідковими зв'язками (тобто сценарія).

На рис. 8 представлені сценарії, сконструйовані з елементів, що породжують сценарії множини подій.

Заключним кроком проектування сценарної частини концептуальної моделі є формалізація асоціативних зв'язків між: «типовими завданнями» / горизонтальні, вертикальні і причинно наслідкові зв'язки /; «типовими завданнями» і «типовими сценаріями» (зв'язок «один – до багатьох» відображає ситуацію наявності альтернативних варіантів рішення типового завдання) / вертикальні міжрівневі зв'язки /; «типовими сценаріями» / горизонтальні зв'язки /; «типовими сценаріями» - «породжують сценарії множиною подій» (зв'язок типу« багато - до багатьох») / вертикальні міжрівневі зв'язки /; «елементами, що породжують сценарії множини подій» (горизонтальні зв'язки і вертикальні зв'язки) / горизонтальні зв'язки /; «елементами множини подій» - «елементами множини причинно наслідкових зв'язків».

Після проектування сценарної частини здійснюється формування ситуаційних пріоритетів. Для цього доцільно використовувати стратегію "знизу вгору". При цьому в якості першого кроку здійснюється проектування СП для елементів, які породжують сценарії множини подій. Потім генеруються

ситуаційні пріоритети для типових сценаріїв і, нарешті, ситуаційні пріоритети для типових задач.

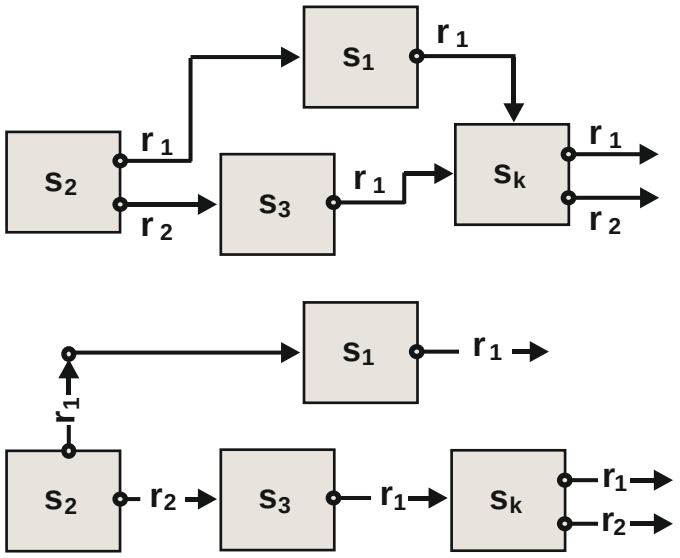

Рисунок 8 - Графічна інтерпретація сценаріїв

На заключному етапі відбувається перехід від концептуального опису до машинно-орієнтованого. При цьому сценарії відображаються в базу правил продукції, а стани - в базу фактів. Кожна типова задача підтримується окремим фрагментом бази знань, що дозволяє комбінувати набір типових завдань, які потрібно вирішувати в рамках впровадження системи на різних платформах. Для їх інформаційного представлення будемо використовувати модель «Конструктор». При її побудові реалізуємо стратегію «зверху - вниз».

Для кожної типової задачі існують один або кілька об'єктів управління з певним набором параметрів, які охоплені зв'язками, описують логіку аналізу поточного стану досліджуваного об'єкта. Вся сукупність параметрів досліджуваного об'єкта однозначно описує його стан.

На відміну від існуючих підходів [26], де стан описувався фізичним станом об'єкта, а ребра - можливі переходи між станами, в даній роботі події описують перевірки в процесі аналізу реального стану об'єкта. Ребра графа описують результати перевірки стану досліджуваного об'єкта. В результаті граф

сценарію приймає форму дерева, в якому коренем є відсутність інформації про стан об'єкта, а листям - інформація про кінцевий стан досліджуваного об'єкта.

В табл. 1 наведені типові завдання, що виникають при підтримці системи автоматизованого тестування програмних комплексів.

Таблиця 1 Типові завдання

Типова задача	Автомати-зуємість	Перено-симість
Управління станом парку тестових і службових машин	Так	Так
Управління конфігурацією тестів	Ні	Так
Управління запуском набору тестів	Так	Так
Управління процесом виконання тестів	Так	Частково
Управління процесом збереження тестових результатів	Так	Так
Автоматизований аналіз тестових результатів і їх зв'язок з системою обліку помилок	Так	Частково
Аналіз статистики запусків і генерація рекомендацій по оптимізації тестування	Так	Так
Генерація нових правил і знань	Частково	Частково

Проілюструємо застосування даного підходу проектуванням фрагмента бази знань (БЗ), який описує одну з типових задач системи підтримки автоматизованого тестування.

На першому етапі проектування моделі СІП САТПК виділимо множину типових задач, які вирішуються в даній предметній області $TP = \{tp_1, tp_2 \ldots, tp_n\}$. Розглянемо типові завдання, що виникають при підтримці САТПК. Як видно з табл. 1, більшість типових завдань переносимі або частково-переносимі, що дозволить впроваджувати подібну систему на різних підприємствах.

З дерева сценарію очевидно, що подія не вносить додаткової інформації в процес аналізу стану пристрою і може бути видалена з дерева аналізу.

Результуюче дерево представлено на рис. 9. Кожному результуючому стану відповідає коригувальна дія.

Для переходу від концептуального опису до машинно орієнтованого опишемо процес аналізу об'єкта «тестовий пристрій» предметної області побудови СІП системою правил продукції.

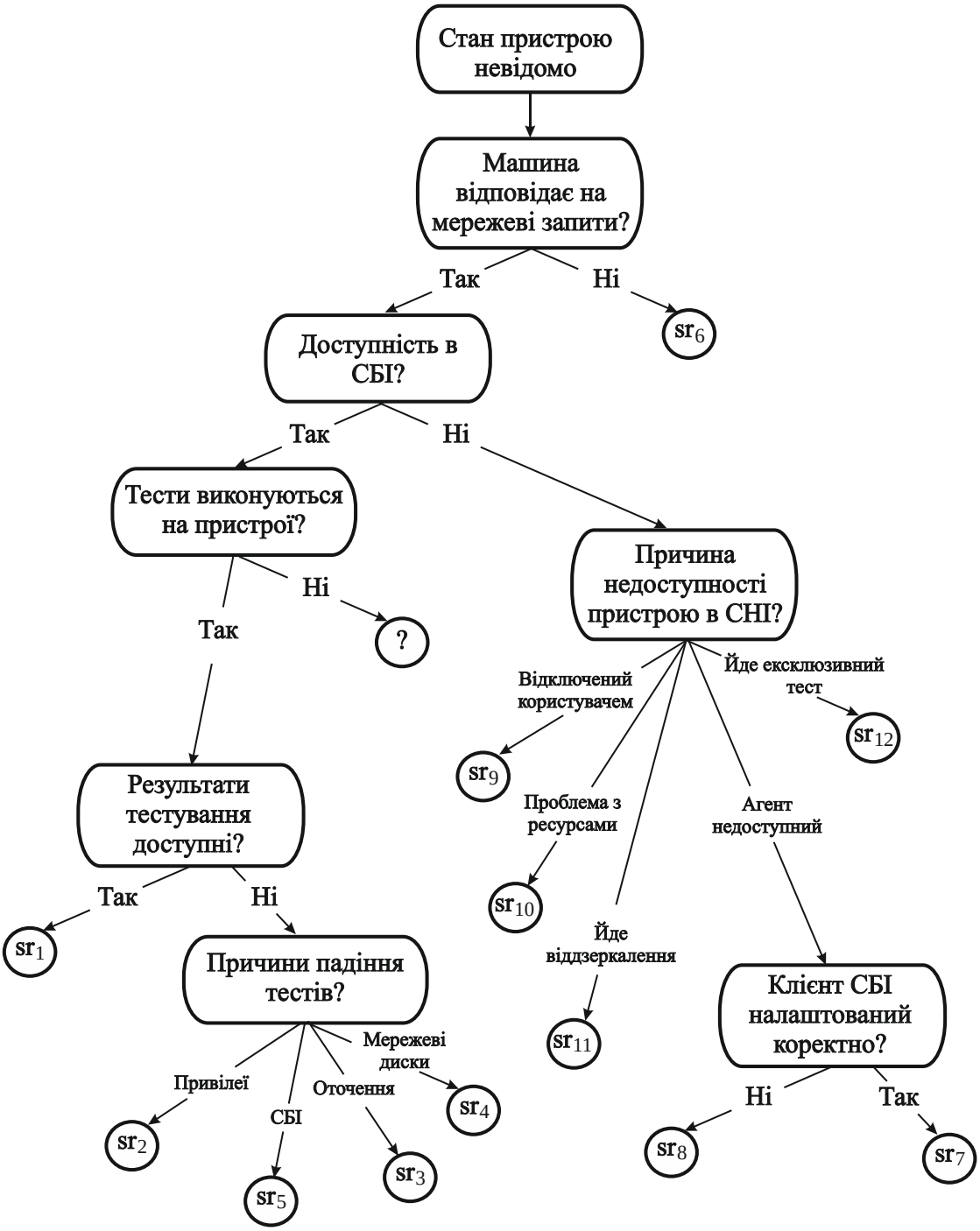

Рисунок 9 - Сценарій аналізу

5 АРХІТЕКТУРА ІНТЕЛЕКТУАЛЬНОГО МОДУЛЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ РОЗРОБЛЕНИХ МЕТОДІВ

В процесі розробки інтелектуального модуля використовувалися дані про роботу системи автоматизованого тестування для окремого проекту. При цьому всі отримані результати були класифіковані наступним чином:

1. Коректне завершення - тестова задача завершилася коректно, надані результати адекватні.

2. Перезапуск тестового завдання для аналізу падінь – тестові інженери можуть перезапускати окремі тести на тестовій матриці для аналізу падінь тесту.

3. Помилка в тесті - некоректний тест був виправлений і тест перезапущений.

4. Проблеми оточення - конфігурація тестового пристрою не дозволяє запускати тести.

5. Проблеми системи безперервної інтеграції - збої обумовлені СБІ.

6. Апаратний збій тестового пристрою - апаратні проблеми і збої також призводять до падіння тестів.

7. Помилки побудови продукту / тестів - помилки побудови призводять до некоректних результатів тестування.

8. Проблеми мережевого сховища - недоступність мережевого сховища.

9. Некоректні ручні дії інженерів - дії тестових інженерів на тестовій машині призвели до некоректної поведінки продукту / тестів.

10. Проблеми системи емуляції - помилки системи емуляції, призводять до некоректного оточенню в момент запуску тесту.

11. Збої мережі - призводять до недоступності тестового пристрою і неможливості отримати результати тестування.

В рамках реалізації запропонованої методики, розроблений модуль інтелектуальної підтримки системи автоматизованого тестування (рис. 10). Архітектура інтелектуального модуля (ІМ) представлена на рис. 10.

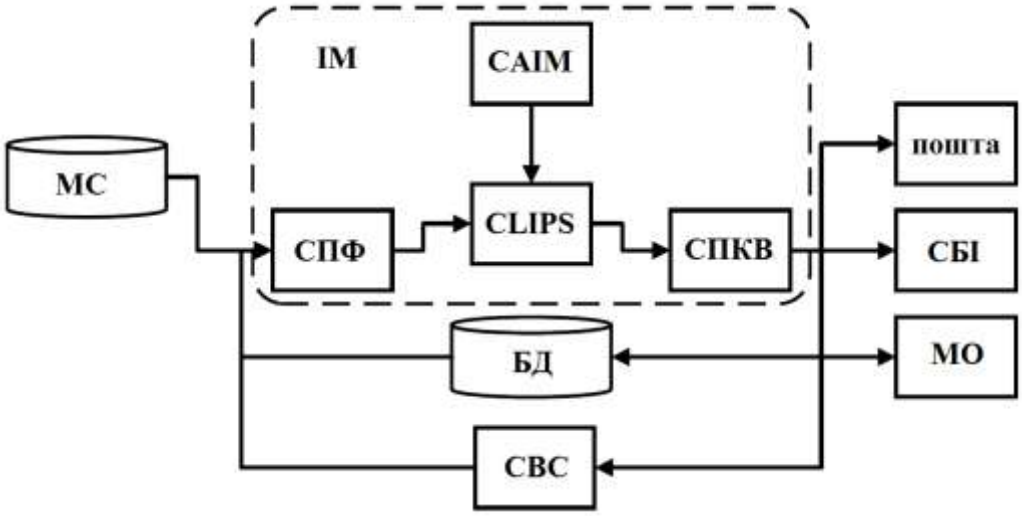

Рисунок 10 - Архітектура інтелектуального модуля

Оболонка для побудови продукційних баз знань CLIPS є ядром даного модуля. Так як дана оболонка оперує з базою фактів і не має вбудованих засобів управління зовнішніми пристроями реалізовані модулі взаємодії:

1. СПФ - система підготовки фактів - перетворює текстові дані в факти відповідно до вимог з експертної системи CLIPS.

2. СПКВ - система підготовки керуючого впливу - перетворює стан експертної системи в конкретну команду або операцію. В якості дії можливо:

- відправка електронного листа і / або повідомлення СМС (пошта);

- зміна статусу об'єктів в системі безперервної інтеграції (СБІ);

- управління мережевим об'єктом (МО);

- модифікація об'єктів в базі даних (БД).

У процесі життєвого циклу СІП АТПК виникають ситуації, що вимагають модифікації бази знань або корекції структури бази фактів. Для виявлення подібних ситуацій, а також, подання інформації про

них адміністраторам використовується система адміністрування інтелектуального модуля (CAIM).

На рис. 10 представлені зовнішні по відношенню до СІП підсистеми, які є основними інтерфейсами для взаємодії з системою автоматизованого тестування програмних комплексів:

1. БД - база даних, що містить інформацію про тестовий процес. Використовується для прискорення отримання даних і доступу до даних.

2. МС - мережеве сховище, містить файли тестових звітів, тестові та утиліти, що використовуються в процесі тестування.

3. СВС - система виконання скриптів, що служить для виконання скриптів, які збирають дані про стан процесу автоматизованого тестування.

Існує кілька практичних завдань, вирішених для успішної реалізації ідей, запропонованих в розділі 2:

1. Одним із завдань, що потребують вирішення для реалізації СІП САТПК, є збір та аналіз даних про предметну область і їх подання до машинно-орієнтованому вигляді. При цьому можна виділити ряд підзадач:

- аналіз стану тестів, тестового парку і формування бази фактів;

- аналіз процесів в предметній області та оптимізація сценаріїв.

2. Іншим завданням є реалізація шляхів керуючого впливу від СІП САТПК до різних елементів системи. Для ілюстрації використовується одна з типових задач в предметної області підтримки системи автоматизованого тестування, розглянута в розділі 2: управління станом парку тестових і службових машин.

В процесі роботи інтелектуального модуля, необхідно зібрати дані про систему автоматизації тестування, перетворити їх в форму придатну для обробки експертною підсистемою, а результат, отриманий від експертної підсистеми, перетворити в керуючий вплив.

Для мінімізації навантаження на інтелектуальний модуль, аналіз проводиться тільки при зміні стану об'єктів управління. Так як не всі зміни стану

об'єктів контролю призводять до зміни даних в базі даних (наприклад, відключення тестового пристрою не призводить до оновлення даних), необхідно ініціювати процедуру опитування з оптимальною частотою. Статистика частоти зміни стану елементів тестової інфраструктури в залежності від часу доби представлена на рис. 11.

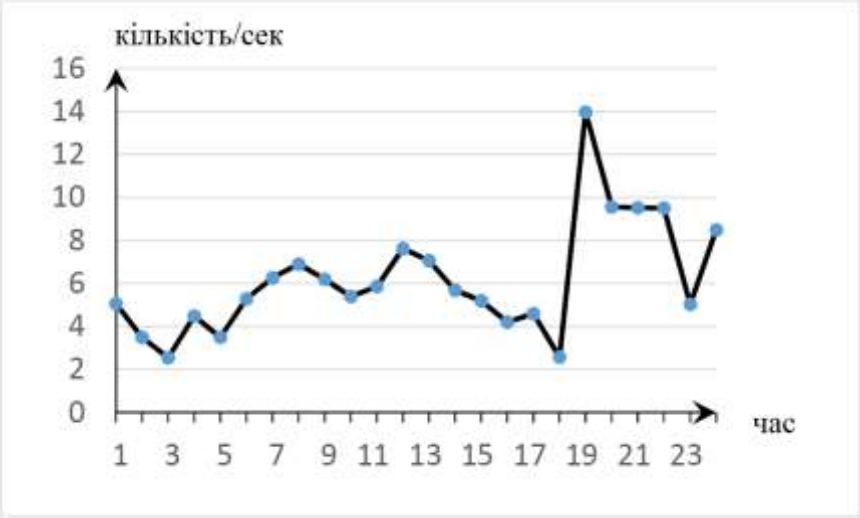

Рисунок 11 - Статистика частоти тестових подій

Як видно з рис. 11, середня кількість подій, що ініціюють аналіз варіюється від 3 до 15 в секунду (пікові значення не перевищують 60 подій в секунду). При цьому кількість об'єктів в системі автоматизованого тестування (тестові завдання, тестові пристрої, об'єкти інфраструктури) становила близько 900 (тести, що знаходяться в черзі виключені для мінімізації кількості контрольованих об'єктів), а 98% подій обумовлені зміною стану тестових завдань.

Одним із завдань, що потребують вирішення, є вибір моменту старту процедури аналізу. З одного боку, чим менше часу пройде з моменту виникнення проблеми до моменту її виявлення і застосування коригуючого впливу, тим менший вплив воно внесе в систему автоматизованого тестування. З іншого боку, дев'ятсот тестів виконуються одночасно на сотнях тестових пристроїв. Крім того, десятки тисяч тестів очікують виконання в черзі. Десятки мережевих пристроїв можуть чинити негативний вплив на стабільність тестових результатів. Постійне опитування всіх елементів системи автоматизованого тестування

вносить істотне додаткове навантаження на мережу і кінцеві пристрої. І нарешті обробка зібраного масиву даних вимагає значних ресурсів (обчислювальних потужностей і часу). Вибір оптимального часу ініціалізації процедури аналізу ґрунтується на балансі між часом реакції на проблему і ресурсами, необхідними на забезпечення аналізу.

Тому ініціація аналізу стану контрольованих об'єктів комбінована:

1. Аналіз стану тестового завдання відбувається в наступних випадках:

- постановка тестового завдання в чергу;
- взяття тестового завдання на виконання;
- завершення виконання тестового завдання.

2. Періодично (раз на годину) відбувається опитування всіх об'єктів системи автоматизованого тестування для виявлення об'єктів «випавших» з області видимості:

- тестових завдань;
- тестових пристроїв;
- об'єктів інфраструктури.

Частота опитування обрана з розрахунку з урахуванням експериментальних даних про середній час, що витрачається на опитування всіх елементів системи з урахуванням таймаутів і вироблення коригувальних дій.

При цьому на підготовку даних, їх аналіз та застосування керуючого впливу потрібно від 5 до 60 секунд. Так як, процес аналізу для різних тестових завдань незалежний, застосовується паралельна процедура підготовки фактів. В результаті для успішної обробки заданої кількості запитів потрібно не менше 75 оброблювачів. У СІП САТПК використовується 100 оброблювачів, які працюють паралельно:

- 1 - збирає інформацію про тестові пристрої;
- 1 - збирає дані про елементи інфраструктури;
- 1 - контролює елементи інфраструктури;

 збирають інформацію про тестові завдання в моменти зміни їх стану.

Експериментально було показано, що для підтримки роботи достатньо чотирьох ядерний процесор Intel Core i5-4670 з номінальною тактовою частотою 2.3 ГГц і оперативною пам'яттю 8 ГБ, мережевий інтерфейс 1Гбіт/с, під управлінням операційної системи Ubuntu 14.04. При цьому середнє завантаження обчислювальних потужностей становить 40%.

Оцінки ефективності розроблених методів. Як програмну платформу для реалізації інтелектуальної системи підтримки було обрано такі інструментальні програмні засоби:

1. Середовище розробки експертних систем CLIPS.

2. MySQL - найбільш поширена реляційна СУБД з відкритими кодами. Була успадкована від розробленої раніше системи автоматизації тестування.

3. Скриптова мова Python був обраний з огляду на компактність, легкості написання програм на ньому, великої кількості розширень, що дозволяють взаємодіяти з різними підсистемами (база даних, web-сервіси, мережеві диски і пристрої) і підтримки регулярних виразів.

Всі перераховані інструментальні програмні засоби є вільно поширюваними і не пред'являють особливих вимог до апаратних засобів, отже, прототипування не вимагає додаткових витрат. Крім того, це дозволяє переносити і адаптувати систему підтримки для автоматизованих систем, відмінних від тієї, для якої здійснювалося прототипування.

Для оцінки ефективності розроблених методів використовується інтегральний критерій ефективності системи інтелектуальної підтримки акумулює кілька часткових параметрів. Одиницею вимірювання роботи є тестовий цикл – повний набір тестів, які необхідно виконати, для отримання інформації про якість продукту. Кожен частковий параметр вимірюється для одного тестового циклу. Розглядаються наступні параметри:

1. Частка відмов, виявлених автоматичному режимі (Qвиявл).

2. Частка відмов, виправлених в автоматичному режимі (Qвиправ).

3. Час необхідний для отримання інформації про якість продукту - повний час тестового циклу (Тцикла).

4. Час що витрачається персоналом на аналіз одного тестового циклу (Тперс).

5. Частка устаткування, виведеного з тестування через відмови (Qустат). Критерій ефективності виражається у вигляді формули (4.1). Зменшення значення даного критерію говорить про зростання ефективності інтелектуальної підтримки.

$$C_{ефект} = \frac{K_{виявл} \cdot Q_{виявл} + K_{виправ} \cdot Q_{виправ}}{K_{цикла} \cdot T_{цикла} + K_{перс} \cdot T_{перс} + K_{устат} \cdot Q_{устат}}$$

Коефіцієнти $K = \left\{ K_{виявл}, K_{виправ}, K_{цикла}, K_{перс}, K_{устат} \right\}$ виставляються в відповідності до внеску параметра в загальну ефективність на підставі експертної оцінки $K = \{100,10,5,10,3\}$. В результаті модель ефективності можна виразити формулою 4.2.

$$MAX\left(C_{ефект}\right) = MAX\left(\frac{100 \cdot Q_{виявл} + 10 \cdot Q_{виправ}}{5 \cdot T_{цикла} + 10 \cdot T_{перс} + 3 \cdot Q_{устат}}\right)$$

Пілотну версію інтелектуального модуля необхідно впроваджувати в базовому проекті в 3 етапи:

1. На першому етапі інтелектуальний модуль аналізує історичні дані. Результати роботи системи інтелектуальної підтримки необхідно порівнювати з результатами аналізу, проведеного тестовими інженерами в процесі щоденної роботи. Потрібно відзначити, що так як аналіз ведеться тільки на підставі файлів звітів (неможливо виконувати інтерактивні команди для історичних даних), контролювати тільки ті стани об'єктів, інформація про яких може бути отримана з цих звітів.

2. На другому етапі система повинна працювати з актуальними даними та надавати інженерам свої рекомендації. Інженери повинні брати рішення про вірність або невірність результатів аналізу наданих системою інтелектуальної підтримки. Після чого необхідно застосовувати або не застосовувати коригувальні дії.

3. На третьому етапі система повинна працювати в автоматичному режимі, коли коригувальна дія застосовується без участі людини (якщо це можливо).

На всіх етапах крім верифікації роботи системи необідна її коригування та оновлення бази знань.

Критерієм якості роботи інтелектуального модуля є відсутність регресій. Під регресією розуміється невірний результат аналізу для заданої відомої інфраструктурної проблеми. Наприклад:

1. Якщо інтелектуальний модуль перестав детектувати проблему недоступності мережевого диска, це є регресією.

2. Якщо інтелектуальний модуль не зміг надати аналіз для інфраструктурної проблеми, що раніше не зустрічалася і повідомив тестового інженера про необхідність ручних досліджень, це не є регресією.

Як вже говорилося, на першому етапі використовуються дані по тестовим запускам, які вже були проаналізовані тестовими інженерами. Використання історичних даних дозволило з одного боку налагодити базу знань, з іншого переконатися в працездатності підходу без впливу на робочий процес. Основними завданнями в рамках першого етапу впровадження:

- перевірка працездатності бази знань, в тому числі переконатися в вірності результату аналізу;

- виправлення всіх виявлених проблем.

На першому етапі використовувалися файли тестових звітів. Процес аналізу проводився ітеративно. У міру виявлення проблем вносилися зміни в базу знань, скрипти підготовки фактів і інші елементи інфраструктури. Після того як всі

файли звітів були оброблені процес починався спочатку. Ознакою закінчення етапу було аналіз всіх вибраних файлів тестових звітів без виявлення проблем в інфраструктурі і інтелектуальному модулі (регресій).

Основними результатами впровадження системи автоматизованої корекції стали:

1. Зниження частки простою обладнання. Одним з проявів інфраструктурних проблем є миттєве «падіння» тесту, що виконується. Якщо існує кілька ідентичних тестових пристроїв і на одному з них з'являється подібна проблема, протягом 10-15 хвилин черга тестів, розрахована на виконання протягом 5-10 годин, завершується, проходячи через проблемний пристрій. В результаті простоюють як проблемний, так і працездатні пристрої тієї ж конфігурації. Аналіз даних по роботі системи автоматизованого тестування показав, що близько 10% часу тестові пристрої простоюють через подібні проблеми.

Потрібно відзначити, що впровадження інтелектуальних підказок на другому етапі дозволило знизити частку обладнання, що простоює через інфраструктурні проблеми з 10% до 8%. Скорочення не настільки значне, так як на даному етапі потрібна участь людини, що неможливо в неробочі години. Автоматична корекція звела частку такого обладнання до 0.1% за рахунок швидкої реакції на виникаючі проблеми в будь-який час доби, вихідні та святкові дні.

2. Скорочення часу тестового циклу. Всі тестові запуски розбиті на тестові цикли, що визначають набір тестів і тестових пристроїв необхідних для цільового (випуск / розробка продукту) тестування. Для завершення тестування необхідно, щоб всі тести з заданого циклу завершилися коректно. Повний час тестового циклу - час від запуску першого тесту до завершення останнього перезапуску останнього тесту. У разі виникнення інфраструктурних проблем час тестового циклу значно збільшується, що призводить до великих черг на окремих

платформах, що в свою чергу ще більше збільшує повний час виконання тестового циклу.

За базу для вимірювання береться максимальна сума часу виконання всіх тестів для однієї тестової конфігурації (платформи) в даному тестовому циклі. Потрібно відзначити що, в ідеальному випадку тривалість циклу складе базовий час, поділений на кількість тестовий пристроїв в даній конфігурації. В якості міри береться перевищення реальної тривалості циклу над базовим часом. Крім того, завершення тестового циклу в неробочий час призводить до збільшення часу циклу на термін очікування робочого часу в разі необхідності аналізу з боку тестових інженерів.

3. Скорочення часу, що витрачається інженерами з тестування на аналіз проблем. Автоматизація процесу підтримки тестування мається на увазі передачу частини функцій, раніше виконуваних інженерами з тестування, інтелектуальному модулю. До впровадження системи в середньому потрібно 48 людино-годин в тиждень. Після впровадження інтелектуальних підказок витрачаємий час скоротився до 30 годин на тиждень. Після повного впровадження системи потрібно близько 17 годин в тиждень на підтримку автоматизованого тестування. З них лише 2 години витрачаються на аналіз інфраструктурних проблем, інші витрачаються на аналіз проблем в тестах і тестуємому продукті.

ВИСНОВКИ

В результаті досліджень в рамках даної роботи було виявлено невідповідність між швидкістю зростання складності програмних продуктів, вимогами до якості і застосовуваними методами підтримки тестування, що складаються в тому, що існуючі методи підтримки не дозволяють досягти необхідної якості продуктів, що випускаються з урахуванням очікуваного зростання складності ПЗ в найближчі 5 років. Причина даного протиріччя полягає в тому, що основне зусилля в дослідженнях в області автоматизації тестування робиться на розробку автоматизованих тестів. Процес підтримки автоматизованого тестування покладається на плечі інженерів з тестування та адміністраторів мережі. Несистемний підхід не може забезпечити зростання продуктивності команд підтримки пропорційно зростанню складності тестованого ПЗ.

Запропоновано використовувати інтелектуальний модуль підтримки, який здійснює первинний аналіз результатів тестування. Реалізація даного модуля зажадала модифікації базового формату правил продукційних систем і його адаптацію до досліджуваної предметної області:

1. Введення пріоритетів правил і розробка методу їх автоматизованого виставлення.

2. Введення ідентифікатора фрагмента бази знань. Фрагментованість запропоновано для спрощення підтримки бази знань і мінімізації сценаріїв.

3. Використання дескриптора ситуації дозволяє уникнути ряд недоліків, властивих продукційним базам знань.

Крім того, були розроблені принципи підтримки інтелектуального модуля на всіх етапах життєвого циклу СІП орієнтовані на виключення можливості виникнення некоректного або неефективного сценарію. Пропонується

використовувати модель «конструктор» в якості механізму побудови бази знань. Вирішено ряд практичних завдань реалізації інтелектуального модуля:

1. Запропоновано алгоритм роботи інтелектуального модуля.

2. Розроблено метод формування бази фактів.

Для підтвердження достовірності отриманих результатів було виконано ряд експериментів на системі автоматизації тестування ПЗ. Було створено програмно-апаратне середовище, яке в пілотному режимі підтвердила працездатність і перспективність запропонованих ідей.

Таким чином, представлені вище результати дослідження дозволяють стверджувати, що сформульована наукова проблема - розробка теоретичного підходу до питань побудови інструментаріїв інтелектуальної підтримки процесу автоматизованого тестування - вирішена з урахуванням прийнятих обмежень і припущень, що підтверджується впровадженням результатів і проведеними обчислювальними експериментами.

В результаті впровадження 92% інфраструктурних проблем стали детектуватися в автоматичному режимі, що дозволило досягти наступних результатів: час, що витрачається інженерами з тестування та адміністраторами на підтримку процесу тестування, скоротилося на 65%; підвищилася ефективність використання тестової інфраструктури на 9%; скоротився час необхідний для отримання достовірних тестових результатів.

Все це підтверджує практичну реалізованість, своєчасність і важливість ідей, запропонованих в рамках даної магістерської роботи. Подальше підвищення ступеня автоматизації в даній предметній області вимагає вирішення питань автоматизації процедури виділення нових знань, їх подання до машинно-орієнтованого виду і верифікації.

ЛІТЕРАТУРА

1. Аверкин, А. Н. Толковый словарь по искусственному интеллекту / А. Н. Аверкин, М. Г. Гаазе-Раппопорт, Д.А. Поспелов – М.: Радио и связь. – 2002. –547с.

2. Алпайдин, Э. Машинное обучение: новый искусственный интеллект / Э. Алпайдин : пер. с англ. — М. : Издательская группа «Точка», Альпина Паблишер, 2017. — 208с.

3. Баранчеев, В.П. Управление знаниями в инновационной сфере: Учеб. / В.П. Баранчеев - М.: ООО «Благовест-В», 2007. - 272с.

4. Баринов, В.А. Организационное проектирование / В.А. Баринов. - М.: ИНФРА-М, 2010. - 384с.

5. Башлыков, А.А. Экспертные системы поддержки принятия решений в энергетике / А.А. Башлыков, А.П. Еремеев/ Под. ред. А.Ф. Дьякова. - М.: Изд-во МЭИ, 2014. –216с.

6. Белоусов, А. И. Дискретная математика./ А. И. Белоусов, С. Б.Ткачев — М.: МГТУ, 2006. —744с.

7. Бешелев, С.Д. Математико-статистические методы экспертных оценок./ С.Д. Бешелев, Ф.Г. Гурвич –М.: Статистика, 2001. –263с.

8. Бобцов, А. А. Адаптивное и робастное управление неопределенными системами по выходу / А.А.Бобцов – СПб: Наука. – 2011. – 174с.

9. Букович, У. Управление знаниями: руководство к действию./ У.Букович, Р.Уильямс –Пер. с англ.— М.: ИНФРА-М, 2002.— 504 с.

10. Вагин, В.Н. Достоверный и правдоподобный вывод в интеллектуальных системах / В.Н. Вагин, Е.Ю. Головина, А.А. Загорянская –М.: Изд-во ФИЗМАТЛИТ, 2008. –787с.

11. Ватутин, Э.И. Комбинаторно-логические задачи синтеза разбиений параллельных алгоритмов логического управления при проектировании логических мультиконтроллеров / Э.И. Ватутин, И.В. Зотов, В.С. Титов.– Курск: изд-во КурскГТУ, 2010.– 200с.

12. Вертакова, Ю.В. Управленческие решения: разработка и выбор: учебное пособие / Ю.В.Вертакова, И.А. Козьева, Э.Н. Кузьбожев.– М. Кронус, 2005.–352 с.

13. Вольфсон, Б. Гибкое управление проектами и продуктами. // Б. Вольфсон –Питер. – 2017. – 144с.

14. Гаврилова, Т. А. Базы знаний интеллектуальных систем / Т. А. Гаврилова, В.Ф. Хорошевский // СПб: Питер. – 2000. –382с.

15. Гаврилова, Т. А. Извлечение и структурирование знаний для экспертных систем / Т. А. Гаврилова, К. Р. Червинская – М.: Радио и связь. – 2004. –200с.

16. Гнеденко Б. В. Введение в теорию массового обслуживания / Б. В. Гнеденко, И. Н. Коваленко. – 6-е изд. – М.: УРСС: Изд-во ЛКИ, 2013. – 352 с.

17. Джексон, П. Введение в экспертные системы / П. Джексон. - М.: Вильямс, 2001. - 624 с.

18. Дюваль, П.М. Непрерывная интеграция. / Поль М. Дюваль, М. Стивен – Вильямс, 2008. –240 с.

19. Искусственный интеллект. Справочник в трех томах / под ред. В. Н. Захарова, Э. В. Попова, Д. А. Поспелов, В. Ф. Хорошевского // М.: Радио и связь. – 2001. –1447с.

20. Коваленко, О.О. Аналіз тенденцій розвитку систем автоматизованого тестування / І.В. Гурман, А.В. Краснік, О.О. Коваленко / Тези доповідей XVII міжнародної наукової конференції студентів, аспірантів та молодих учених. / ред. кол. Д. Струнін (голова ВНТ ВІКНУ) – К., - 2019. –С.73

21. Коваленко, О.О. Модифікована архітектура системи автоматизованого тестування / О.О. Коваленко, В.М.Джулій //Всеукраїнська конференція студентів та молодих науковців «Інтелектуальний потенціал 2019», 20-22.11.2019, - ХНУ.- Частина 1. Комп‘ютерні системи та кібербезпека – 34-38с.

22. Костевич, Л. С. Математическое программирование: Информационные технологии оптимальных решений: Учеб. пособие / Л. С. Костевич. —Мн.: Новое знание, 2003.– 150 с.

23. Круглов, В.В. Интеллектуальные информационные системы: компьютерная поддержка систем нечеткой логики и нечеткого вывода. / В.В. Круглов, М.И.Дли – М.: Физматлит, 2002. – 350 с.

24. Лапшин, В.А. Онтологии в компьютерных системах. /В.А. Лапшин – М.: Научный мир, 2010.–222с.

25. Левитин, А. В. Алгоритмы. Введение в разработку и анализ. / А. В. Левитин – М.:Вильямс, 2006.– 576 с.

26. Мариничева, М.К. Управление знаниями на 100%: Путеводитель для практиков / М.К. Мариничева. — М.: Альпина Бизнес Букс, 2008. — 320 с.

27. Мартин, Роберт Быстрая разработка программ. Принципы, примеры, практика. /Роберт Мартин, Джеймс Ньюкирк, Роберт Косс — Изд-во: Диалектика-Вильямс, 2004. — 752 с.

28. Мэскон, М. Основы менеджмента / М. Мэскон, М. Альберт, Ф.Хедоури. – М.: Дело, 2002. - 704 с.

29. Минский, М. Фреймы для представления знаний / М. Минский. – М:Энергия, 2001. - 151 с.

30. Мильнер, Б.З. Управление знаниями / Мильнер Б.З.– М.: Инфра-М, 2008.– 178с.

31. Новицкий, Н.И. Сетевое планирование и управление производством. Учеб.-практ. пособие / Н.И. Новицкий. – М.: Новое знание, 2004. –159 с.

32. Ногин, В. Д. Принятие решений в многокритериальной среде: количественный подход / В. Д. Ногин– М.: Физматлит,2005.– 176с.

33. Олифер, В. Г. Компьютерные сети. Принципы, технологии, протоколы /В. Г. Олифер, Н. А.Олифер - СПб.: Питер, 2017. - 992 с.

34. Орлов, А. И. Экспертные оценки. Учебное пособие./ А. И. Орлов– М.: ИВСТЭ, 2002. – 252с.

35. Поспелов, Д. А. Ситуационное управление. Теория и практика / Д.А.Поспелов – М.: Наука. – 2006. –228 с.

36. Рассел, С. Искусственный интеллект: современный подход = Artificial Intelligence: A Modern Approach (AIMA) / С. Рассел, П. Норвиг – М.: "Вильямс". – 2007. – 1424с.

37. Сердюк, В. А. Организация и технологии защиты информации / В. А. Сердюк. – М.: Издательский дом Государственного университета – Высшей школы экономики, 2011. – 571 с.

38. Советов, Б. Я. Моделирование систем : учебник для бакалавров / Б. Я. Советов, С. А. Яковлев. — 7-е изд. — М. : Издательство Юрайт, 2015. — 343 с.

39. Советов, Б. Я. Представление знаний в информационных системах: учебник для студ. учреждений высш. проф. образования / Б. Я.Советов, В. В. Цехановский, В. Д. Чертовской. — 2-е изд., стер. — М. : Издательский центр ≪Академия≫, 2012. — 144 с.

40. Сольницев, Р.И. Системы управления "природа - техногеника"/ Р. И. Сольницев, Г. И. Коршунов. – Санкт-Петербург : Политехника, 2013. - 204с.

41. Таранцев, А. А. Инженерные методы теории массового обслуживания / А. А. Таранцев. – Санкт-Петербург: Наука, 2007. – 164 с.

42. Тарасюк, М. В. Защищенные информационные технологии / М. В. Тарасюк. – М.: Солон-пресс, 2004. – 191 с.

43. Тархов, Д. А. Нейросетевые модели и алгоритмы : справочник / Д. А. Тархов. — Москва : Радио- техника, 2014. — 349 с. : ил

44. Татт, У. Теория графов / У. Татт, пер. с англ. Г. П. Гаврилова. – М. Мир, 1988. – 424 с.

45. Тихоненко, О. М. Модели массового обслуживания в информационных системах: учебное пособие для ВУЗов / О. М. Тихоненко. – Минск: Технопринт, 2003. – 327 с.

46. Частиков, А.П. Разработка экспертных систем. Среда CLIPS – А.П. Частиков, Т.А. Гаврилова, Д.Л. Белов. - Санкт-Петербург.: БХВ- Петербург, 2003. – 608 с.

47. Шаньгин, В. Ф. Защита информации в компьютерных системах и сетях. / В. Ф. Шаньгин. - М.: ДМК Пресс, 2012. – 576 с.

ДОДАТОК А

Код (лістинг) програмного забезпечення (сценарій аналізу стану пристрою)

```
;(clear)
;(reset)
;(run)
(set-salience-evaluation every-cycle)
;(defmodule DEVMON)
;(focus DEVMON)
(Deftemplate device-mon; ідентифікатор дескриптора
(Slot hostname (type STRING)); мережеве ім'я пристрою
(Slot time (type NUMBER)); дата / час збору факту
(Multislot repair (default unknown)); коригувальна дія
(Multislot tests_executing (default unknown)); тести виконуються
(Multislot test_result (default unknown)); тестові звіти доступні
(Multislot test_fail (default unknown)); причина тестових помилок
(Multislot ci_available (default unknown)); доступність агента CCM
(Multislot ci_config (default unknown)); конфігурація агента CCM
(Multislot network (default unknown)); доступність пристрою
; по мережі
)
(Deftemplate device-res; ідентифікатор дескриптора
(Slot hostname (type STRING)); мережеве ім'я пристрою
(Multislot repair (default unknown)); коригувальна дія
)
(Defglobal? * Sr1s * = 0); статистичний пріоритет
(Defglobal? * Sr1e * = 20); експертний пріоритет
(Defrule sr1-ok "Тестове пристрій в порядку"
(Declare (salience (+? * Sr1s *? * Sr1e *))); пріоритет правила
розраховується як сума експертного та статистичного пріоритетів
? Data-fact <- (device-mon (repair unknown) (test_fail ok) (tests_executing
OK) (test_result available))
=>
(Bind? * Sr1s * (+? * Sr1s * 1))
(Modify? Data-fact (repair ok)); коригувальна дія - не потрібно
)
(Defglobal? * Sr2s * = 0); статистичний пріоритет
(Defglobal? * Sr2e * = 20); експертний пріоритет
(Defrule sr2-group-policy "Неправильні налаштування групових політик"
(Declare (salience (+? * Sr2s *? * Sr2e *))); пріоритет правила
розраховується як сума експертного та статистичного пріоритетів
? Data-fact <- (device-mon (repair unknown) (test_fail group_policy))
=>
(Bind? * Sr2s * (+? * Sr2s * 1))
(Modify? Data-fact (repair set-group-policies)); коригувальна дія -
змінити групові плоітікі
)
(Defglobal? * Sr3s * = 0); статистичний пріоритет
(Defglobal? * Sr3e * = 20); експертний пріоритет
```

```
(Defrule sr3-environment "Некоректне оточення"
(Declare (salience (+? * Sr3s *? * Sr3e *))); пріоритет правила
розраховується як сума експертного та статистичного пріоритетів
? Data-fact <- (device-mon (repair unknown) (test_fail environment))
=>
 (Bind? * Sr2s * (+? * Sr3s * 1))
(Modify? Data-fact (repair disable send-notice-to-admin)); коригуючий
дію - відключити і повідомити адміністратора
)
(Defglobal? * Sr4s * = 0); статистичний пріоритет
(Defglobal? * Sr4e * = 20); експертний пріоритет
(Defrule sr4-environment "Мережеві диски недоступні"
(Declare (salience (+? * Sr4s *? * Sr4e *))); пріоритет правила
розраховується як сума експертного та статистичного пріоритетів
? Data-fact <- (device-mon (repair unknown) (test_fail net-disks))
=>
(Bind? * Sr2s * (+? * Sr4s * 1))
(Modify? Data-fact (repair remap-network-disks)); коригувальна дія -
підключити мережеві диски
)
(Defglobal? * Sr5s * = 0); статистичний пріоритет
(Defglobal? * Sr5e * = 20); експертний пріоритет
(Defrule sr5-environment "Проблема CCM"
(Declare (salience (+? * Sr5s *? * Sr5e *))); пріоритет правила
розраховується як сума експертного та статистичного пріоритетів
? Data-fact <- (device-mon (repair unknown) (test_fail ci-failure))
=>
(Bind? * Sr2s * (+? * Sr5s * 1))
(Modify? Data-fact (repair disable send-notice-to-admin)); коригуючий
дію - відключити і повідомити адміністратора
)
(Defglobal? * Sr6s * = 0); статистичний пріоритет
(Defglobal? * Sr6e * = 0); експертний пріоритет
(Defrule sr6-network-unavailable "Тестове пристрій недоступно по мережі?"
(Declare (salience (+? * Sr6s *? * Sr6e *))); пріоритет правила
розраховується як сума експертного та статистичного пріоритетів
? Data-fact <- (device-mon (repair unknown) (network unavailable))
=>
(Bind? * Sr6s * (+? * Sr6s * 1))
(Modify? Data-fact (repair send-notice-to-admin)); коригувальна дія -
Повідомити адміністратора
)
(Defglobal? * Sr7s * = 0); статистичний пріоритет
(Defglobal? * Sr7e * = 20); експертний пріоритет
(Defrule sr7-ci-agent-down "CCM агент зупинений"
(Declare (salience (+? * Sr7s *? * Sr7e *))); пріоритет правила
розраховується як сума експертного та статистичного пріоритетів
? Data-fact <- (device-mon (repair? Repair))
(Device-mon (repair unknown) (ci_available offline) (ci_config ok))
```

```
=>
(Bind? * Sr6s * (+? * Sr6s * 1))
(Modify? Data-fact (repair start-ci-agent)); коригувальна дія - Запуск
агента CCM
)
(Defglobal? * Sr8s * = 0); статистичний пріоритет
(Defglobal? * Sr8e * = 20); експертний пріоритет
(Defrule sr8-ci-agent-unconfigured "CCM агент налаштований"
(Declare (salience (+? * Sr8s *? * Sr8e *))); пріоритет правила
розраховується як сума експертного та статистичного пріоритетів
? Data-fact <- (device-mon (repair unknown) (ci_available offline)
(Ci_config ko unknown))
=>
(Bind? * Sr7s * (+? * Sr8s * 1))
(Modify? Data-fact (repair config-ci-agent start-ci-agent)); коригуюча
дія - Конфігурувати і запустити агента CCM
)
(Defglobal? * Sr9s * = 0); статистичний пріоритет
(Defglobal? * Sr9e * = 20); експертний пріоритет
(Defrule sr9-ci-agent-disabled "CCM агент відключений користувачем"
(Declare (salience (+? * Sr9s *? * Sr9e *))); пріоритет правила
розраховується як сума експертного та статистичного пріоритетів
? Data-fact <- (device-mon (repair unknown) (ci_available disabled))
=>
(Bind? * Sr9s * (+? * Sr9s * 1))
(Modify? Data-fact (repair send-notice-to-user)); коригувальна дія -
відправити повідомлення користувачу, котрий відключив пристрій
)
(Defglobal? * Sr10s * = 0); статистичний пріоритет
(Defglobal? * Sr10e * = 20); експертний пріоритет
(Defrule sr10-ci-low-resourses "проблеми з локальними ресурсами"
(Declare (salience (+? * Sr10s *? * Sr10e *))); пріоритет правила
розраховується як сума експертного та статистичного пріоритетів
? Data-fact <- (device-mon (repair unknown) (test_fail low-disk lowmemory))
=>
(Bind? * Sr10s * (+? * Sr10s * 1))
 (Modify? Data-fact (repair disable send-notice-to-admin)); коригуючу
дію - відключити і повідомити адміністратора
)
(Defglobal? * Sr11s * = 0); статистичний пріоритет
(Defglobal? * Sr11e * = 20); експертний пріоритет
(Defrule sr11-ci-mirror "віддзеркалення"
(Declare (salience (+? * Sr11s *? * Sr11e *))); пріоритет правила
розраховується як сума експертного та статистичного пріоритетів
? Data-fact <- (device-mon (repair unknown) (ci_available disabled mirror))
=>
(Bind? * Sr11s * (+? * Sr11s * 1))
(Modify? Data-fact (repair ok)); коригувальна дія - не потрібно
)
```

(Defglobal? * Sr12s * = 0); статистичний пріоритет
(Defglobal? * Sr12e * = 20); експертний пріоритет
(Defrule sr12-ci-exclusive "виповнюється тест в ексклюзивному режимі"
(Declare (salience (+? * Sr12s *? * Sr12e *))); пріоритет правила
розраховується як сума експертного та статистичного пріоритетів
? Data-fact <- (device-mon (repair unknown) (ci_available disabled
exclusive))
=>
(Bind? * Sr12s * (+? * Sr12s * 1))
(Modify? Data-fact (repair ok)); коригувальна дія - не потрібно
)
(Defglobal? * Sr13s * = 0); статистичний пріоритет
(Defglobal? * Sr13e * = 20); експертний пріоритет
(Defrule sr13-unavailable-results "недоступні тестові результати"
(Declare (salience (+? * Sr13s *? * Sr13e *))); пріоритет правила
розраховується як сума експертного та статистичного пріоритетів
? Data-fact <- (device-mon (repair unknown) (test_result not_available))
=>
(Bind? * Sr13s * (+? * Sr13s * 1))
(Modify? Data-fact (repair send-notice-to-admin)); коригувальна дія -
повідомити адміністратора
)
if ci-agent-not-available then (ci-fail-reasons ci-agent-missing))
(Defrule check-ci-agent "Клієнт CCM налаштований коректно?"
(Ci-fail-reasons ci-agent-missing) (not (repair?))
=> (If ci-agent-is-configured
then (repair start-ci-agent)
else (repair configure-ci-agent))