Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка
другий (магістерський)
(рівень вищої освіти)

Дослідження методів тестування програмного забезпечення на основі моделей
(тема)

Виконав: студент 2 курсу, групи ІПЗм–18–2
спеціальності 121–Інженерія програмного забезпечення
(код і повна назва спеціальності)
Освітньо–професійної програми
Інженерія програмного забезпечення
(повна назва освітньої програми)
Кащенко Ю.Р.
(прізвище, ініціали)
Керівник проф. Смеляков К.С.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри, проф. _____ З.В.Дудар

2020 р.

Харківський національний університет радіоелектроніки

Факультет Комп’ютерних наук

Кафедра Програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121–Інженерія програмного забезпечення

(код і повна назва)

Освітньо–професійна програма Інженерія програмного забезпечення

(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Кащенку Юрію Романовичу

(прізвище, ім’я, по батькові)

1. Тема роботи Дослідження методів тестування програмного забезпечення на основі моделей

затверджена наказом по університету від “___” _____20 ___ р № _____

заповнюється вручну після отримання наказу

2. Термін подання студентом роботи до екзаменаційної комісії

15 травня 2020 р.

3. Вихідні дані до роботи алгоритми та методи аналізу та створення моделей для тестування програмних систем, пояснювальна записка. Використовувати ОС Linux, середовище композиційного проєктування

4. Перелік питань, що потрібно опрацювати в роботі мета роботи, аналіз проблемної галузі і постановка задачі, огляд вже існуючих методів створення моделей для тестування програминх систем, порівняння методів, мов та стилів створення моделей, класифікація методів опанування складності моделей, комбінування мови, стилю проектування, та знайдених властивостей у єдиний метод

5. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка *
1.	Аналіз предметної галузі		
2.	Огляд існуючих методів та алгоритмів		
3.	Алгоритми аналізу та прогнозування цін		
4.	Підготовка пояснювальної записки		
5.	Спецчастина		
6.	Підготовка презентації та доповіді		
7.	Попередній захист		
8.	Нормоконтроль, рецензування		
9.	Занесення диплома в електронний архів		
10.	Допуск до захисту у зав. кафедри		
* заповнюється вручну після виконання чергового пункту			

Дата видачі завдання _____2020 р.

Студент _____
(підпис)

Керівник роботи _____ __проф. Смеляков К.С._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до роботи містить: 47 с., 3 рис., 1 таблиця, 25 джерел.

ТЕСТУВАННЯ НА ОСНОВІ МОДЕЛЕЙ, ФОРМАЛЬНІ МЕТОДИ, ОПАНУВАННЯ СКЛАДНОСТІ, МЕРЕЖІ ПЕТРІ, КЛАСТЕРИЗАЦІЯ, ПРОЄКТУВАННЯ МОДЕЛЕЙ, МЕТОД МОДЕЛЮВАННЯ.

Метою роботи є дослідження проблем у галузі тестування на основі моделей, що перешкоджає поширеною цієї техніки, і створення методу моделювання, який не вимогливим, інтуїтивним способом допомагає дозволяє ефективно працювати зі складністю моделей.

Методологією дослідження є наука проєктування, значною частиною якої є систематичний огляд літератури — спосіб навмисного та цілеспрямованого пошуку, відбору, оцінки та узагальнення наукових доказів та рішень для відповіді на конкретне дослідницьке питання таким чином, що є прозорим і відтворюваним. Витяги із літератури слугують складовими блоками створюваного методу моделювання.

MODEL-BASED TESTING, FORMAL METHODS, MASTERING COMPLEXITY, PETRI NETS, CLUSTERING, MODEL DESIGN, MODELING METHOD.

The purpose of the work is to investigate problems in the field of model-based testing, which prevents this technique widespreading, and to create a not demanding, intuitive modeling method that helps to work effectively with the complexity of models.

The research methodology is design science, much of which is a systematic literature review - a way of deliberately and purposefully finding, selecting, assessing, and aggregating scientific evidence and solutions to answer a particular research question in a transparent and reproducible way. Excerpts from the literature serve as the building blocks of the created modeling method.

ЗМІСТ

ВСТУП

Тестування на основі моделей (model-based testing – MBT) не дуже поширене, всупереч можливості автоматизувати генерацію тестових випадків шляхом їх формування з моделі системи, що тестується [1].

Мета дослідження — з'ясування прогалин та проблем в галузі MBT, які заважають поширенню цієї техніки, та створення рішення для основного чинника проблеми.

Об'єктом дослідження є складність наявних підходів та методів моделювання. Відповідно до недавнього огляду області MBT [2], із самого її початку вона мала декілька основних причин малої поширеності, основною із яких є складність. Зокрема, моделі, як правило, ускладнюються через їх розміри, геометричне зростання зв'язків між об'єктами моделі та відсутність правильного балансу між абстракцією та імплементацією. Інша причина складності — виразність, а саме: легкість уявлення, вивчення та використовування.

Дуже мало публікацій в області MBT підіймати питання процесу створення моделей [3]. Також недостатня систематичність чинних методів є одним із головних чинників нетривіальності процесу створення моделей. Ця нетривіальність виникає через те, що складність неможливо розрахувати, її можна лише опанувати [4].

Кінцевий продукт інструментів MBT — це набір автоматично сформованих тестових випадків, які отримані з моделі, яка описує деякі аспекти, найчастіше функціональні, системи, що перевіряється (system under testing - SUT). Створені моделі можуть зображати потрібну поведінку системи або використовуватись для створення тестових стратегій або тестових середовищ. Оскільки моделі зазвичай будуються на основі вимог або очікуваної поведінки, таке тестування зазвичай розглядають як одну з форм тестування чорної скриньки. Таким чином, MBT забезпечує добру основу для багаторазового

тестування системи / програмного забезпечення, максимально охоплює різні його дії та створює прямий зв’язок між вимогами та тестами.

Мережі Петрі [5] — це графічний формалізм для представлення одночасної поведінки багатьох паралельних станів за допомогою переходів (об’єктів дій), місць (об’єктів станів) та маркерів, що позначають відповідні місця для представлення виконання умов. Мережі Петрі - прекрасний інструмент для системного аналізу та візуалізації.

Після аналізу прогалин сформулювався основний інтерес цього дослідження — потенціал мереж Петрі для МВТ та практичному застосуванні цієї комбінації в автоматизованому тестуванні (АТ).

Актуальність та важливість АТ Уттінг підкреслив [1], вказавши на швидке зростання відповідного ринку. Організація автоматизованої підсистеми тестування дозволяє виявити велику різноманітність помилок, особливо що стосується функцій та дизайну системи на кожному етапі її розвитку. Ось чому інструменти для автоматизованого виконання тестів широко розповсюджені і є важливою частиною сучасного підходу до розробки програмного забезпечення для проєктів будь-якого масштабу, але переважно складних систем.

Основною задачею дослідження є розробка методу створення моделей та опанування їх складності через поєднання виразності мереж Петрі та стилю проєктування знизу-вгору.

Основною перевагою створеного методу у порівнянні з іншими методами моделювання є управління складністю моделей не вимогливим, інтуїтивним способом. Цей метод є авторською розробкою і вперше описується у даному дослідженні.

Методологіями дослідження є систематичний огляд літератури (СОЛ) та наука проєктування (Design Science), які детально окреслюються в тексті цієї роботи.

Матеріали проведеного дослідження будуть корисними для подальшого вивчення специфіки автоматизованого тестування та основою для програмного забезпечення, яке втілює зазначені ідеї.

1 МЕТОДОЛОГІЯ НАУКИ ПРОЄКТУВАННЯ

Наука проєктування — це методологія дослідження інформаційних технологій, яка пропонує конкретні вказівки щодо оцінки та ітерації в рамках дослідницьких проєктів.

Науково-дослідні дослідження в галузі дизайну зосереджені на розробці та виконанні (спроєктованих) артефактів з явним наміром покращити функціональні показники артефакту. Дослідження в галузі дизайну зазвичай застосовуються до категорій артефактів, включаючи алгоритми, методології проєктування (включаючи моделі технологій) та мови. Його застосування є найбільш помітним у дисциплінах інженерії та інформатики, однак не обмежується цим і може бути знайдено у багатьох дисциплінах та галузях [6]. У науково-дослідних дослідженнях дизайну або конструктивних дослідженнях [7], на відміну від роз'яснювальних наукових досліджень, цілі академічних досліджень мають більш прагматичний характер. Дослідження в цих дисциплінах можна розглядати як прагнення до розуміння та покращення працездатності людини [8]. Такі відомі дослідження такі установи, як медіа лабораторія MIT, Центр досліджень дизайну в Стенфорді, Інститут програмного забезпечення Карнегі-Меллона, PARC Xerox і Центр організації та системного дизайну Брунеля, застосовують підхід дослідження наукових досліджень в галузі дизайну [6].

За словами Ван Акена, головна мета наукових досліджень з дизайну — розробити знання, які професіонали відповідної дисципліни можуть використовувати для проєктування рішень для своїх проблем. Цю місію можна порівняти з тією з «роз'яснювальних наук», як природознавство та соціологія, яка полягає у формуванні знань для опису, пояснення та прогнозування [8]. Гевнер заявляє, що основною метою наукових досліджень у галузі дизайну є досягнення знань та розуміння проблемної галузі шляхом побудови та застосування розробленого артефакту [9].

З перших днів інформатики займаються дослідженнями дизайну, не знаючи цього. Вони розробили нові архітектури для комп’ютерів, нові мови програмування, нові компілятори, нові алгоритми, нові структури даних і файлів, нові моделі даних, нові системи управління базами даних тощо. Значна частина ранніх досліджень була зосереджена на підходах та методах розвитку систем. Панівна філософія дослідження полягала в розробці кумулятивних, заснованих на теорії досліджень, щоб можна було виписати рецепти. Схоже, що ця дослідницька стратегія «теорія з практичним наслідком» серйозно не дала результатів, які представляють реальний інтерес на практиці. Цей збій призвів до пошуку практичних методів дослідження, таких як дослідження в галузі дизайну [10].

Процес проєктування — це послідовність експертних заходів, яка виробляє інноваційний продукт (див. рис. 1).

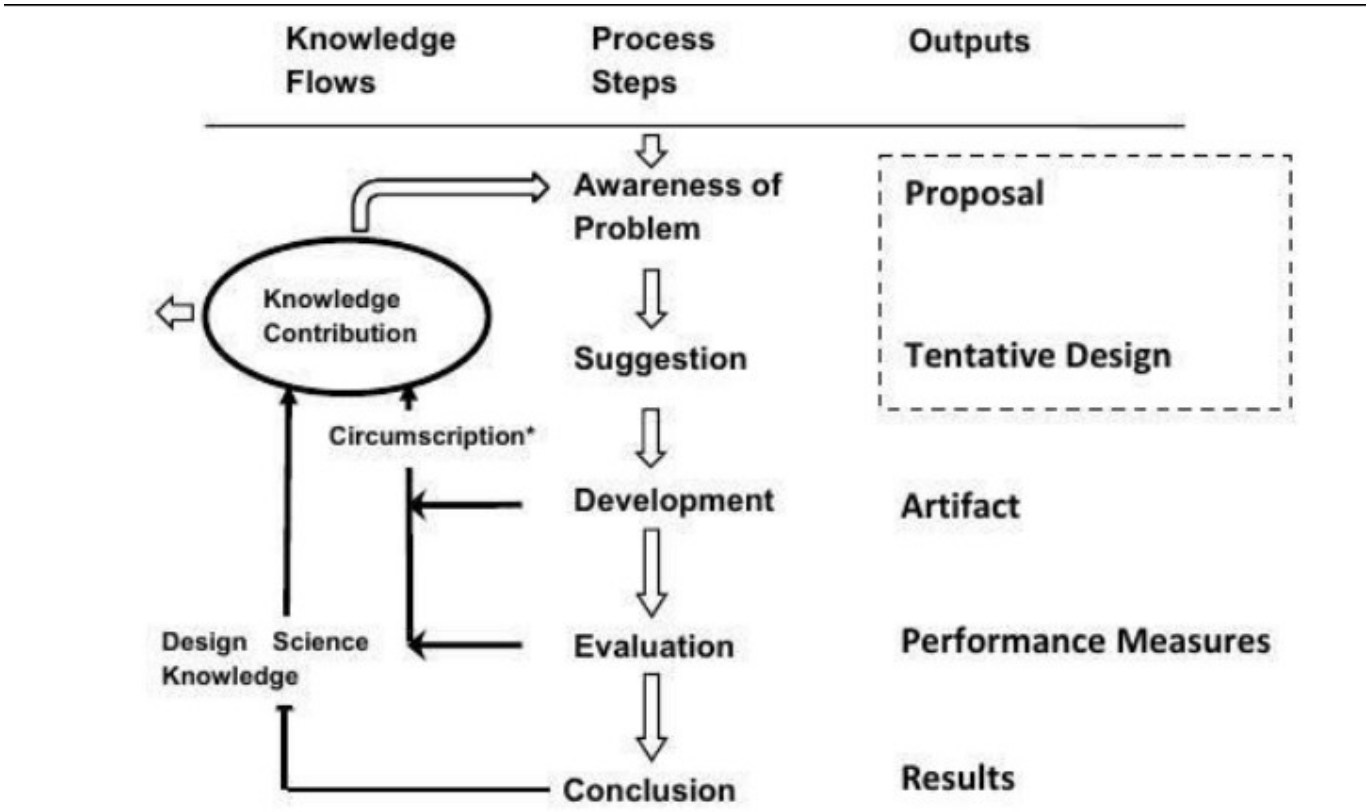

Рис. 1 — Процес методології науки проєктування

Артефакт дозволяє досліднику краще зрозуміти проблему; переоцінка проблеми покращує якість процесу проєктування тощо. Цей цикл складання та

оцінки зазвичай повторюється кілька разів до створення остаточного артефакту дизайну [11]. У науково-дослідних дослідженнях у галузі дизайну основна увага приділяється так званому випробуваному на місцях та обґрунтованому технологічному правилу як можливому продукту досліджень другого режиму, який може покращити актуальність академічних досліджень в управлінні. Виробництво знань першим способом є суто академічним та монодисциплінарним, а другим — мультидисциплінарним та спрямованим на вирішення складних та відповідних польових проблем [8].

Артефакти в у науці проектування сприймаються об'єкти, що містять знання. Ці знання варіюються від логіки проєктування, методів побудови та інструменту до припущень про контекст, в якому артефакт повинен функціонувати.

Створення та оцінка артефактів, таким чином, є важливою частиною процесу науки проєктування.

Артефакти можуть широко включати: моделі, методи, конструкції, інстанції та теорії дизайну, соціальні інновації, нові або раніше невідомі властивості технічних або інформаційних ресурсів, нові пояснювальні теорії, нові моделі проєктування та розробки та процеси та методи впровадження.

2 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

2.1 Класифікація формальних методів

Формальні методи — це математичні методи з використанням формальних мов, теорії логіки або автоматичних систем для розробки, уточнення та перевірки програмного забезпечення та апаратних систем. Перевірка — це процес математичного доведення або спростування правильності системи щодо формально заданих властивостей [12].

МВТ — це найлегший і практичний формальний метод. Далі наведено основні типи формальних методів у порядку від найскладніших до найлегших.

ФМ за допомогою дедуктивної перевірки. Мови їх специфікацій зазвичай дуже виразні, тобто здатні представляти системи з нескінченним простором стану, проте вони вимагають багато досвіду, часу та комп’ютерних ресурсів.

Перевірка моделей — те саме, що і дедуктивна перевірка, але лише для систем із кінцевою кількістю станів. Перевагами цього типу є достатня виразність та повністю автоматична повна верифікація системи. Значним недоліком є вимога до значного рівня експертиза та спроможності придумати хороші абстракції.

Обмежена перевірка моделі — це спеціальний випадок перевірки моделі, що досліджує простір стану лише на задану глибину. З позитивних особливостей можна визначити знайдення помилок, ніж доведення правильності системи. Всупереч тому, що вибух станів менш важкий, він все ще часто викликає проблеми.

Тестування на основі моделі автоматично генерує тестові випадки зі створеної моделі. Вибух станів рідко або ніколи не перешкоджає практичному застосуванню МВТ, проте повна перевірка системи зазвичай неможлива — залежить від доступних обчислювальних ресурсів.

Отже, кінцевий продукт інструментів МВТ — це набір автоматично сформованих тестових випадків, які отримані з моделі, яка описує деякі аспекти, найчастіше функціональні, системи, що перевіряється (system under test - SUT).

Створені моделі можуть зображати потрібну поведінку системи або використовуватись для створення тестових стратегій або тестових середовищ. Оскільки моделі зазвичай будуються на основі вимог або очікуваної поведінки, таке тестування зазвичай розглядають як одну з форм тестування чорної скриньки. Таким чином, МВТ забезпечує добру основу для багаторазового тестування системи / програмного забезпечення, максимально охоплює різні його дії та створює прямий зв’язок між вимогами та тестами.

Створення хороших тестових моделей є критично важливим для ефективного застосування МВТ. Дуже мало публікацій МВТ обговорювали, як будувати тестові моделі. Як правило, тестові моделі залежать не тільки від формалізму моделювання, але і від тестових цілей та вимог конкретних застосувань.

2.2 Підґрунтя дослідження та прогалини у галузі тестування

Відповідно до недавнього огляду області МВТ [2], основна дослідницька проблема мала наступні періоди:

Вихідними даними для побудови є один з варіантів:

– 2007-2010: емпіричні дані;

– 2010-2012: складність моделювання;

– 2014-2015: доступність підходів та інструментів для певних напрямків;

– 2015-сьогодення: ефективність та складність чинних підходів.

МВТ — це комплексний вид діяльності, який може бути спрощений шляхом зменшення його вимог до майстерності та необхідних знань. Це підвищить рівень техніко-економічної спроможності — і таким чином розповсюдженість. Складність сучасних підходів створення моделей є основною причиною приблизно у п’ятої частини (19%) усіх досліджених проблем. Автори також

повідомили, що чинних нотації моделей, які є основними складовими у процесі створення моделей, є одним із найскладніших аспектів МВТ.

Зокрема, моделі, як правило, ускладнюються через їх розміри [S04, S11], state explosion (геометричне зростання зв'язків між об'єктами моделі) [S01, S11] та відсутність правильного балансу між абстракцією та імплементацією [S20]. Інші причини складності — виразність [S04], а саме: легкість уявлення, вивчення та використовування [S03, S04, S11].

Що стосується мов моделювання, то UML нотації засновані на переходах домінують в області МВТ. Вірогідно через їх можливості візуалізації моделі та значним рівнем формальності. Однак бракує досліджень щодо підходів, що не належать до UML [S05, S08].

2.3 Питання та методологія огляду літератури

Маючи опис підґрунтя, було сформульовано основне питання цього дослідження: “Як опанувати складність моделей у МВТ?”

Ключем до відповіді на це питання може бути наступна думка Пірса: “Формальні методи ніколи не матимуть істотного впливу, поки їх не можуть використовувати люди, які їх не розуміють” [4]. Іншою стороною цієї проблеми є те, що дуже мало публікацій МВТ підіймали питання процесу створення моделей [5]. Також недостатня систематичність чинних методів є одним із головних чинників нетривіальності процесу створення моделей [13]. Таким чином, усе зазначене вище об'єдналося в ідею створення нового методу моделювання. Його мета — управління складністю не вимогливим, інтуїтивним способом.

Головне питання є дуже широким і розпливчастим, щоб мати будь-які конкретні уявлення з того, з чого почати. Тому для СОЛ воно було розділене на ще чотири питання по трьох фундаментальних частинах МВТ:

– як ми визначаємо і оцінюємо складність?

– якою мовою використовувати і чому?

– які особливості мови можуть допомогти в опануванні складністю?

– який стиль проєктування найкраще підходить для обраної мови?

Методологію дослідження є наука проєктування (Design Science) поєднана із систематичним оглядом літератури (СОЛ). СОЛ — це спосіб навмисного та цілеспрямованого пошуку, відбору, оцінки та узагальнення наукових доказів та рішень для відповіді на конкретне дослідницьке питання таким чином, що є прозорим і відтворюваним. Отже, ця чітко визначена методологія робить меншою ймовірність зіткнутись з упередженими літературними результатами. Особливо, враховуючи, що витяги із літератури будуть слугувати складовими блоками створюваного методу моделювання.

СОЛ було проведено з використанням настанов щодо систематичних оглядів для дослідників інженерії програмного забезпечення [14] та дослідження Кубігема [15]. Однак природа науки проєктування ітеративна та орієнтована на практичні рішення, і це змістило фокус СОЛ на відповідні джерела та циклічний пошук літератури.

2.4 Опис пошуку літератури

Процес пошуку повинен визначити первинні дослідження, які безпосередньо стосуються зазначених дослідницьких питань. У випадку цього СОЛ кожне запитання залежить від попереднього, крім Q1. Ця залежність випливає з ітеративного характеру науки проєктування — розроблюваний метод моделювання доповнюється по мірі появи нових питань та відповідей на них.

Пошуки проводилися власноруч, без автоматизації, та послідовно. Наявна література була досліджена для кожного дослідницького питання, коли висвітлювалося попереднє. У пошукових запитах були наступні загальні критерії та фільтри:

– мова: англійська;

– пошукові системи: OneSearch (Лінеус Університет, 122 бази даних [16]), Google Академія, Пошук Google;

– географічний обсяг: у всьому світі;

– пошукові терміни застосовувалися лише до заголовків;

– роки публікації матеріалів: 2000-2020.

Використовувалися як академічні, так і неакадемічні бази даних: практична сторона МВТ часто не помічається в наукових публікаціях. Як доповнення до академічної літератури, додаткові джерела (наприклад, вебсторінки, звіти, таблиці) використовувались для широкого висвітлення наявних знань з МВТ та конкретних питань для СОЛ.

Основна ідея кожного пошуку полягала в тому, щоб створити найбільш релевантний список статей, який слід прочитати. Наступним кроком стратегії було знайти і зосередити увагу на одному документі з найкращим змістом щодо відповідного аспекту процесу моделювання.

Наступна оцінка якості: чи містить документ розв'язання поточної проблеми без конфліктів та максимальною синергією з попередніми відповідями-рішеннями?

Було випробувано найрізноманітніші пошукові терміни, щоб знайти максимально релевантні. Результати наведено у додатку А.

2.5 Відповідність використаних методологій та розділів

Отримані статті мають різні типи та потребують якісного аналізу та видобутку концепцій і рішень. Згідно [17], вилучення якісних доказів є циклічним процесом. По мірі появи нових ідей, тем та питань природно може виникнути потреба перемикнутися між читанням знайдених досліджень, вилученням даних та їх інтерпретацією, щоб отримати послідовний результат без розбіжностей.

Пошук моделей, зв'язків та з'ясування оптимальних способів комбінування речей — саме цим займається наука проєктування.

Наступні розділи відповідають етапам дослідження та питанням цього систематичного огляду літератури: оцінка складності, вибір мови та її особливості, а також стиль проєктування моделей.

Результати методології науки проєктування — створений метод та отриманні знання стосовно області тестування на основі моделей та усіх супутніх тем — представлені у відповідному розділі, що закінчує пояснювальну записку.

3 ВИМІРЮВАННЯ СКЛАДНОСТІ

3.1 Спрощення та розуміння

Моделі, як і всі системи, як правило, ускладнюються. Для того, щоб оцінити складність моделі, її потрібно певним чином виміряти.

Згідно [4], існує два основні способи наближення до складності: кількісний та якісний.

Кількісний аналіз має тенденцію до спрощення, що гарантує збій при самостійному використанні. У використанні простих речей і спробах зробити їх простими немає проблем, але це слід робити дуже обережно, щоб уникнути ігнорування внутрішньої сутності складності — закономірностей організації. Аналогічно зменшенню без синтезу, кількісний аналіз без якісного аналізу недостатній і фактично не може виміряти складність.

Якісний аналіз забезпечує розуміння замість спрощення, що є кращим способом вирішення складності. Розуміння приймає складність того, що воно є, спостерігає за ним, а потім використовує знайдені внутрішні якості як будівельні блоки для подальшого вивчення та аналізу.

Коли кількісний та якісний аналіз використовуються разом, вони врівноважують один одного і допомагають дизайнерам моделей не вводити в оману суб'єктивність чи псевдооб'єктивність.

3.2 Кількісний аналіз

З наукової точки зору найбільш природний підхід — це кількісний аналіз. Попередній розділ міг би зробити так, що вимірювати складність з математичною точністю неможливо. Однак у контексті створення моделей з використанням мови математичного моделювання ця мета стає частково здійсненною. Оскільки кожна

модель складається з компонентів та відносин між цими компонентами, складність може розглядатися математично як кількість та різноманітність цих двох. Як Вестербі в [4] зазначає, що “все, що збільшує ці фактори, збільшує складність”. Такий кількісний аналіз може бути наданий, використовуючи шість основних величин як основи, які показані у вигляді перетину рядків та стовпців у таблиці 1.1.

Таблиця 1.1 – Шість основних величин для вимірювання складності

Кількість:	Компонентів	Зв'язків
Загалом		
Різних типів об'єктів		
Кожного типу		

Ці шість не можна нехтувати та утворюють ядро для розуміння кількісних аспектів складності. Звичайно, його можна розширити, щоб зробити вимірювання більш детально.

3.3 Якісний аналіз

Кількість та різноманітність компонентів та взаємозв'язків разом утворюють модель організації. Ці загальні шаблони можна знайти на різних рівнях системи та за різних обставин. Їх часто називають ізоморфізмами, загальними системними принципами або аналогами. Поєднання цих шаблонів створює організаційну складність. А розуміння того, як вони впливають на природу системи, забезпечує широке та глибоке розуміння структури, процесів та джерела складності. У двох словах, якісний аналіз — це визнання та розуміння цих моделей організації в різних формах. Результатом цього типу аналізу повинні бути засоби для розуму, які корисні для пошуків освоєння складності моделі.

4 ПОРІВНЯННЯ ТА ВИБІР МОВИ МОДЕЛЮВАННЯ

4.1 Приблизність моделей та специфічність тестів

«Programs must be written for people to read, and only incidentally for machines to execute» - так писав Гарольд Абельсон у книзі «Структура та інтерпретація комп'ютерних програм».

Як було зазначено [18], основні мови моделювання в області MBT є UML та FSM. Вони є візуальними інструментами для опису програмної системи.

Основна проблема як UML, так і FSM полягає в тому, що вони є дуже специфічними інструментами моделювання. Це не проблема сама по собі, але природа моделювання полягає в описі речей приблизно, а не конкретно.

Коли ви хочете описати такі поняття високого рівня [19], як архітектура системи, ви, природньо, використовуєте дошку з простими фігурами та стрілками для опису програмної системи.

Звичайно, це не дуже конкретно. Але в цьому і справа. Карта - це не територія. Ви не хочете малювати все, а лише релевантні речі. Специфіка - не мета, коли ви описуєте поняття високого рівня, ви робите карту.

Якщо вам дійсно потрібна конкретність, для цього ми маємо комп'ютерний код. Комп'ютерні мови дозволяють описати, як саме конкретні речі працюють. Проте загальні речі не повинні описуватися дуже специфічними інструментами та мовами. Це просто неправильне використання.

Це те, чого багато людей не розуміють. Комп'ютери не «розуміють» так званих мов програмування високого рівня. Код, написаний у них, перекладається на машинний код, який для людей є нерозбірливим. Отже, мови програмування існують для людського спілкування між собою.

Специфікації програмного забезпечення, написані на природних контекстно-чутливих мовах, не працюють. Чому? Ця чутливість контексту робить опис розпливчастим і грубим, а іноді твердження навіть можуть бути взаємозаперечними.

А специфікація — це документ, який на словах описує те, що неможливо візуалізувати чи пояснити на кресленні чи в моделі. Це стосується не лише будівництва; ті ж принципи можуть бути застосовані у всіх галузях промисловості, від аерокосмічної, нафтогазової та автомобільної промисловості до виробничої.

Чому візуалізація погано працює для мов програмування та хороша для моделювання? Вважається, що візуальні блоки та конструкції ефективні для опису складних взаємозв'язків між компонентами системи. Але це не так. Подібні об'єкти дуже добре описують грубі ідеї, а не конкретні реалізації.

Головне зрозуміти — грубий опис також може бути корисним. Але тільки тому, що природні контекстно-чутливі мови та візуалізація добре описують поняття реальності, ми використовуємо їх для опису понять у конкретному вигляді. І це працює не дуже добре.

Наприклад, щоб описати такі речі, як фізика, нам знадобиться мова, яка дуже добре адаптована при описі фізики — математика. Щоб спеціально описати програмну систему, нам потрібна мова, призначена для виконання саме цього — мови програмування.

Через це специфікації не працюють на практиці. Якщо специфікація могла б ідеально описати всю програмну систему, це специфікація була б кодом. Код — це ідеальний опис програмної системи. Це територія, а не мапа.

UML та інші подібні мови як інструменти використовуються для того, щоб дати огляд і безліч деталей одночасно. Такий підхід створює інструмент, який не підходить ні для одного з варіантів.

Для того, щоб абстракція працювала, вона повинна позбутися багато деталей і бути дуже відокремленою від фактичної системи або об'єкта. Будь-яка абстракція чи модель — це карти якоїсь території, вони повинні бути «неправильними».

Для MBT метою моделі є генерування тестових випадків. З іншого боку, модель може бути хорошою специфікацією з оглядом системи.

Пріоритет повинен бути наступним: знайти найбільш зручні інструменти та

підходи для побудови моделі, але якщо це можливо, ми повинні скласти модель, що описує глобальну поведінку системи.

Треба зосередитись на одній цілі, тому що зазвичай, переслідуючи дві цілі одночасно, це призводить до посередності в обох. Щоб згенерувати тести, модель повинна бути дуже специфічною, що своєю чергою означає жертву в спроможності оглянути систему.

Але, оскільки мови програмування можна перекладати на машинну мову, повинен існувати спосіб поєднання огляду та конкретності для тестових випадків, коли це необхідно.

4.2 Пошук найменш складної мови

Стандартна кінцеві автомати (finite state machine - FSM) містять лише один поточний стан. Тоді як в мережах Петрі місця, більш-менш схожі на стани в FSM, можуть містити один або більше токенів. FSM є одно поточними, тоді як мережа Петрі є багато поточною. У FSM активний стан змінюється у відповідь на подію. У мережі Петрі переходи виконуються, як тільки всі вхідні місця містять принаймні один маркер.

Перевага FSM — інженери програмного забезпечення, ймовірно, більше знайомі з ними, і вони мають більше інструментів для роботи. Але загальна рекомендація — використовувати FSM, якщо система, що моделюється, є одно поточною. Рекомендація щодо сіток Петрі полягає в тому, що їх слід використовувати лише тоді, коли потрібна одночасність або додаткова виразність. А виразність — це те, що може допомогти зменшити складність ще більше.

Мережі Петрі можна використовувати для моделювання, візуалізації запущених, масивних одночасних систем, таких як мікросервіс архітектури або системи контейнерно-оркестрового обслуговування, що працюють на Kubernetes, Microsoft Azure або AWS Lambda.

Крім того, існує більше теоретичних досліджень про мережі Петрі, ніж про FSM. Також FSM можна привести до мереж Петрі.

FSM могли би підійти під критерії складності, проте цей формалізм породжує занадто багато дубльованих дуг для функцій. Також проблемою є те, що недостатньо інструментів для ієрархічної організації субмоделей — абстракції.

UML занадто специфічний і містить багато зайвих деталей. Також різні діаграми використовуються для різних рівнів абстракції, що ускладнює моделювання. Наприклад, діаграми послідовності, класів, компонентів.

Висновок в дослідженні [20], що діаграми діяльності UML як мови моделювання дуже схожі на мережі Петрі. Однак теоретична основа діаграм діяльності навіть не наближається до рівня досліджень щодо сіток Петрі. Автори роблять висновок, що семантика сітки Петрі краще підходить для моделювання закритих та активних систем, тоді як діаграми активності UML протилежні і набагато краще підходять до відкритих та реактивних систем. Беручи до уваги характер дизайну знизу вгору на основі бібліотеки зразків, цей момент є серйозним плюсом.

Мережі Петрі використовує лише 3 компоненти — місця, переходи, дуги — і створює менше відносин внаслідок уніфікованої кластеризації. Він також дуже здатний до абстракції, оскільки нехтує непотрібними відносинами під час підйому на рівень вище.

Паралельність, як особливість формалізму може бути використана на користь моделі. Наприклад, тестування продуктивності системи вимагає паралельності моделі, оскільки воно визначається кількістю одночасно підключених користувачів, часом обчислень, а також пропускною здатністю.

Найголовніша особливість цієї мови моделювання — це кластеризація, яка допомагає поєднати ієрархічність, абстракцію та математичну суворість за допомогою морфізмів кластерів — субмоделей [21].

Іншою особливістю є повторне використання, а саме — шаблони [22]. Існує значна академічна основа та дослідження щодо закономірностей у мережах Петрі.

5 СТИЛЬ ПРОЕКТУВАННЯ МОДЕЛЕЙ

Цей розділ йде після розділу про мову, оскільки мова не визначає повністю, але значно впливає на дизайн та спосіб підходу до проблеми.

Використання кластеризації робить не інтуїтивно зрозумілим моделювання дизайну «зверху вниз». Підхід «знизу вгору» набагато кращий через його природність при використанні кластеризації.

Моделюючи навіть невелику, але систему, що зростає, стає зрозуміло, наскільки важливо не лише зробити так, щоб моделі представляли вимоги, але і спроєктувати їх добре організовано. Попри експресивну силу будь-якої мови моделювання, відносини між сутностями роблять моделі великими та неконтрольованими [21], що означає складні. Продуманий дизайн моделей стає необхідним. Єдина відмінність моделювання малих і великих систем полягає в тому, що для великих «коефіцієнт смертності» стає очевидним, коли моделі не мають структури. Складність, як правило, зростає набагато швидше, ніж розміри моделей. І якщо ви не подбаєте про це заздалегідь, то досить швидко настає момент, коли ви втрачаєте контроль над моделлю через її складність. Це стає важко зрозуміти, важко перевірити і важко виправити. Правильна конструкція моделей економить багато сил, часу та грошей. Часто він зазвичай визначає, чи був процес моделювання марною витратою цих ресурсів чи давав очікувані переваги.

Отже, як у багатьох інших областях, попри різноманітність критеріїв, головне завдання проєктування тестової моделі для МВТ — зменшити складність. Відповідно до цього принципу модель повинна бути поділена на шматки, і чим більша модель, тим більше її потрібно розділити. Як слід проводити процес поділу?

Процес створення моделі є актом дослідження і відкриття, адже ми дізнаємося в чому саме ми потребуємо, оскільки самі формуємо матеріал. Це означає, що ми не зобов'язані мати повне уявлення про модельовану систему з

самого початку (камінь в город спроб написати повну специфікацію з першого разу). Ми можемо поступово осягати цю систему в процесі моделювання. Ми досліджуємо систему, створюючи її модель [23], і ця модель не зобов'язана повною, перш ніж можна буде почати нею користуватися. Моделювання є постійним експериментом, щоб подивитися як поводиться система і чи правильні наші припущення.

Під час дослідження було зроблено наступні спостереження стосовно стилів проєктування:

– начебто прості декомпозиції можуть стати дуже складними, особливо при спробі створення узагальнень;

– майже все пов'язано, тому що об'єкти природним чином ускладнюються;

– підхід зверху вниз ставить функціональне вище за структурне, що в майбутньому часто призводить до катастрофічних наслідків;

– дизайн зверху вниз може зайняти багато часу і призведе до масивного роздуття моделі;

– це типова практика як в академічних, так і у бізнес-середовищах.

5.1 Низхідне моделювання

Традиційний і добре описаний [24] підхід називається проєктування зверху вниз або поділом і правилом. Це стосується стилю побудови системи, починаючи з опису на високому рівні того, що вона повинна робити. Цей опис стає специфікацією вимог, яка служить основою для наступного процесу моделювання. Це досягається шляхом розбиття специфікації на все простіші фрагменти, поки не буде досягнуто рівня, що відповідає примітивам мови моделювання, що використовується. Потім проводиться процес моделювання зверху вниз. Наприклад, метою SUT є виконання семи операцій, тому її функціональність розділена на сім основних підмоделей. Перші підмоделі мають

зробити ще чотири операції, тож своєю чергою вони матимуть чотири власні підмоделі. Подібно до написання специфікації, цей процес триває до тих пір, поки вся модель не отримає належного рівня деталізації — кожна частина є достатньо великою, щоб зробити щось істотне, але достатньо мало, щоб бути розуміється як єдине ціле.

Після завершення всіх різних моделей SUT готовий до тестування. Це один з основних недоліків підходу моделювання зверху вниз. Жоден тестовий випадок не може бути створений до самого пізнього моменту, окрім підходів, що дозволяють заглушити або макети. Звичайно, це можливо лише в тому випадку, якщо мова моделювання це підтримує. Крім того, моделювання зверху вниз має тенденцію до «форми» модулів, які є дуже специфічними для SUT, тому не дуже використовуються. Основним недоліком є те, що всі рішення, прийняті з початку проєкт, безпосередньо або опосередковано залежать від високоточного специфікації SUT. Загальновідомий факт, що технічні характеристики мають тенденцію змінюватися з часом. Коли це станеться, існує великий ризик, що великі частини моделі потрібно буде переробити.

Моделювання зверху вниз має тенденцію створювати модулі, засновані на функціональності. Зазвичай специфікація функціональності штатів SUT. Це створює ще один великий ризик, що деталі щодо впровадження багатьох ресурси (структури даних для програмного забезпечення) повинні розподілятися між модулями та, таким чином, піддаватися глобальному впливу. Іншим модулям стає спокусливо використовувати ці деталі реалізації, створюючи тим самим небажані залежності між різними частинами моделі.

Підхід зверху-вниз асоціюється з декомпозицією, ієрархічністю на функціях.

5.2 Цикл взаємного розвитку

Шаблони — це гіпотези, а моделювання — це експеримент. Процес створення моделі — це акт дослідження та відкриття, адже ми з’ясовуємо саме те, що нам потрібно, тому що самі формуємо матеріал.

Це означає, що нам не потрібно мати повне уявлення про змодельовану систему або процес з самого початку. Ми можемо поступово осягати цю систему в процесі моделювання. Ми досліджуємо систему, створюючи її модель, і ця модель не повинна бути повною, перш ніж ви зможете її використовувати.

На рисунку 2 показано графічне зображення процесу MBT і де в ньому відбуваються дві «цикли взаємного розвитку», які розглянуті далі.

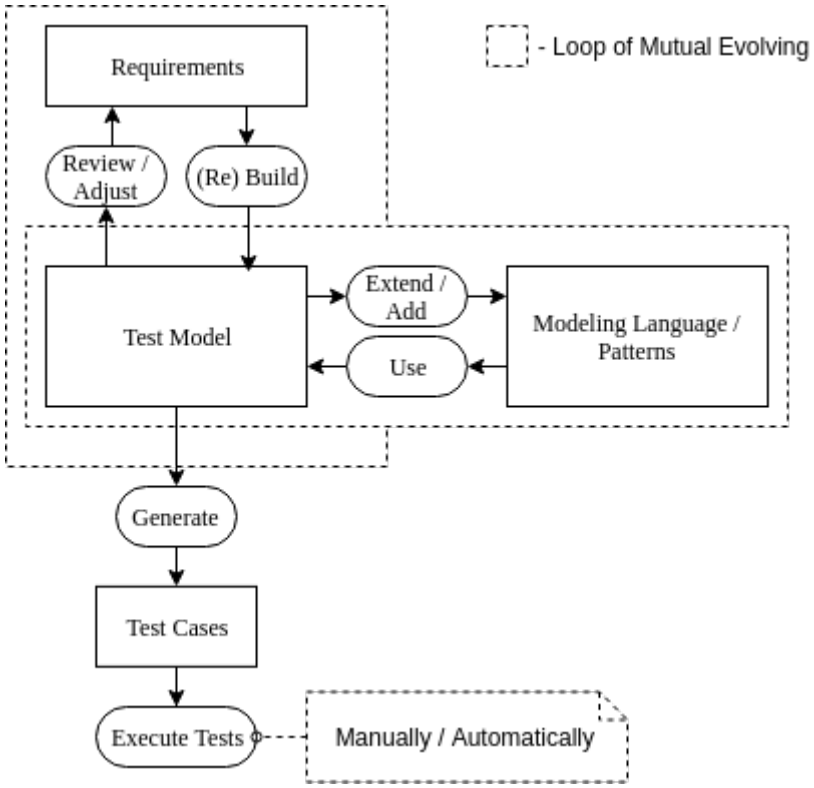

Рис. 2 — Цикли взаємного розвитку у процессі MBT

Моделювання — циклічний процес. Водночас знання про досліджуваний об'єкт розширюються та вдосконалюються, а початкова модель поступово

вдосконалюється. Дефіцити, виявлені після першого циклу через малі знання про об'єкт або помилки в побудові моделі, можна виправити в наступних циклах.

Основні вимоги ретельно переглянуто, створивши тестові моделі. Цей процес перегляду створює «цикл взаємного розвитку» (LME), що є основною ідеєю запропонованого в цьому документі методу.

Так само, між мовою та моделлю є ще один цикл LME. Модель — це не просто побудова до мови, а коли мова також побудована до моделі. Під час створення моделі може з’явитися новий шаблон для мови. Згодом ви розумієте, що використання нового шаблону спростить дизайн іншої частини моделі. Мова та модель розвиваються разом. Так само, як і модель та вимоги. Як і межа між двома ворожими державами, межа між мовою та моделлю промальовується та перекреслюються, доки в кінцевому підсумку вона не зупиниться уздовж гір та річок, природних кордонів вимог. Врешті-решт модель виглядатиме так, ніби мова та її візерунки були розроблені для неї. І коли мова і модель добре підходять один одному, ви отримуєте набір примітивів, який зрозумілий, маленький та ефективний.

5.3 Висхідне моделювання

Протилежністю підходу зверху вниз є знизу вгору. Це стосується стилю побудови системи, починаючи з базового опису SUT. Цей опис отримує окремі базові блоки, які потім відображаються на примітиви мови моделювання. Згодом ці примітиви пов'язані між собою, утворюючи більші підмоделі, які потім, своєю чергою, пов'язані, іноді на багатьох рівнях. Це стратегія, коли початки невеликі, але з часом ростуть у повноті.

Варто підкреслити, що дизайн знизу вгору не означає просто будувати одну і ту ж модель в іншому порядку. Робота знизу вгору зазвичай закінчується іншою моделлю. Замість єдиної, монолітної моделі ви отримаєте розширену мову з більш

абстрактними візерунками, а на них будується менша модель.

Ви можете посилити ефект потужної мови, використовуючи стиль знизу вгору, коли ви будуєте модель на багатьох рівнях, а нижчі рівні виступають як своєрідна мова для вищих рівнів. Якщо ви все зробили правильно, вам потрібно мати на увазі лише найвищий рівень.

Одним з головних недоліків підходу «знизу вгору» є те, що він вимагає великого досвіду для визначення меж шаблону чи модуля. Але моделювання знизу вгору має ряд переваг перед підходом зверху вниз.

Створення тестових випадків спрощено, оскільки жодні заглушки не потрібні. Отже, ви можете «швидко провалитись» і відновити те, що було зроблено, якщо це потрібно.

Сформовані за схемою «знизу вгору», як правило, більш загальні та, отже, більш багаторазові, ніж структури, що будуються зверху вниз. Можна стверджувати, що метою моделювання знизу вгору є створення побудови мови, що залежить від моделі, на основі шаблонів. Щодня моделюючи, багато моделей, написаних для першої моделі, також будуть корисними для наступних моделей. Після того, як буде придбано великий субстрат візерунків, написання нової моделі може зайняти лише частину зусиль, які знадобляться, якби потрібно було починати з сирої мови. Цей факт значно спрощує технічне обслуговування, зокрема додаючи нові функції до моделі. Це також робить можна відкласти остаточне рішення щодо точної структури моделі.

Змушуючи мову робити більшу роботу, дизайн знизу вгору дає моделі, які є меншими та гнучкішими. Менша модель не повинна ділитися на стільки компонентів, а менше компонентів означає моделі, які легше читати чи змінювати. Менше компонентів також означає меншу кількість зв’язків між компонентами, і, таким чином, менше шансів на помилки. Оскільки промислові дизайнери прагнуть зменшити кількість рухомих деталей у машині, модельєри, які використовують конструкцію знизу вгору, прагнуть зменшити розмір та складність своїх моделей.

Дизайн знизу полегшує розуміння моделей через загальний характер

моделей.

Оскільки це завжди означає бути уважним до огляду моделей в моделі, робота знизу вгору допомагає уточнити ідеї щодо дизайну моделі. Якщо дві віддалені компоненти моделі схожі за формою, подібність буде легко помітити та, можливо, це спровокує перепроєктування моделі простішим способом.

Цей стиль краще підходить моделям, які можуть бути написані невеликими групами або особами. Однак водночас це розширює межі того, що можуть зробити модельєри. У [25] Фредерік Брукс запропонував, щоб продуктивність групи не зростала лінійно з її розмірами. Зі збільшенням чисельності групи продуктивність індивідуального програміста знижується. Така ж ситуація трапляється з командою тестувальників або окремим спеціалістом.

Підхід знизу-вгору асоціюється із композицією, ієрархічністю на даних.

6 МЕТОД МОДЕЛЮВАННЯ

Результатом цього дослідження є метод створення моделей, який має наступні кроки:

а) Встановіть значення критеріїв складності: визначте максимальне значення для кількості типів та кількості в межах типів для компонентів (місця-властивості та переходи-функції) та відношень (дуг);

б) Створіть компонентну мережу (кластер);

1) Вкажіть окремі сукупності даних та їх властивості, стани;

2) Перенесіть властивості на місця мережі Петрі;

3) (ОПЦІОНАЛЬНО) Перевірте, чи є відповідний шаблон, і якщо результат його використання підходить, то переходимо до іншої (під) моделі, частини поточного шару;

4) Вкажіть взаємодії (дуги) між властивостями, станами;

5) Вкажіть або використайте вже вказаний перехід для кожної дуги між двома місцями.

6) Класифікуйте всі рівні створеної мережі Петрі знизу вгору способом аналогічним алгоритму пошуку в ширину.

в) (ОПЦІОНАЛЬНО) Додайте компонентну мережу до бібліотеки шаблонів.

г) Порівняйте результати та встановлені значення критеріїв складності для моделі.

ґ) Повторюйте починаючи із кроку 1, поки не буде створено модель усієї системи.

Приклад використання цього методу зображено на рисунку 3.

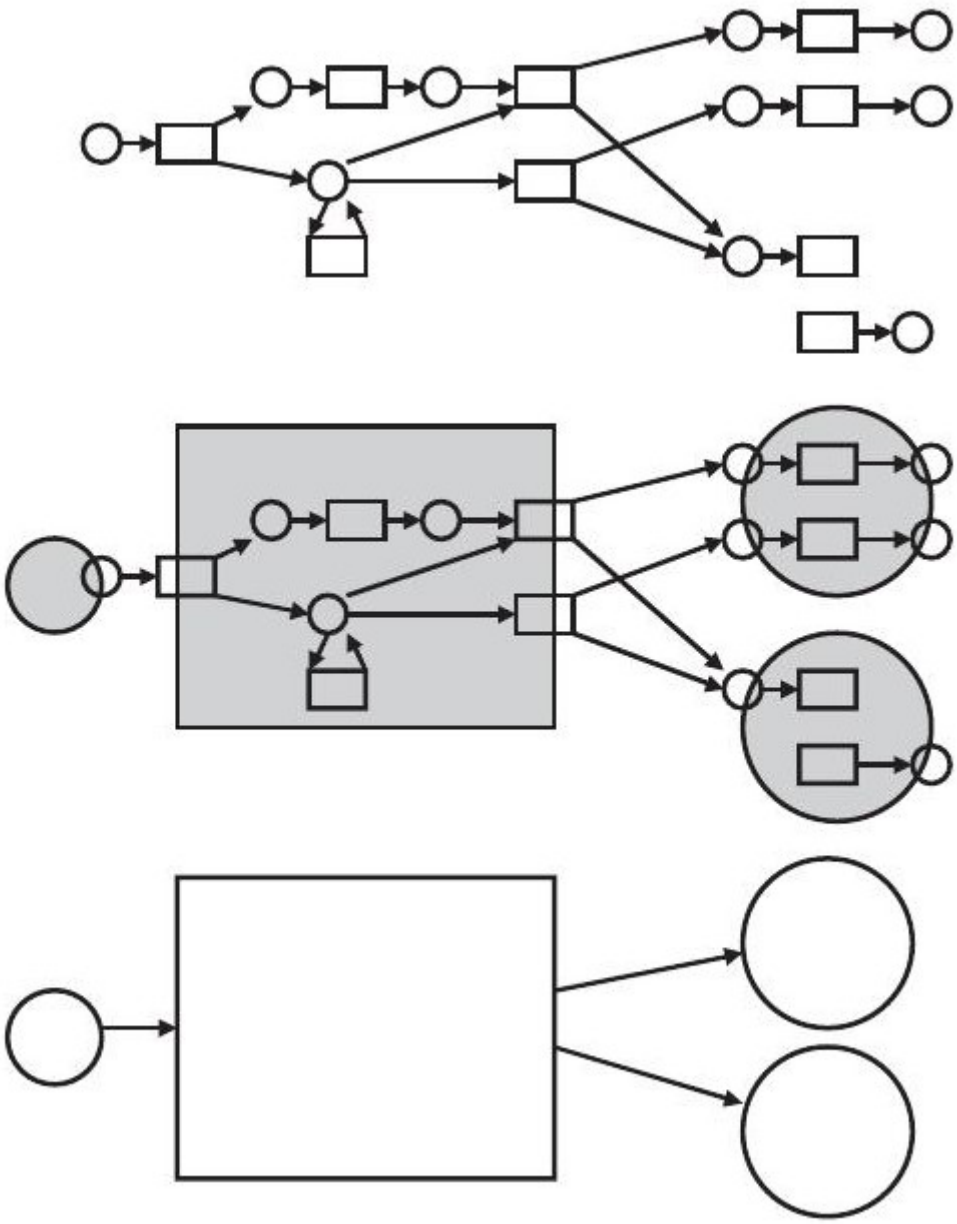

Рис. 3 — Приклад використання методу моделювання

Артефакти в у науці проектування сприймаються об'єкти, що містять знання. І процес створення цього методу показує, що складність — це основна проблема тестування на основі моделей, потенціал мереж Петрі може бути розкритий через створення відповідних програмних інструментів.

ВИСНОВКИ

В ході виконання атестаційної роботи було зроблено аналіз предметної галузі через класифікування формальних методів, аналіз прогалин в області тестування на основі моделей, пошук необхідної літератури та опис цього процесу.

Було знайдено найбільш узагальнений підхід до оцінки та вимірювання складності, що потім був використаний для порівняння графічних мов моделювання.

Було описано методологію науки проєктування, що дозволяє поєднати науковий підхід та творчі розробки у єдину роботу.

Через порівняння мов моделювання було встановлено, що мережі Петрі — найкраща опція для якісного та довгострокового інтегрування тестування на основі моделей через використання кластеризації, можливості створювати шаблони субмоделей для багаторазового використання, найбільш узагальнену структуру мови, яку легко опанувати навіть людині, що ніколи не вивчала мови моделювання.

Детально було розглянуто два несумісних стилі проєктування моделей — низхідний та висхідний. Було встановлено, що для використання мереж Петрі необхідно дотримуватися стилю знизу-вгору, через особливості процесу кластеризації та знайдений цикл взаємного розвитку.

Результатом дослідження є створений метод, який є інтуїтивно зрозумілим, простим у використанні, та об'єднує усі знайдені під час дослідження способи зменшити складність в усіх її означених проявах — від мови, до стилю проєктування.

Отримані знання та результати є корисними для подальшого розвитку досліджень в області тестування на основі моделей, автоматизованого тестування, та графічних мов моделювання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. M. Utting and B. Legeard, Practical Model-Based Testing: A Tools Approach, Elsevier Inc., Preface, 2007.

2. Villalobos-Arias, Leonardo, et al. "Model-based testing areas, tools and challenges: A tertiary study." CLEI Electronic Journal 22.1 (2019). [S-number] - sources cited in [2]

3. Dianxiang Xu, M., Weifeng Xu, Kent, Thomas, and Linzhang Wang. "An Automated Test Generation Technique for Software Quality Assurance." IEEE Transactions on Reliability 64.1 (2015): 247-68. Web.

4. Vesterby, Vincent. "Measuring complexity: Things that go wrong and how to get it right." Emergence: Complexity and Organization 10.2 (2008): 90-102.

5. Murata, T. "Petri Nets: Properties, Analysis and Applications." Proceedings of the IEEE 77.4 (1989): 541-580. Web.

6. Hevner, Alan, and Samir Chatterjee. "Design science research in information systems." Design research in information systems. Springer, Boston, MA, 2010. 9-22.

7. Dresch, Aline, Daniel Pacheco Lacerda, and José Antônio Valle Antunes. "Design science research." Design Science Research. Springer, Cham, 2015. 67-102.

8. Van Aken, Joan Ernst. "Management research as a design science: Articulating the research products of mode 2 knowledge production in management." British journal of management 16.1 (2005): 19-36.

9. Hevner, Alan R., et al. "Design science in information systems research." MIS quarterly (2004): 75-105.

10. Iivari, J. "A paradigmatic analysis of Information Systems as a design science, forthcoming in Scandinavian Journal of Information Systems." Draft 27p, (2007).

11. Hevner, Alan R. "A three cycle view of design science research." Scandinavian journal of information systems 19.2 (2007): 4.

12. Broy, Manfred, et al. "Model-based testing of reactive systems." Volume 3472 of Springer LNCS. 2015.

13. Siavashi, Faezeh, and Dragos Truscan. "Environment modeling in model-based testing: concepts, prospects and research challenges: a systematic literature review." Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering. ACM, 2015.

14. Kitchenham, Barbara. "Procedures for performing systematic reviews." Keele, UK, Keele University 33.2004 (2004): 1-26.

15. Kitchenham, Barbara, et al. "Systematic literature reviews in software engineering–a systematic literature review." Information and software technology 51.1 (2009): 7-15.

16. Linnaeus University Library. (2019) Articles and databases. [Online]. Available: https://lnu.se/en/library/search-and-evaluate/articles-and-databases/

17. Noyes J & Lewin S. Chapter 5: Extracting qualitative evidence. In: Noyes J, Booth A, Hannes K, Harden A, Harris J, Lewin S, Lockwood C (editors), Supplementary Guidance for Inclusion of Qualitative Research in Cochrane Systematic Reviews of Interventions. Version 1 (updated August 2011). Cochrane Collaboration Qualitative Methods Group, 2011. Available from URL http://cqrmg.cochrane.org/supplemental-handbook-guidance

18. Zoltán Micskei. (2017) Model-based testing overview, tools and projects. Department of Measurement and Information Systems Budapest University of Technology and Economics. [Online]. Available: http://mit.bme.hu/~micskeiz/pages/modelbased_testing.html

19. Boccara, Nino. Modeling complex systems. Springer Science & Business Media, 2010.

20. Mussa, M., S. Ouchani, W. Al Sammane, and A. Hamou-Lhadj. "A Survey of Model-Driven Testing Techniques." 2009 Ninth International Conference on Quality Software (2009): 167-72. Web.

21. Keller, Walter. "Clustering for Petri Nets." Theoretical Computer Science 308.1-3 (2003): 145-97. Web.

22. Van der Aalst, Wil MP, Christian Stahl, and Michael Westergaard. "Strategies for modeling complex processes using colored petri nets." Transactions on petri nets and other models of concurrency vii. Springer, Berlin, Heidelberg, 2013. 6-55.

23. Smelyakov, Kirill, et al. "Investigation of network infrastructure control parameters for effective intellectual analysis." 2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET). IEEE, 2018.

24. Ahuja, J. S., and K. P. Valavanis. "A hierarchical modeling methodology for flexible manufacturing systems using extended Petri nets." [Proceedings] 1988 International Conference on Computer Integrated Manufacturing. IEEE, 1988.

25. Brooks Jr, Frederick P. The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition, 2/E. Pearson Education India, 1995.