МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 04.00.00.000 ПЗ

Група ІПм-23-1

Арламовський Руслан

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Арламовський Руслан Ігорович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

МАГІСТЕРСЬКА РОБОТА

Методології функціональної безпеки програмних додатків

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Арламовський Р.І.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Бандура Вікторія Валеріївна, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Нормоконтроль

доц. Вовк Р.Б.

(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ 04 ” вересня 2024 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Ардамовському Руслану Ігоровичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “ Методології функціональної безпеки програмних додатків”

керівник проекту (роботи) Бандура Вікторія Валеріївна, к.т.н., доцент

затверджені наказом закладу вищої освіти від “ 18 ” листопада 2024 р. № /7

2. Строк подання студентом проекту (роботи) 15 грудня 2024 р.

3. Вихідні дані до проекту (роботи) Теоретичні концепції та формальні моделі побудови та функціонування інформаційних та програмних технологій функціональної безпеки

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Дослідження та аналіз предметної області функціональної безпеки
2. Дослідження методології інженерії на основі моделі в контексті функціональної безпеки
3. Функціональна безпека на основі моделі
4. Імплементація моделей для реалізації методології функціональної безпеки ПЗ

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Процес генерації коду платформи Motar (рис. 1.1)
2. Співвідношення між MBSE та подібними методологіями (рис. 1.2)
3. MBSE V&V-модель (рис. 1.3)
4. Життєвий цикл згідно ISO 26262 (рис. 1.4)
5. Порівняння робочого процесу між MBD/MQO та ISO 26262 (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Перевірка на плагіат	доц., к.т.н. Вовк Р.Б.	

7. Дата видачі завдання 04 вересня 2024 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури по темі магістерської роботи	15.09.2024	виконано
2	Аналіз концепцій та алгоритмів предметної області	29.09.2024	виконано
3	Дослідження та аналіз предметної області функціональної безпеки	15.10.2024	виконано
4	Дослідження методології інженерії на основі моделі в контексті функціональної безпеки	08.11.2024	виконано
5	Функціональна безпека на основі моделі	20.11.2024	виконано
6	Імплементація моделей для реалізації методології функціональної безпеки ПЗ	01.12.2024	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	15.12.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 76 с., 41 рис., 3 табл., 49 джерел.

Тема: Методології функціональної безпеки програмних додатків

Об'єкт дослідження: процеси забезпечення функціональної безпеки програмного забезпечення в системній інженерії.

Мета роботи: розробка методології інтеграції функціональної безпеки програмного забезпечення у процес системної інженерії на основі моделі, що відповідає стандартам безпеки.

Предмет дослідження: методи та підходи інтеграції стандартів функціональної безпеки у системну інженерію на основі моделі.

Результати дослідження

Виконано розробку методології Magic-V-Grid, яка інтегрує принципи функціональної безпеки, у процес системної інженерії на основі моделі. Запропоновано використання профілю для ефективного моделювання аналізу безпеки, що дозволяє надавати необхідні дані в рамках пропонованої методології.

Висновок

В цьому дослідженні представлена методологія для розробки автомобільного програмного забезпечення яка була реалізована шляхом інтеграції робочого процесу розробки програмного забезпечення ISO в технології MagicGrid.

ФУНКЦІОНАЛЬНА БЕЗПЕКА, СИСТЕМНА ІНЖЕНЕРІЯ НА ОСНОВІ МОДЕЛІ, MAGIC-V-GRID, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, АРХІТЕКТУРНИЙ ДИЗАЙН, ВЕРИФІКАЦІЯ ТА ВАЛІДАЦІЯ (V&V), КРИТИЧНІ СИСТЕМИ.

ABSTRACT

Master Thesis: 76 pp., 41 fig., 3 tab., 49 sources.

Thesis Subject: Methodology of functional security of software applications

Research object: processes of ensuring functional security of software in system engineering.

The purpose of the work: development of a methodology for integrating functional security of software into the process of system engineering based on models that meet security standards.

Research subject: methods and approaches of integration of functional safety standards into model-based system engineering.

Research results

The development of the Magic-V-Grid methodology, which integrates the principles of functional safety, into the process of system engineering based on the model, was carried out. It is proposed to use a profile for effective modeling of security analysis, which allows to provide downloaded data within the framework of the proposed methodology.

Conclusion

This study presents an automotive software development methodology that was implemented by integrating the ISO software development workflow into MagicGrid technology.

FUNCTIONAL SAFETY, MODEL BASED SYSTEMS ENGINEERING, MAGIC-V-GRID, SOFTWARE, ARCHITECTURAL DESIGN, VERIFICATION AND VALIDATION (V&V), CRITICAL SYSTEMS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ADAS - Advanced Driver-Assistance System

ADS - Automated Driving System

ASIL - Automotive Safety Integrity Level

BEC - Battery Eliminator Circuit

DBSE -Document-Based System Engineering

E/E - Electrical and Electronic

EAFs - Enterprise Architectural Frameworks

ECU - Electronic Control Unit

FEMMP - Framework for the Evaluation of MBSE Methodologies for Practitioners

FMEA - Failure Mode and Effect Analysis

FTA - Fault Tree Analysis

FuSa - Functional Safety

HARA - Hazard And Risk Analysis

HAZOP - Hazard and Operability Study

HIL - Hardware in Loop

ICD - Interface Control Document

LSA - Logical System Architecture

LSSA - Logical Subsystems Architecture

MBD - Model-Based Design

MBD - Model-Based Development

MBE - Model-Based Engineering

MBFS - Model-Based Functional Safety

MBSE - Model-Based System Engineering

MIL - Model in Loop

MoE - Measurements of Effectiveness

MQO - MBD Quality Objectives

MQR - MBD Quality Requirements

PIL - Processor in Loop

QFD - Quality Function Deployment

QM - Quality Management

RAAML - Risk Analysis and Assessment Modeling Language

RE - Requirement Engineering

S&R - Safety and Reliability

SIL - Software in Loop

SoI - System of Interest

SoS - System of Systems

SOTIF - Safety of the intended functionality

STPA - System Theoretic Process Analysis

SW - Software

TCL - Tool Compliance Level

TeSa - Technical Safety

V&V - Verification and Validation

ВСТУП

Актуальність теми дослідження.

У сучасному світі стрімко зростає залежність від програмних рішень у критичних галузях, таких як автомобільна промисловість, авіація, медична техніка та інші, де безпека та надійність є вирішальними факторами. Функціональна безпека (Functional Safety) стає ключовою вимогою для програмного забезпечення, особливо у системах, де навіть незначний збій може спричинити катастрофічні наслідки. Стандарт ISO 26262, що регулює вимоги до функціональної безпеки для автомобільних систем, підкреслює важливість систематичного підходу до розробки безпечних і надійних програмних компонентів.

Зі збільшенням складності програмних та апаратних систем зростає й складність управління функціональною безпекою. Моделі та методи системної інженерії на основі моделювання (MBSE) надають нові можливості для управління складними системами, дозволяючи інтегрувати всі аспекти розробки, включаючи функціональну безпеку, на рівні моделі. Проте, незважаючи на це, інтеграція вимог функціональної безпеки в MBSE залишається значним викликом, особливо для систем зі складною архітектурою та множинними взаємодіями між підсистемами.

Необхідність узгодження вимог функціональної безпеки з розробкою складних програмних систем і апаратних компонентів створює потребу в ефективних методологіях, що можуть адаптуватися до різних етапів життєвого циклу продукту. Запропонована методологія Magic-V-Grid, що поєднує принципи функціональної безпеки ISO 26262 та MBSE, є актуальною, оскільки дозволяє зменшити складність процесу розробки та підвищити ефективність управління ризиками. Впровадження цієї методології може суттєво вплинути на підвищення безпечності програмного забезпечення в системах, що вимагають високої надійності та мають критичне значення для життя і здоров'я людей.

Зростаюча складність сучасних програмних систем, особливо в контексті автомобільної галузі, вимагає інтеграції функціональної безпеки на кожному етапі життєвого циклу розробки програмного забезпечення. Стандарт ISO 26262 регламентує функціональну безпеку в автомобільних системах, однак його повна інтеграція в системну інженерію на основі моделі (MBSE) потребує подальших досліджень і адаптації до специфічних умов розробки програмних продуктів. Запропонована методологія Magic-V-Grid спрямована на подолання цих викликів, що підкреслює актуальність обраної теми для підвищення надійності та безпечності програмного забезпечення.

В цьому дослідженні представлена методологія для розробки автомобільного програмного забезпечення. Ця методологія була реалізована шляхом інтеграції робочого процесу розробки програмного забезпечення ISO в MagicGrid.

Таким чином, тема дослідження є актуальною як з точки зору наукових досліджень у галузі системної інженерії, так і для практичного застосування у промислових проектах, де безпека програмного забезпечення відіграє ключову роль.

Мета дослідження - розробка методології інтеграції функціональної безпеки програмного забезпечення у процес системної інженерії на основі моделі, що відповідає стандартам безпеки.

Об’єкт дослідження - процеси забезпечення функціональної безпеки програмного забезпечення в системній інженерії.

Предмет дослідження - методи та підходи інтеграції стандартів функціональної безпеки у системну інженерію на основі моделі.

Відповідно до мети роботи було сформовано наступні **задачі:**

- провести огляд літератури щодо сучасних підходів до методологій на основі моделей і функціональної безпеки.

- Визначити найбільш придатні інструменти та методи для інтеграції функціональної безпеки.
- Розробити методологію Magic-V-Grid для інтеграції стандарту функціональної безпеки у процес MBSE.
- Провести порівняння розробленої методології з іншими підходами з літератури.
- Оцінити можливість практичного застосування отриманої методології.

Методи дослідження.

У роботі використано такі методи дослідження:

- аналіз літературних джерел щодо сучасних підходів до MBSE, MBFS і функціональної безпеки;

- моделювання процесів інтеграції функціональної безпеки на основі стандарту ISO в середовищі Rhapsody за допомогою мови SysML;

- застосування профілю RAAML для моделювання аналізу небезпек і оцінки ризиків;

- порівняльний аналіз розробленої методології з іншими підходами з літератури.

Наукова новизна отриманих результатів полягає в розробці та впровадженні нової методології Magic-V-Grid, яка інтегрує принципи функціональної безпеки, у процес системної інженерії на основі моделі. Запропоновано використання профілю RAAML для ефективного моделювання аналізу безпеки, що дозволяє надавати необхідні дані в рамках методології.

Практичне значення магістерської роботи полягає в представленні методології, що дозволяє підвищити ефективність і надійність процесу розробки програмного забезпечення, забезпечуючи дотримання вимог

функціональної безпеки. Впровадження цієї методології може бути корисним у реальних інженерних проектах для створення безпечних і надійних програмних рішень у галузях з високими вимогами до функціональної безпеки.

Структура магістерської роботи. Робота складається зі вступу, трьох розділів та висновків. Загальний обсяг роботи становить 76 сторінок, і містить 41 рисунок, 4 таблиці, список використаних джерел із 49 найменувань.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ФУНКЦІОНАЛЬНОЇ БЕЗПЕКИ

1.1. Особливості складних систем прикладного призначення з точки зору безпеки

Автомобільні системи стають дедалі складнішими та мульти дисциплінарними завдяки електрифікації трансмісії та використанню розширеної системи допомоги водієві (ADAS) і автоматизованої системи водіння (ADS), із прогнозованим зростанням ринку автомобільного програмного забезпечення та електричного та електронного обладнання до 462 мільярдів доларів у 2030 році [1].

У зв'язку з цим зростанням управління складністю також стає все більш важливим, особливо для критично важливих систем безпеки, оскільки ризики від систематичних і випадкових збоїв апаратного забезпечення зростають зі складністю. Одним із способів реалізувати це є використання системної інженерії на основі моделі (MBSE). Цей підхід системної інженерії моделює дизайн системи в одному документі, на відміну від кількох документів, створених за допомогою однієї або кількох програм. Тому використання MBSE полегшує керування складними системами. Прикладом цього є те, що в [2] виявлено, що підхід MBSE охоплює архітектурні знання більш повно, ніж традиційний підхід системної інженерії.

Подібно до управління складністю шляхом застосування MBSE, стандарт функціональної безпеки (FuSa) ISO 26262 використовувався для забезпечення безпечної поведінки цих складних E/E систем. Стандарт є адаптацією IEC 61508, яка відповідає конкретним потребам електронних/електронних систем у транспортних засобах, охоплюючи повний життєвий цикл безпеки цих систем. Стандартні посібники з розробки функціональної безпечної системи за робочими продуктами, починаючи від вимог до рівня цілісності автомобільної безпеки (ASIL) [3].

Системи, розроблені за допомогою платформи Motar ICT Group, є прикладом складних систем, якими керує MBSE. Платформа Motar — це набір інструментів розробки на основі моделі для Simulink. Він може генерувати програмне забезпечення на основі AUTOSAR з моделі Simulink, компілювати його та перезавантажувати в електронний блок керування (ECU) (рис. 1.1).

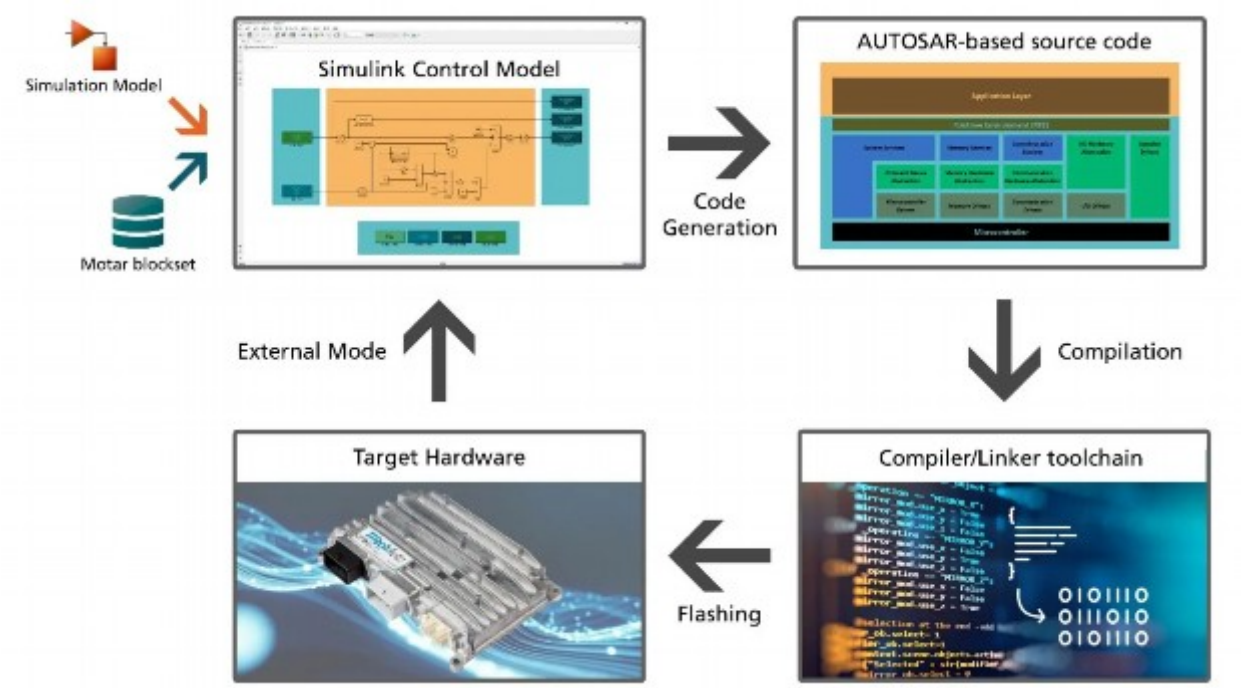

Рис. 1.1. Процес генерації коду платформи Motar

Використовуючи платформу Motar, переклад моделі в код повністю автоматизований; якщо клієнт знайомий із моделюванням у Simul, він може генерувати код AUTOSAR без будь-яких знань програмування. Ще одна перевага автоматизації цього перекладу полягає в тому, що він значно зменшує ймовірність програмних помилок, оскільки згенерований код буде бездоганним, якщо є модель.

Крім того, як контролер, так і платформа Motar довели свою ефективність у галузі; наприклад, він використовувався як контролер автомобіля в гоночному автомобілі [5], міні-навантажувачі [6], машині для дорожніх робіт [7], а також як контролер машини в мобільному крані.

На відміну від позашляхового використання платформи Motar, дотримання ISO 26262 необхідне для майбутніх дорожніх застосувань. Тому команда розробників Motar зараз розробляє платформу для відповідності стандарту ISO 26262.

Оскільки і стандарт ISO 26262, і MBSE зосереджуються на складних (E/E) системах, забезпечуючи безвідмовну поведінку та керуючи складністю відповідно, і оскільки обидва мають певний перетин у робочому процесі, здається очевидним об’єднати їх в одну методологію. Крім того, інтеграція стандарту ISO 26262 у методологію MBSE та моделювання аспектів FuSa за допомогою RAAML призведе до кількох переваг порівняно з використанням стандарту окремо від MBSE.

По-перше, стандарт FuSa потребує деяких вхідних даних, які легко може надати MBSE. Крім того, можна працювати лише в одному файлі, що полегшує ведення огляду всієї системи та всіх її аспектів безпеки, оскільки всі зв’язки між компонентами та їх аналіз безпеки можуть легко знайти. Крім того, процес FuSa може виграти від автоматизації за допомогою методології MBSE, що зробить процес більш ефективним і, можливо, менш схильним до помилок, що призведе до безпечніших систем.

Незважаючи на ці можливі переваги, в літературі можна знайти небагато інформації про комбінацію MBSE з FuSa. Крім того, майже не існує методологій MBSE, які взагалі включають безпеку моделювання.

1.2. Системна інженерія на основі моделей

Системна інженерія — це міждисциплінарний та інтегративний підхід, що дозволяє успішно реалізувати, використовувати та виводити з експлуатації інженерні системи, використовуючи системні принципи та концепції , а також наукові, технологічні та управлінські методи. [12]. Перекладаючи на більш звичну мову, системна інженерія - це цілісний і міждисциплінарний підхід, який дозволяє створювати успішні інженерні

системи протягом їхнього повного життєвого циклу, використовуючи системні принципи та концепції, а також наукові, технологічні та управлінські методи.

Щоб отримати таку успішну інженерну систему, системна інженерія зосереджується на зборі та визначенні потреб зацікавлених сторін, створенні моделі життєвого циклу, створенні й оцінці концепцій і архітектур рішень, моделюванні вимог і архітектури для всіх фаз процесу, а також на виконанні синтезу дизайну, перевірка й валідація системи. Розглядаючи проблеми та домени вирішення, активні системи та служби, функції компонентів у системі, взаємозв'язки між цими компонентами та визначення того, як узгодити всі ці елементи для досягнення задовільного результату.

MBSE відрізняється від класичної системної інженерії, також відомої як DBSE, тим, що повна модель створюється моделюванням, як правило, в одному файлі, замість того, щоб створювати окремі документи, які містять всю інформацію про систему, хоча також можна використовувати MBSE. і DBSE поруч один з одним. Відомо, що «системна інженерія на основі моделей – це системна інженерія з формальним застосуванням моделей, щоб зробити інженерну інформацію доступною для машин для підтримки системних інженерів» [13]. Ця підтримка стає все більш корисною із ускладненням систем, оскільки вона може допомогти зберегти огляд усіх взаємозв'язків у системі та між ними.

Моделі в MBSE побудовані на трьох стовпах: інструменті, мові та методі [14]. Інструмент — це програма, у якій створюється модель, мова описує, які моделі можна створити та як їх можна написати, а метод забезпечує робочий процес і структуру для моделювання.

На рисунку 1.2 можна знайти співвідношення між MBSE та подібними абревіатурами. Можна побачити, що сама MBSE є спеціалізацією Model-Based Engineering (MBE), яка є підходом до розробки, заснованим на візуальному моделюванні, де MBSE зосереджується виключно на візуальному моделюванні систем. Подібним чином MBD є спеціалізацією

MBSE, яка зосереджується лише на проектуванні компонентів системи, таких як зв’язок, обробка сигналів і керування.

Рис. 1.2. Співвідношення між MBSE та подібними методологіями

Крім того, спеціалізація всіх вищезгаданих абревіатур замінює «базований» на «керований», утворюючи модельно-керовану інженерію (MDE), модельно-керовану системну інженерію (MDSE) і модельно-керовану конструкцію (MDD). Різниця тут полягає в тому, що «базовані» підходи підтримують процес розробки, тобто процес базується на підході. «Керовані» підходи автоматизують процес розробки частково або повністю, тобто процес керується підходом.

У цій роботі використовуватиметься лише «базована» термінологія, навіть якщо частини процесу можуть бути автоматизованими. Основна причина цього полягає в тому, що використання обох термінів у цій дисертації може викликати плутанину, особливо коли немає чіткої межі щодо того, скільки автоматизації необхідно, щоб можна було говорити про підхід, керований моделлю. Крім того, «керовані» терміни є спеціалізацією «основних» термінів; отже, використання «базованого» для «керованого» підходу не є неправильним.

1.2.1. Аналіз реалізацій на основі моделей

Зі збільшенням використання MBSE у промисловості також спостерігалося збільшення досліджень мотивації MBSE. У цьому розділі буде розглянуто аналіз найбільш актуальної літератури по темі MBSE.

У дослідженні про те, як змінити DBSE на MBSE, в [16] дійшли висновку, що MBSE є значною перевагою для виконання проекту. Вони виявили, що MBSE є більш ефективним, ніж DBSE, завдяки покращенню повноти вимог, послідовності та комунікації. Було встановлено, що MBSE є найефективнішим у покращенні стратегій запобігання дефектам, що запобігає дорогій переробці, але працює, лише якщо MBSE використовується не лише для керування вимогами. Крім того, вони виявили, що системний інженер, який керує процесом розробки, має значний успіх у продуктивності. Потрібні додаткові дослідження для врахування культурних відмінностей, мотивації, стилів управління, тиску ринку, урядових постанов і досвіду персоналу.

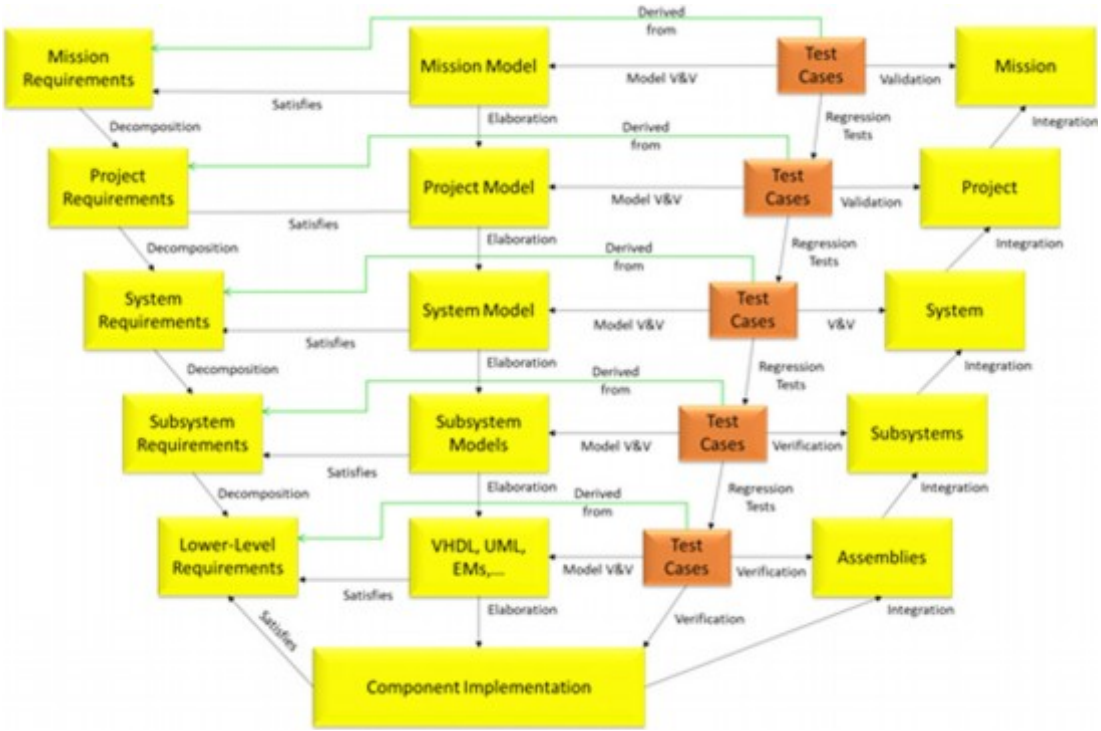

Рис. 1.3. MBSE V&V-модель [17]

В [17] виявлено, що MBSE головним чином мотивувався виправленням DBSE шляхом автоматичного створення документів для різних зацікавлених сторін і покращенням комунікації між дисциплінами та зацікавленими

сторонами. Тут зроблено висновок про зростаючий інтерес до MBSE, який застосовувався у все більшій кількості додатків, таких як верифікація та валідація на основі моделі (V&V) і цифрові близнюки на ранніх етапах. Крім того, курси MBSE у великих аерокосмічних компаніях та університетах демонструють зростаючий імпульс. З іншого боку, він міг помітити розрив між результатами MBSE та потребами клієнтів, наприклад, що методи MBSE повинні охоплювати повний життєвий цикл системи, що MBSE слід демонструвати в реальних проектах і що дослідники повинні більше думати про V&V як у його прикладі (рис. 1.3), а QoM — як інженери програмного забезпечення. Хоча його приклад дає розуміння зв'язків між вимогами, моделлю, верифікацією та інтеграцією, модель може сприймається як надто високий рівень, оскільки не описано, що мають передбачати конкретні елементи.

Аналіз літературних джерел, проведений в [18], виявив значну розбіжність між декларованими перевагами MBSE та наявними емпіричними даними. Незважаючи на численні твердження про ефективність MBSE, об'єктивних доказів, які б підтверджували ці твердження, виявилося вкрай мало. Лише два з 360 розглянутих досліджень містили вимірювані дані, проте й вони мали методологічні обмеження, що ускладнює однозначні висновки.

Дослідження [21] присвячене використанню моделей Requirement Engineering (RE) в автомобільній промисловості, також виявило відсутність консенсусу щодо переваг такого підходу. Автори відзначають, що застосування моделей RE часто обмежується конкретними цілями та стикається з проблемами, пов'язаними з обмеженими можливостями інструментів та високою складністю моделей.

В [23] проведено кількісне порівняння MBSE та традиційних підходів (DBSE) і дійшов висновку про більш повне охоплення архітектурних знань у MBSE. Однак було виявлено ряд обмежень MBSE, пов'язаних з представленням певних концептуальних категорій.

Огляд літератури, проведений в [24], підтвердив, що технології, пов’язані з SysML, є найбільш зрілими в контексті MBSE. Автори також підкреслили важливість стандартизації обміну даними та необхідність досліджень, орієнтованих на весь життєвий цикл розробки продукту.

Загалом, аналіз літератури демонструє, що MBSE часто мотивується потенційними перевагами в продуктивності проекту. Однак для підтвердження цих переваг необхідні більш об’єктивні та ретельні дослідження. Майбутні дослідження повинні зосередитися на розробці ефективних методів оцінки MBSE, а також на розширенні функціональних можливостей інструментів і стандартів, що підтримують MBSE.

1.3. Функціональна безпека в автомобільній техніці

Зі зростанням складності автомобільних систем суттєво підвищуються ризики системних збоїв. Для забезпечення безпеки транспортних засобів було розроблено низку міжнародних стандартів. Нижче наведено короткий огляд трьох основних стандартів: ISO 26262, ISO 21448 та ISO 21434.

1.3.1. Функціональна безпека

Стандарт ISO 26262 зосереджений на зниженні ризиків, пов’язаних із системними та випадковими апаратними збоями в автомобільних системах. Він визначає вимоги та процеси, які необхідно дотримуватися протягом усього життєвого циклу розробки, включаючи:

- Термінологію
- Управління функціональною безпекою
- Функціональну безпеку на етапі концепції
- Функціональну безпеку на рівні системи, апаратного забезпечення та програмного забезпечення
- Функціональну безпеку під час виробництва, експлуатації та допоміжних процесів

- Аналіз ризиків

ISO 26262 описує життєвий цикл автомобіля (рис. 1.4) і дії, які необхідно виконувати на кожному його етапі, щоб відповідати вимогам функціональної безпеки. Однією з відмінних рис ISO 26262 є те, що впроваджуються відповідні заходи безпеки на основі рівня ризику, якщо програмне забезпечення виходить з ладу.

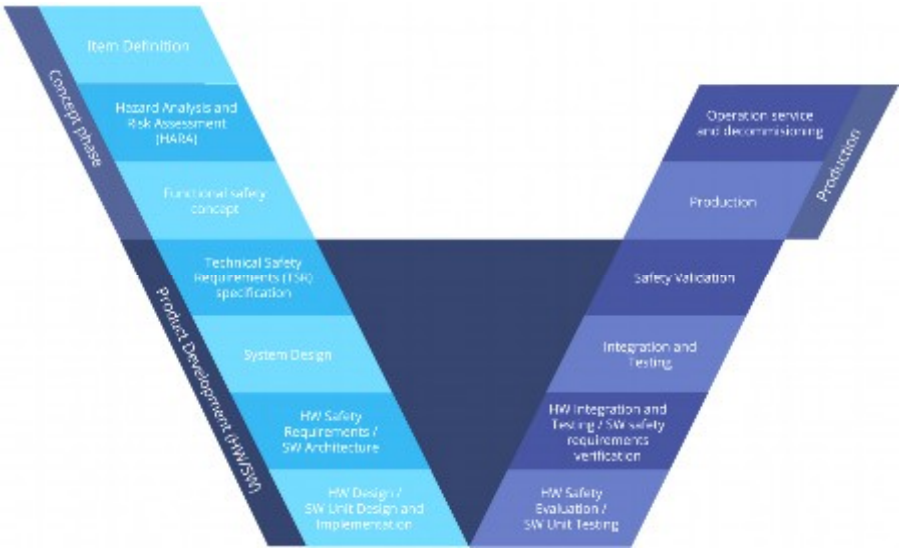

Рис. 1.4. Життєвий цикл згідно ISO 26262

1.3.2. Безпека передбаченої функціональності

Стандарт ISO 21448 доповнює ISO 26262, розглядаючи ризики, пов’язані з неправильною роботою систем, що використовують складні датчики та алгоритми для оцінки ситуації (наприклад, системи допомоги водієві). На відміну від ISO 26262, який фокусується на випадкових збоях, ISO 21448 спрямований на усунення систематичних помилок у функціональності.

1.3.3. Інженерія кібербезпеки

Стандарт ISO 21434 забезпечує систематичний підхід до забезпечення кібербезпеки автомобільних систем. Він охоплює весь життєвий цикл розробки, від концепції до виведення з експлуатації, та включає такі аспекти:

- Управління кібербезпекою

- Аналіз загроз та оцінка ризиків

- Захист від кібератак

У рамках цього дослідження основна увага приділяється стандарту ISO 26262. Стандарти ISO 21448 та ISO 21434 не розглядаються детально, оскільки дослідницька установка не містить функцій автономного водіння та не піддається значним кіберзагрозам. Однак, ці стандарти можуть бути враховані у подальших дослідженнях, якщо виникатиме необхідність.

1.3.4. Розробка на основі моделей

Стандарт ISO 26262 приділяє певну увагу підходам, заснованим на моделюванні (MBD), хоча й не надає детальних рекомендацій щодо їх застосування.

Переваги MBD:

- Зниження ризику помилок: Автоматична генерація коду може зменшити кількість помилок, пов’язаних з ручним написанням коду.

- Поліпшення якості: Використання моделей дозволяє забезпечити послідовність та трасування вимог протягом усього життєвого циклу розробки.

- Автоматизація перевірки: Моделі можуть бути використані для автоматизації перевірки моделей у циклі (MIL) та програмного забезпечення в циклі (SIL).

Ризики та обмеження MBD:

- Систематичні помилки: Неповна або неправильна формалізація вимог у моделі може призвести до системних помилок.

- Залежність від інструментів: Якість результатів залежить від якості інструментів, що використовуються для моделювання та генерації коду.

- Складність моделей: Складні моделі можуть бути важко підтримувати та модифікувати.

- Неповне охоплення: Моделі можуть не охоплювати всі аспекти системи, що може призвести до пропуску потенційних проблем.

Рекомендації стандарту ISO 26262:

- Комбінація моделей та текстових документів: Для забезпечення повноти та точності специфікації вимог рекомендується використовувати як моделі, так і текстові документи.

- Ретельна верифікація моделей: Необхідно забезпечити, що моделі адекватно відображають реальну систему та не містять помилок.

- Обережне використання автоматичної генерації коду: Якість згенерованого коду залежить від якості моделі та інструменту генерації.

MBD може бути ефективним інструментом для підвищення якості та безпеки розробки програмного забезпечення в контексті ISO 26262. Однак, для успішного застосування MBD необхідно ретельно оцінити ризики та обмеження цього підходу, а також дотримуватися рекомендацій стандарту.

1.3.5. Застосування стандартів у Simulink

Оскільки Motar Toolbox реалізовано в Simulink, для цього проекту також важливо обговорити, як конструкції в Simulink можуть відповідати ISO 26262. Таким чином, цей розділ відповідає на питання про те, як ISO 26262 можна застосувати в Simulink.

Як показано на рисунку 1.5, робочий процес MBD починається з фази планування програмного забезпечення. Тут визначаються використовувані стандарти, визначаються вимоги до якості MBD (MQR) і розподіляються між зацікавленими сторонами, складається стратегія досягнення цілей якості MBD (MQO), а відповідність MQR вже планується наприкінці проекту [27].

Далі, на етапі вимог до програмного забезпечення, функціональна модель і вимоги моделюються одночасно. Основна увага у функціональній моделі зосереджена на розробці вимог до програмного забезпечення за допомогою структурованих алгоритмів, які можна використовувати для

перевірки вимог до програмного забезпечення [27]. Для конформації всі вимоги мають бути відстежені до архітектури [28].

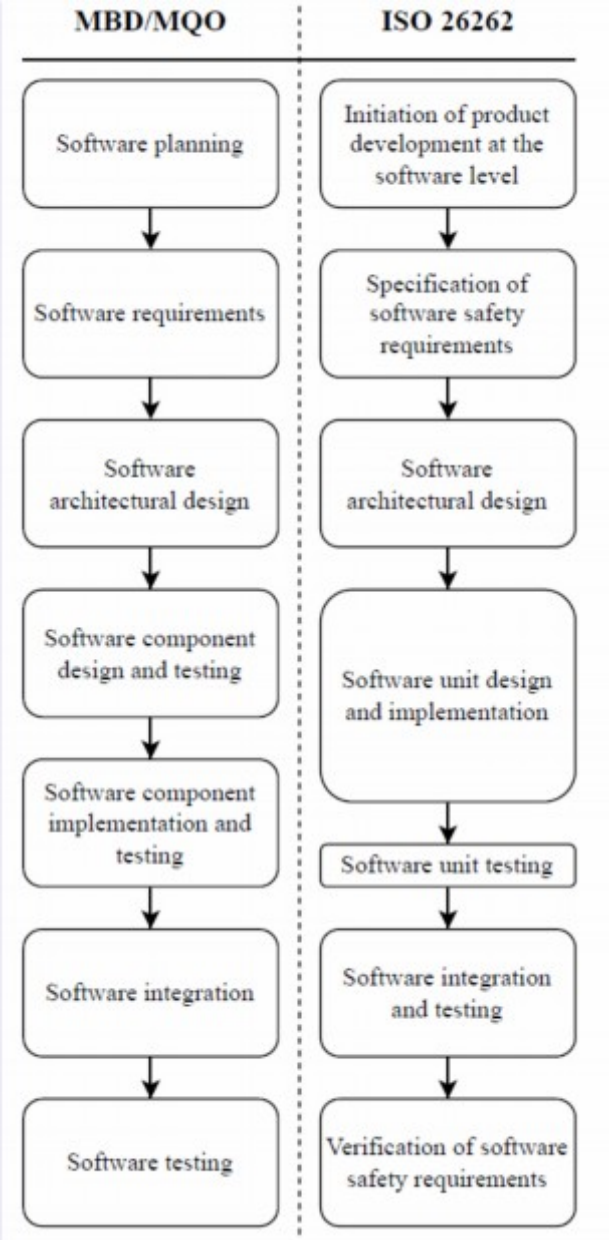

Рис. 1.5. Порівняння робочого процесу між MBD/MQO та ISO 26262

Статичний дизайн архітектури програмного забезпечення повністю визначений інтерфейсами та плануванням на етапі проектування архітектури програмного забезпечення. Оскільки архітектура є частиною проекту, повна специфікація має відповідати всім використаним стандартам якості, безпеки та/або архітектури. Для узгодженості функціональна модель може служити

базовою лінією для архітектурного проекту [2]. Статичні та динамічні аспекти розробки архітектури програмного забезпечення можуть бути виконані за допомогою Simulink і System Composer. Simulink Check можна використовувати для автоматичної перевірки відповідності проекту стандартам, таким як керівні принципи високої цілісності для ISO26262 [28].

Проектування та тестування програмного компонента включає повну специфікацію дизайну програмного компонента та перевірку цього дизайну як динамічним, так і статичним аналізом. Дизайн компонентів далі визначає дизайн програмного забезпечення за допомогою структури, планування та алгоритмів. Інтерфейси мають узгоджуватися з інтерфейсами архітектурної моделі та/або їх можна повторно використовувати. Крім того, знову ж таки, для узгодженості, алгоритми з функціональної моделі можуть функціонувати як базова лінія для визначення алгоритмів виробництва [27].

Для перевірки можна використовувати структуру ISO26262 V&V від Simulink Test. Тут аналіз структурного покриття для перевірки повноти тесту можна виконати за допомогою покриття оператора, покриття гілки або MC/DC, усі з яких можна реалізувати за допомогою Sim ulink Coverage. Якщо охоплення недостатнє, Simulink Design Verifier може автоматично генерувати більше тестів для досягнення повного охоплення. Крім того, Simulink Test також може включати підтримку циклічних тестів, таких як MIL, SIL, Hardware in Loop (HIL) і Processor in Loop (PIL) [28].

Етап впровадження та тестування програмного компонента визначає як дизайн компонента програмного забезпечення (ПЗ), так і впровадження. Структуру, планування та алгоритми можна повторно використовувати з дизайну компонента для узгодженості. Крім того, спосіб реалізації алгоритмів можна адаптувати для задоволення нефункціональних вимог [27].

Загалом, при використанні функцій MATLAB у Simulink усі вимоги мають бути пов’язані з блоками або функціями MATLAB. Функціональність MATLAB також можна використовувати в Simulink за допомогою функцій MATLAB, наприклад, для функцій ADAS.

1.4. Функціональна безпека на основі моделі

У контексті літератури, присвяченої застосуванню MBSE у поєднанні з функціональною безпекою (FuSa), дослідження [32] демонструє, як моделі SysML можуть бути використані для підтримки аналізу безпеки з використанням інструменту ANSYS Medini на прикладі ракетної системи. Проте, у цій роботі не розглядається застосування стандарту ISO 26262 у рамках MBSE, хоча існують певні паралелі між представленим підходом та вимогами ISO 26262.

Подібно, в [33] при розробці методу інтеграції системних та програмних моделей виявив, що стандартний профіль SysML є корисним, але недостатнім для повноцінної підтримки FuSa. Дослідник також відзначив, що забезпечення високого рівня простежуваності вимог може залежати від застосовуваного інструментарію.

В [34] продемонстровано додаткову перевагу MBSE для FuSa, показавши, що розробка програмного забезпечення для керування функціями віконної системи за допомогою методу моделювання на основі моделей (MBD) у MATLAB відповідно до стандарту ISO 26262 полегшує виконання тестування на рівні моделей (MIL).

Автор в [35] вказав на те, що оскільки стандарт ISO 26262 вимагає інформацію про функціональні можливості системи з зовнішніх процесів, а сам стандарт обмежується лише аспектами FuSa, потрібен процес розробки, який враховує всю необхідну функціональність системи. Для цього автори запропонували інтегровану V-модель для FuSa (рис. 1.6), в якій додали вимоги та процес проектування до спрощеної версії потоку проєктування ISO 26262. Вони зазначили, що така модель є більш придатною для нетрадиційних автомобільних компаній, які прагнуть відповідності ISO 26262, оскільки вона не відповідає жорстким нормам традиційної автомобільної промисловості.

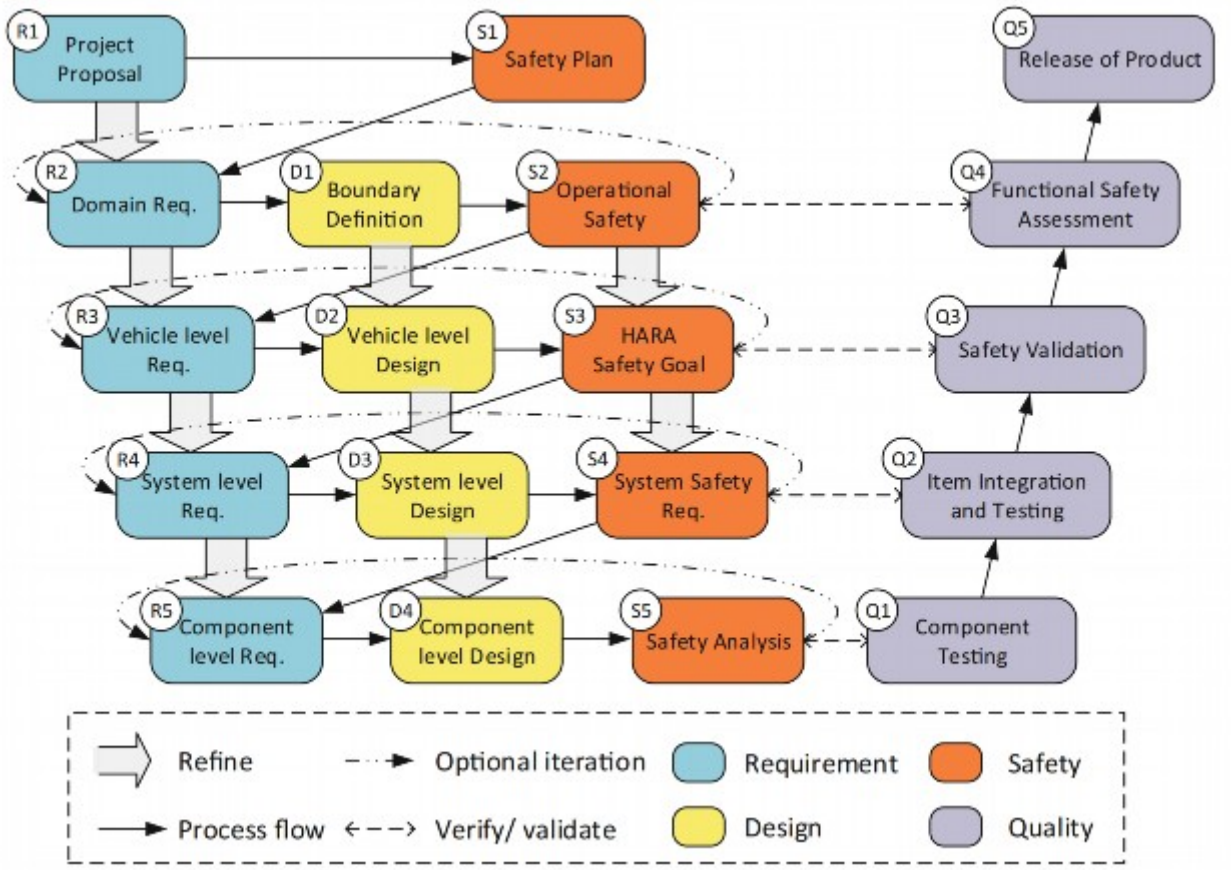

Рис. 1.6. Інтегрована V-модель [35]

Подібно до [36] та [37] запропонували структуру, що включає не лише функціональну безпеку, але й заходи з кібербезпеки. Їхня структура відповідає вимогам стандартів кібербезпеки ISO 26262 та ISO 21434. У порівнянні зі структурою, їхні моделі ближчі до стандарту та не включають функціональні частини, такі як вимоги та проектування.

1.4.1. Мова моделювання аналізу та оцінки ризиків

Мова системного моделювання SysML не охоплює аспекти моделювання безпеки та надійності (S&R), незважаючи на зростаючий попит у спільноті системних інженерів. Відповідаючи на цю потребу, Object Management Group (OMG) нещодавно випустила новий профіль SysML для моделювання S&R під назвою Risk Analysis and Assessment Modeling Language (RAAML). Цей профіль забезпечує можливість моделювання S&R відповідно до існуючих стандартів у таких галузях, як аерокосмічна, медична

та залізнична, з використанням методів FMEA, FTA, а також у сфері функціональної безпеки автомобільної промисловості (FuSa) відповідно до стандарту ISO 26262 (рис. 1.7). Наприклад, RAAML підтримує моделювання дослідження небезпеки та працездатності (HAZOP) [38].

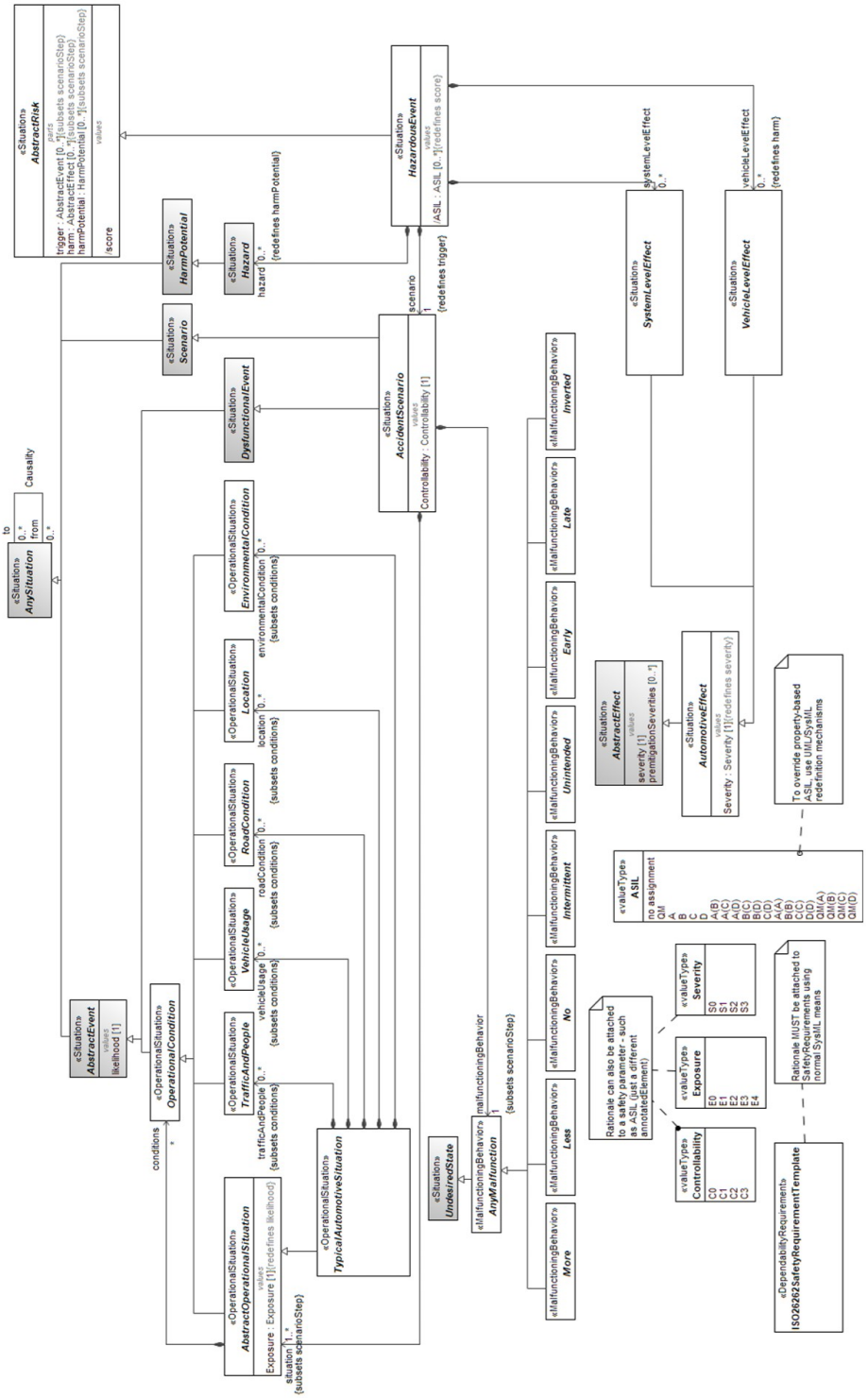

Рис. 1.7. Бібліотека ISO26262

Профіль RAAML для ISO 26262 включає додаткові елементи, які уточнюють або розширюють загальний профіль. Для моделювання HAZOP, наприклад, він передбачає моделювання неправильної поведінки системи, яка може призвести до небезпечних ситуацій, та операційні умови, що в поєднанні з неправильною поведінкою можуть створити сценарії аварій та небезпечних подій. Ці моделі включають інформацію про сценарії аварій, а також відповідний рівень ризику (ASIL) для кожної події.

У науковому дослідженні [39] продемонстровано застосування RAAML для аналізу дерева відмов (FTA) на основі моделі. Він виявив, що основні переваги RAAML перед DBSE включають пряму простежуваність та можливість обміну результатами між різними інструментами і компаніями. В [40] показано використання RAAML для автоматичного створення аналізу STPA, підкреслюючи, що RAAML сприяє управлінню зростаючою складністю систем і забезпечує можливість формальної перевірки завдяки автоматизації. Однак він також зазначив, що для інтеграції RAAML в інструменти SysML необхідний API інструменту. Крім того, в [41] показано застосування RAAML для аналізу функціональних небезпек (FHA), що є частиною стандарту ARP 4761 для розробки авіаційних систем, критично важливих для безпеки.

Висновки до розділу

У цьому розділі проведено систематичний огляд літературних джерел та галузевої практики з метою визначення та уточнення понять MBSE та суміжних термінів. Зокрема, детально розглянуто три ключові стандарти в галузі автомобільної безпеки, проаналізувавши їхні відмінності та сфери застосування. Окрема увага приділена можливостям інтеграції вимог стандарту ISO 26262 у процес розробки на основі моделей (MBD), що є невід’ємною частиною парадигми MBSE. Наведено конкретні приклади застосування ISO 26262 в середовищі моделювання Simulink, демонструючи

практичні аспекти такої інтеграції. Крім того, у розділі представлено аналіз сучасного стану методів MBSE та MBFS, із особливим акцентом на ролі мови моделювання RAAML. Проведено оцінку потенціалу RAAML для підтримки процесів забезпечення функціональної безпеки в контексті розробки складних систем.

Таким чином, у цьому розділі сформовано теоретичне підґрунтя для подальшого дослідження та розробки методології, а також визначено актуальні напрямки наукових досліджень у цій галузі.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ МЕТОДОЛОГІЇ ІНЖЕНЕРІЇ НА ОСНОВІ МОДЕЛІ В КОНТЕКСТІ ФУНКЦІОНАЛЬНОЇ БЕЗПЕКИ

2.1. Порівняння інструментів, мов і методологій MBSE

В розділі досліджуються, як інструменти, мови та методології MBSE (рис. 2.1) під час аналізу літератури. Крім того, представляється повна колекція всіх методологій, які були оцінені відповідно до FEMMP. Розділ завершується поясненням того, який інструмент, мову та методологію було обрано для використання в якості основи для методу MBFS.

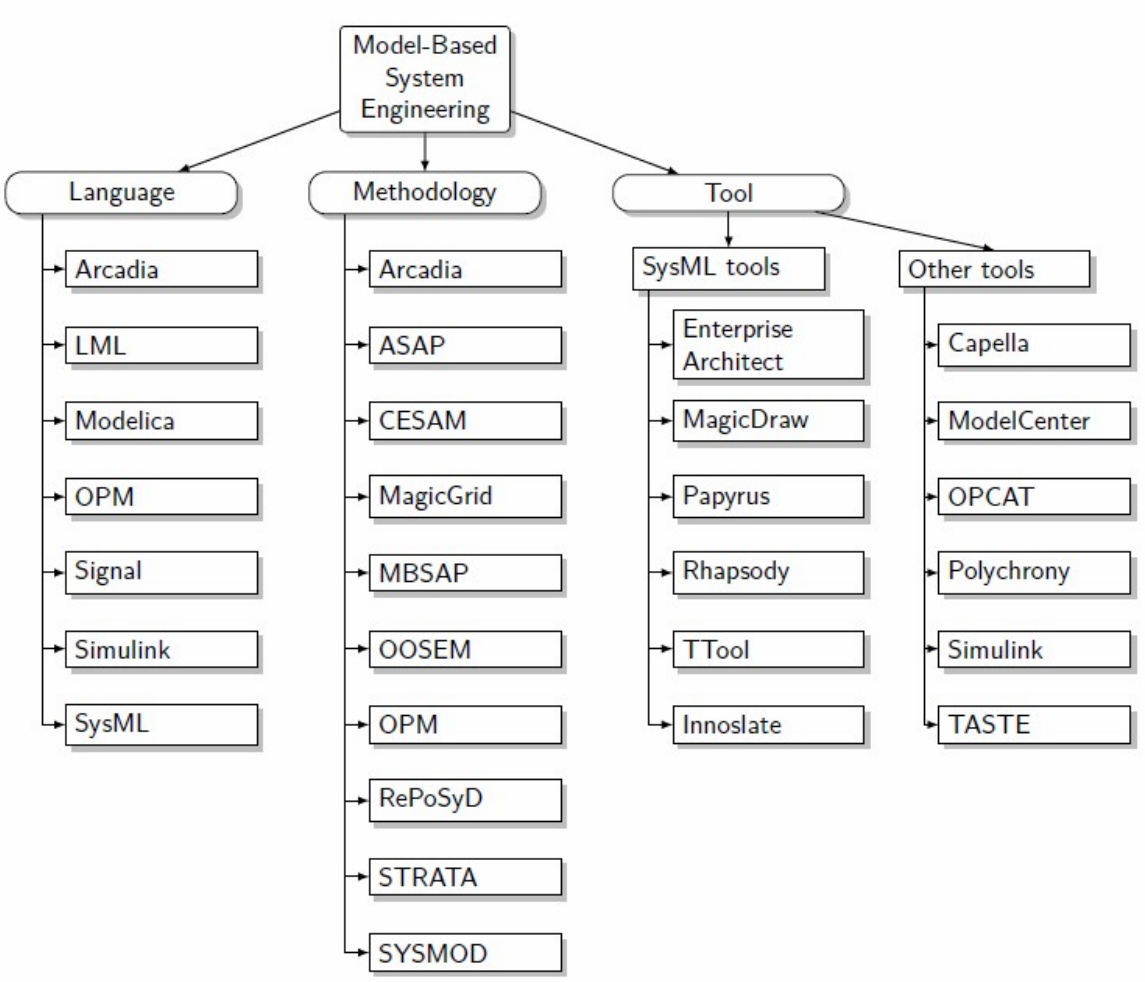

Рис. 2.1. Представлення мови, методології та інструментів MBSE

При ухваленні рішення на користь використання MBSE замість DBSE в проекті, виникає складність у виборі відповідних інструментів, мови та

методології з численних доступних варіантів. Щодо вибору мови та інструментів для реалізації MBSE, дослідження [42] надає допомогу студентам і практикам у цій сфері. Автори виділяють три типи мов: неформальні (наприклад, природна мова або графічні схеми), напівформальні (SDL, OPM, Simulink, SysML) і формальні мови (діаграми станів, кінцеві автомати). Інструменти вони класифікують за наявністю редактора, симулятора, засобів перевірки, методичної підтримки та/або генератора коду. Дослідження також розрізняє інструменти, що підтримують SysML, і ті, що його не підтримують.

Ще одне дослідження, що допомагає у виборі інструменту MBSE це [43]. У ньому описується основа для вибору інструментів на основі аналізу ринку, підходів до розгортання функцій якості (QFD) і матриці рішень. Основною перевагою цього підходу є можливість налаштування вагомості різних категорій за допомогою QFD, що спрощує процес вибору інструмента відповідно до специфічних потреб користувача. Однак ця методологія ще не пройшла практичну перевірку у конкретному дослідженні.

В [44] запропоновано метод FEMMP, який може використовуватися для порівняння різних MBSE-методів. Оцінювання методів здійснюється на основі стандартних критеріїв, таких як процес, якість моделі, практична реалізація в інструменті та можливість застосування на прикладі. Пізніші дослідження вдосконалили цей підхід і застосували його до додаткових методологій, включаючи Arcadia та OOSEM [45], STRATA, OPM, RePoSyD і Arcadia [46].

У контексті критики, в [46] виявили необхідність додавання критеріїв для оцінки кривої навчання, метамоделі та механізмів розподілу безпеки. Також було зазначено, що відповідність стандарту ISO 15288 не обов'язково означає, що методи MBSE легко дозволяють дотримуватися цього стандарту. Крім того, інші тематичні дослідження можуть охоплювати інші аспекти MBSE, що дозволяє краще порівнювати методи.

Також зазначено, що у FEMMP відсутні критерії зрілості та впровадження методів у промисловості. Крім того, невідокремлене оцінювання інструментів і методів може вводити в оману. Подібну критику висловивлено в [48], стверджуючи, що необхідно окремо аналізувати три стовпи MBSE: метод, інструмент і мову. Він також вказав на відсутність еталона для порівняння інструментів у FEMMP і запропонував структуру COFRA, яка порівнює методи й інструменти MBSE на основі критеріїв точності, сумісності, повторного використання, зручності та ефективності. Однак їх дослідження порівнює лише два інструменти/методи/мови — Rhapsody/SysML і Capella/Arcadia.

Результати оцінки всіх методів, розглянутих у дослідженнях на основі FEMMP, представлені в таблиці 2.1. Ця таблиця базується на сітці детальної оцінки в [47], яка об'єднує порівняння шести методологій за 19 критеріями. У таблицю також додано методи SYSMOD та MDDM, OPM. Однак, через різницю у критеріях порівняння деякі комірки для доданих методів залишилися порожніми.

Таблиця 2.1.

Оцінка методів MBSE

Criterion	OOSEM	STRATA	RePoSyD	Arcadia	ASAP	Grid	SYSMOD	MDDM	OPM
Tailoring	A	A	C	B	B	A	A	A	B
Complexity	B	A	B	A	A	A	B	B	B
Documentation	B	C	C	A/B	B	C	B	C	B
Training		B		A	A	C	B	B	A
Maturity	Yes	Yes		Yes	Yes				
Industry adoption	Yes	Yes		Yes	Yes	Yes			
Philosophy	Yes	Yes		Yes	Yes	Yes	Yes	Yes	Yes
Language	SysML	SDL	RML	SysML-like	SysML	SysML	SysML	UML-like	OPM
Integration	C	B	A	C	B	B	C	A	B
Abstraction	B	A	B	A	A	A			
Meta-Model	SysML	Yes	Yes	Yes	SysML	SysML	SysML		
Scalability	A	B	A	A	A	A	A	B	B
Variants	B	B	B	B	B	A	B	C	B
Independent View generation				C	A	A			
Redundancy	B	A	C	A	B	B	A	A	B
NFR				B	B	A			
Pb vs. Solution				A	B	A			
Traceability				A	A	A			
Consistency	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

* примітка: A - повністю відповідає, B - прийнятна продуктивність, C - обмежена застосовність.

2.2. Аргументація вибору моделі та платформи

У цьому розділі детально розглядаються причини вибору методу, інструменту та мови, використаних у дослідженні, на основі їхніх переваг порівняно з іншими альтернативами. Окрім цього, пропонується докладний аналіз обраних технологій.

Мова. У цьому дослідженні було обрано мову SysML не лише через її популярність у науковій літературі, але й завдяки можливості моделювання аспектів безпеки та надійності (S&R) у поєднанні з профілем RAAML. Такий вибір забезпечує інтеграцію стандартів безпеки в різних галузях.

Методологія. Обрано методологію MagicGrid, оскільки вона охоплює всі основні принципи SysML та S&R у трьох ключових сферах. MagicGrid не лише забезпечує методологічний підхід і структуру, але також розроблена як гнучка основа, яка може бути адаптована під потреби користувача, що спрощує інтеграцію вимог функціональної безпеки (FuSa) у процес. Хоча результати порівняння методологій у таблиці 2.1 є неповними, MagicGrid демонструє рівні або кращі результати за багатьма критеріями. Важливою перевагою цієї методології є її поєднання з адаптованою V-моделлю та включення аспектів безпеки та надійності, що робить її ідеальною для інтеграції стандарту ISO 26262.

Інструмент. Для цього дослідження було обрано інструмент Rhapsody, оскільки разом із Cameo Systems Modeler він є одним із найбільш ефективних інструментів для моделювання SysML. Крім того, автор дослідження вже має ліцензію на Rhapsody та значний досвід роботи з цим інструментом, що сприяє його перевазі над Cameo. Недоліком використання Rhapsody у поєднанні з MagicGrid є необхідність розробки додаткового профілю SysML для моделювання всіх елементів MagicGrid, тоді як Cameo має повну інтеграцію з MagicGrid. Однак цей недолік не є критичним, оскільки обидва інструменти базуються на SysML, і додавання відсутніх елементів через профіль у Rhapsody не повинно викликати труднощів.

Методологія MBSE та архітектурна структура MagicGrid, раніше відома як MBSEGrid, була розроблена на основі досвіду компаній Bombardier Transport і Kongsberg Defence and Aerospace, а також на основі спроб впровадження MBSE в інших організаціях. Як і інші методології MBSE, MagicGrid надає рекомендації щодо робочих процесів системного моделювання, часто з використанням ітеративного підходу. Проте, більшості таких методологій бракує чіткої основи для керування моделюванням на різних рівнях абстракції, що робить їх застосування надто загальним і складним для практичного використання.

MagicGrid вирізняється тим, що черпає натхнення з Enterprise Architectural Frameworks (EAF), таких як UAF, MODAF, DODAF, TOGAF, NAF і DNODAF. Хоча ці EAF занадто складні для повного застосування в системній інженерії, MagicGrid пропонує спрощену, але структуровану модель. Вона представлена у вигляді сітки (рис. 2.2), де на осі абсцис розміщено чотири стовпи SysML: вимоги, структура, поведінка та параметри.

			Pillar				
			Requirements	Structure	Behavior	Parameters	Safety & Reliability
Domain	Problem	Black Box	Stakeholder Needs	System Context	Use Cases	Measures of Effectiveness (MoEs)	Conceptual and Functional Failure Mode & Effects Analysis (FMEA)
		White Box		Conceptual Subsystems	Functional Analysis	MoEs for Subsystems	Conceptual Subsystems FMEA
	Solution		System Requirements	System Structure	System Behavior	System Parameters	System Safety & Reliability (S&R)
			Subsystem Requirements	Subsystem Structure	Subsystem Behavior	Subsystem Parameters	Subsystem S&R
			Component Requirements	Component Structure	Component Behavior	Component Parameters	Component S&R
	Implementation		Implementation Requirements				

Рис. 2.2. Платформа MagicGrid

На осі ординат розташовані різні рівні абстракції системи. Проблемний домен поділяється на шари "білого ящика" і "чорного ящика", а домен

рішення охоплює рівні системи, підсистеми та компонентів. Нарешті, домен імплементації стосується етапу реалізації. Кожна комірка цієї сітки може бути представлена однією або кількома діаграмами SysML, що робить методологію гнучкою та придатною для практичного застосування.

Пізніше MagicGrid було вдосконалено та розширено. Його було зіставлено з ISO 15288, щоб відповідати існуючим стандартам процесу. Крім того, було додано елемент «Безпека та надійність», оскільки галузь збільшила інтеграцію S&R у MBSE. Крім того, було показано показали, як MagicGrid можна використовувати в V-моделі (рис. 2.3) із діями верифікації та перевірки.

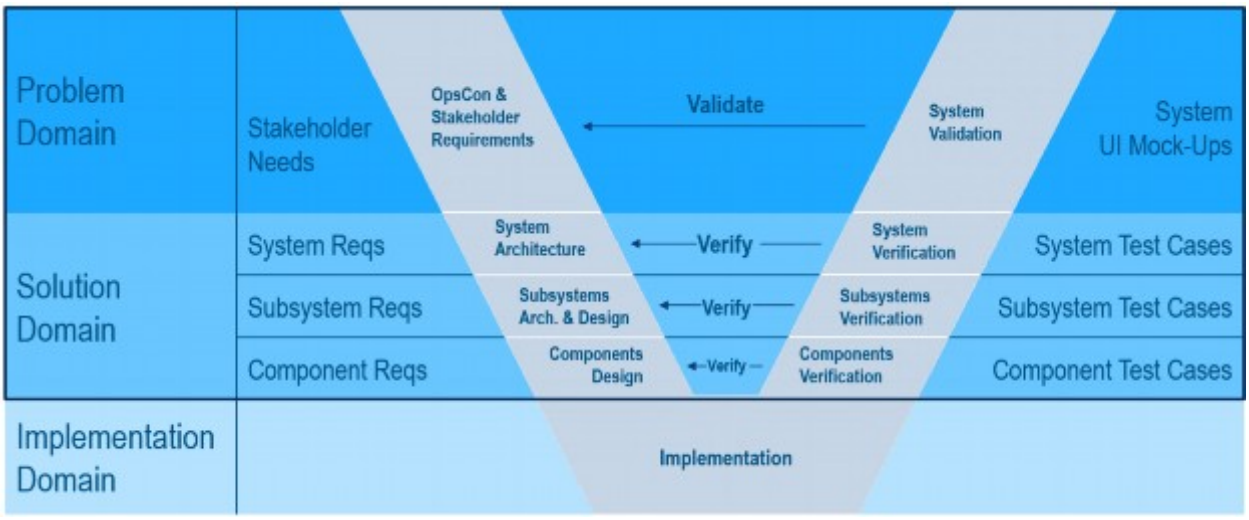

Рис. 2.3. V-модель MagicGrid

Стовп безпеки та надійності в MagicGrid розділяється на два основних етапи: попередній аналіз ризиків у проблемній області та функціональний HARA в області вирішення. Часто обирають метод FMEA для попереднього аналізу ризиків. Вхідними даними для попереднього аналізу ризиків є елементи, що стосуються потреб зацікавлених сторін, системного контексту, функціонального аналізу та взаємозв'язків логічної підсистеми.

Що стосується функціонального HARA, будь-який метод може бути використаний для його виконання. Основними вхідними даними на цьому

етапі є елементи, пов'язані з вимогами до системи, її поведінкою та структурою.

При інтеграції MagicGrid із V-моделлю, перевірка здійснюється за принципом зворотного шляху, тобто знизу вгору, за допомогою тестових випадків. Для їх фіксації в SysML можуть бути використані діаграми автоматів, послідовності або активності. Тестові приклади можна пов'язати з відповідними компонентами, підсистемами чи системою за допомогою зв'язків перевірки в SysML. Така перевірка виконується для забезпечення відповідності вимогам зацікавлених сторін, наприклад, через макети інтерфейсів користувача.

2.3. Представлення діаграм та програмних бібліотек для реалізації

Як вже було зазначено, для використання MagicGrid у середовищі Rhapsody необхідно створити додатковий профіль, що зображено на рисунку 2.4.

Рис. 2.4. Елементи з профілю MagicGrid для Rhapsody, які відображаються в браузері моделі

Цей профіль включає в себе матрицю відстеження, стереотипи для функціональних вимог, показники ефективності (MoE), а також сигнали та макети таблиць для контрольного документа інтерфейсу (ICD) і таблиць вимог, що відображають потреби зацікавлених сторін. Деталі цього профілю

наведено на рисунках нижче. Оскільки документацію по роботі з Rhapsody знайти складно, то також розглянемо поради та рекомендації щодо використання Rhapsody у поєднанні з MagicGrid. Ці поради базуються враховують моменти, де Rhapsody відхиляється від стандартного підходу до моделювання епізодичних систем.

Таблиці вимог або діаграми можуть бути створені для підсистем або компонентів шляхом зміни обсягу діаграми або таблиці.

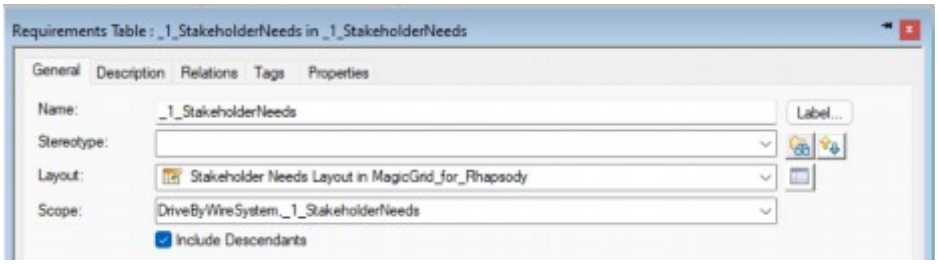

Рис. 2.5. Загальні характеристики з таблиці вимог, де можна вказати область застосування

Також можна додати будь-яку інформацію як блок, створивши блок і потім знайшовши його в функціях потоку.

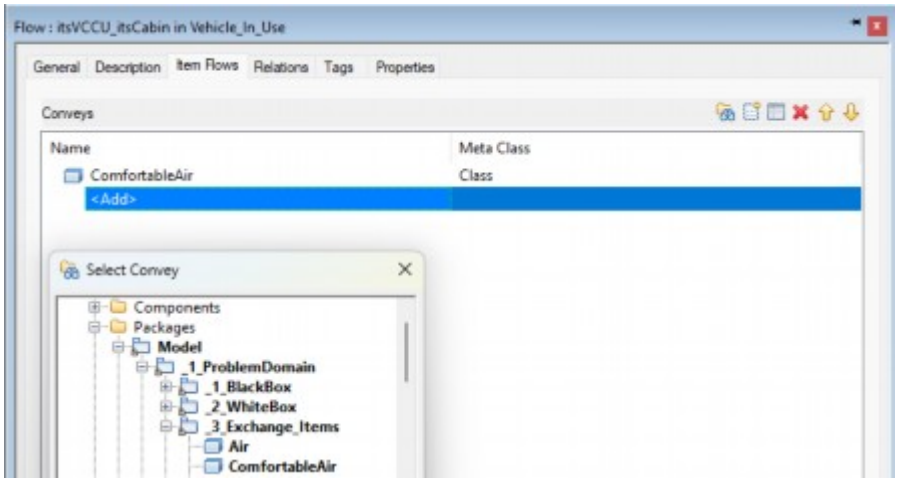

Рис. 2.6. Додавання інформації до потоку

Для автоматичного створення доріжки потрібно спочатку перетягнути частини в провіднику на діаграму UC, потім створити доріжку зі шляху, а

потім перетягнути частини туди, звідки вони походять, і, нарешті, перейменуйте доріжки у провіднику файлів.

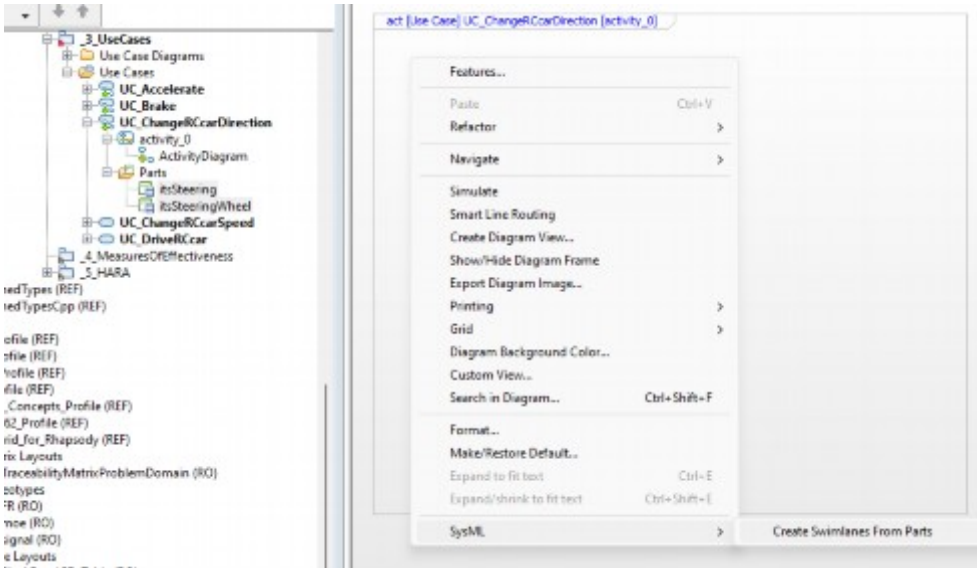

Рис. 2.7. Процес створення доріжок

Щоб створити залежність імпорту на діаграмі пакета, потрібно клацнути правою кнопкою миші, і додати залежність і клацнути на імпорт стереотипу.

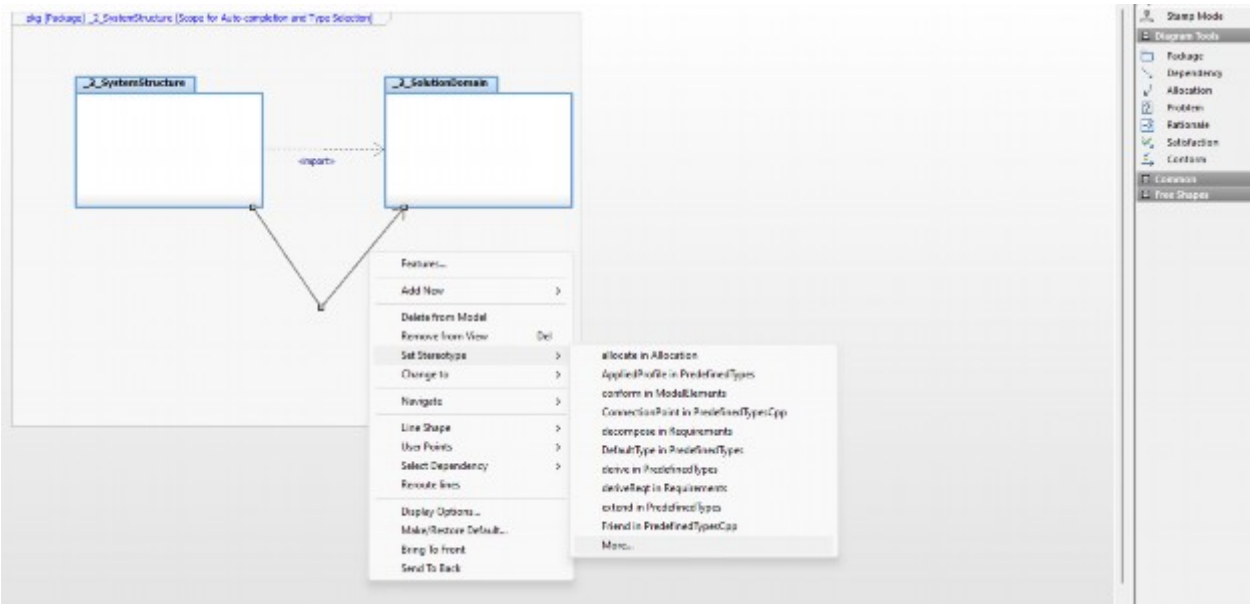

Рис. 2.8. Процес додавання стереотипу імпорту до залежності на діаграмі пакета

Щоб створити залежність імпорту на діаграмі пакета, потрібно клацнути правою кнопкою миші і додати залежність, а потім натиснути імпорт стереотипу.

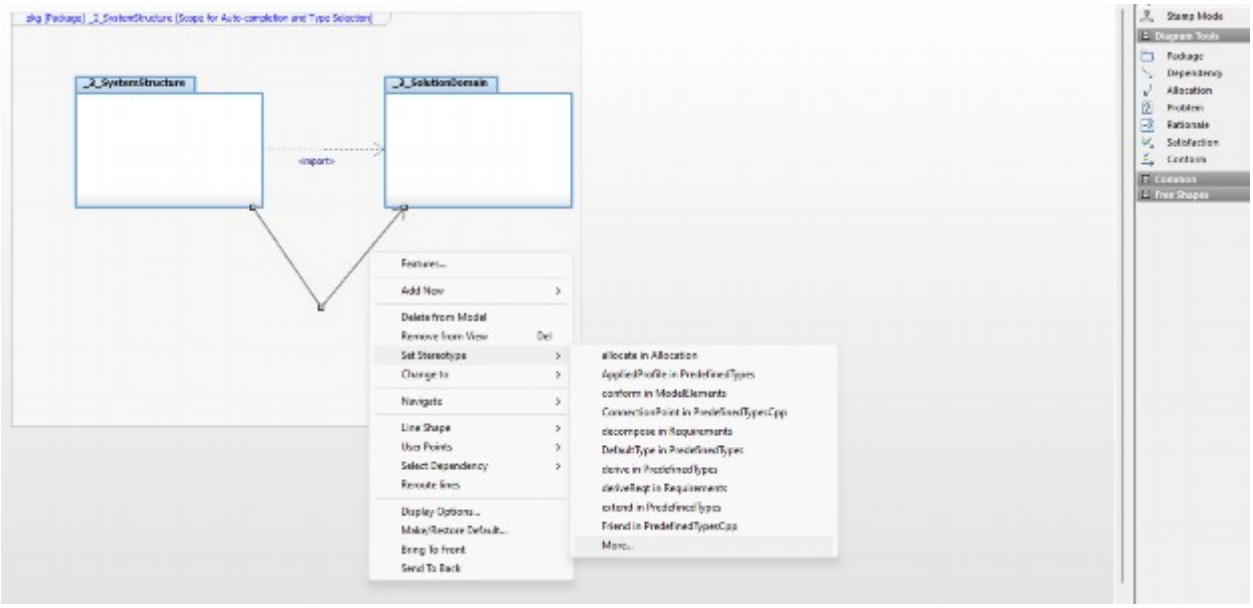

Рис. 2.9. Процес додавання стереотипу імпорту до залежності на діаграмі пакета

На рисунку 2.9 показано, як створити параметри в обмеженні.

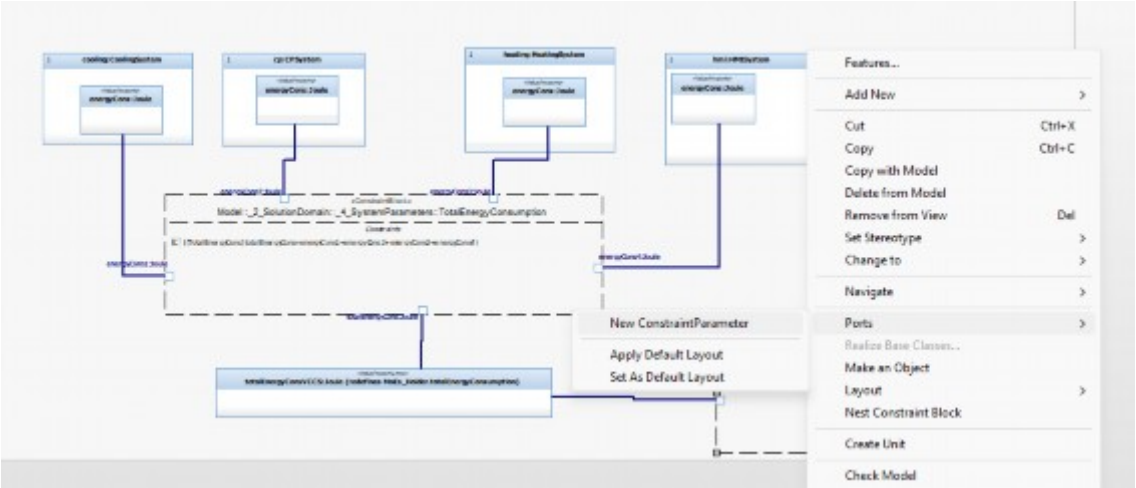

Рис. 2.9. Процес додавання нового параметру обмеження до блоку обмежень

Застосування профілю RAAML у середовищі Rhapsody не можна вважати тривіальним завданням, оскільки попередні дослідження з RAAML

вказували на необхідність використання API інструменту для інтеграції RAAML у SysML-інструменти. Однак, профіль RAAML вже доступний для Rhapsody. Додатково до цього, було опубліковано навчальний посібник, який охоплює основи моделювання за допомогою RAAML. У посібнику детально описується процес створення таблиць і діаграм для аналізу HAZOP, як показано на рисунках 2.10 і 2.11.

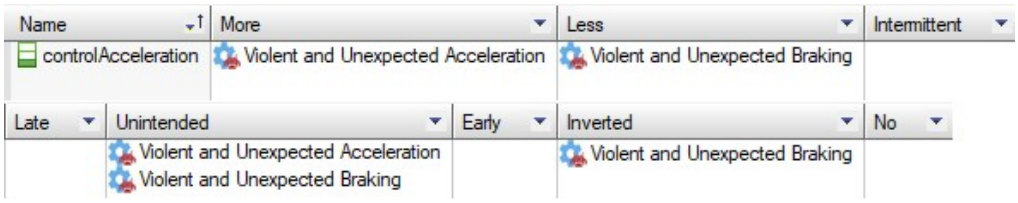

Рис. 2.10. Приклад таблиці аналізу RAAML HAZOP

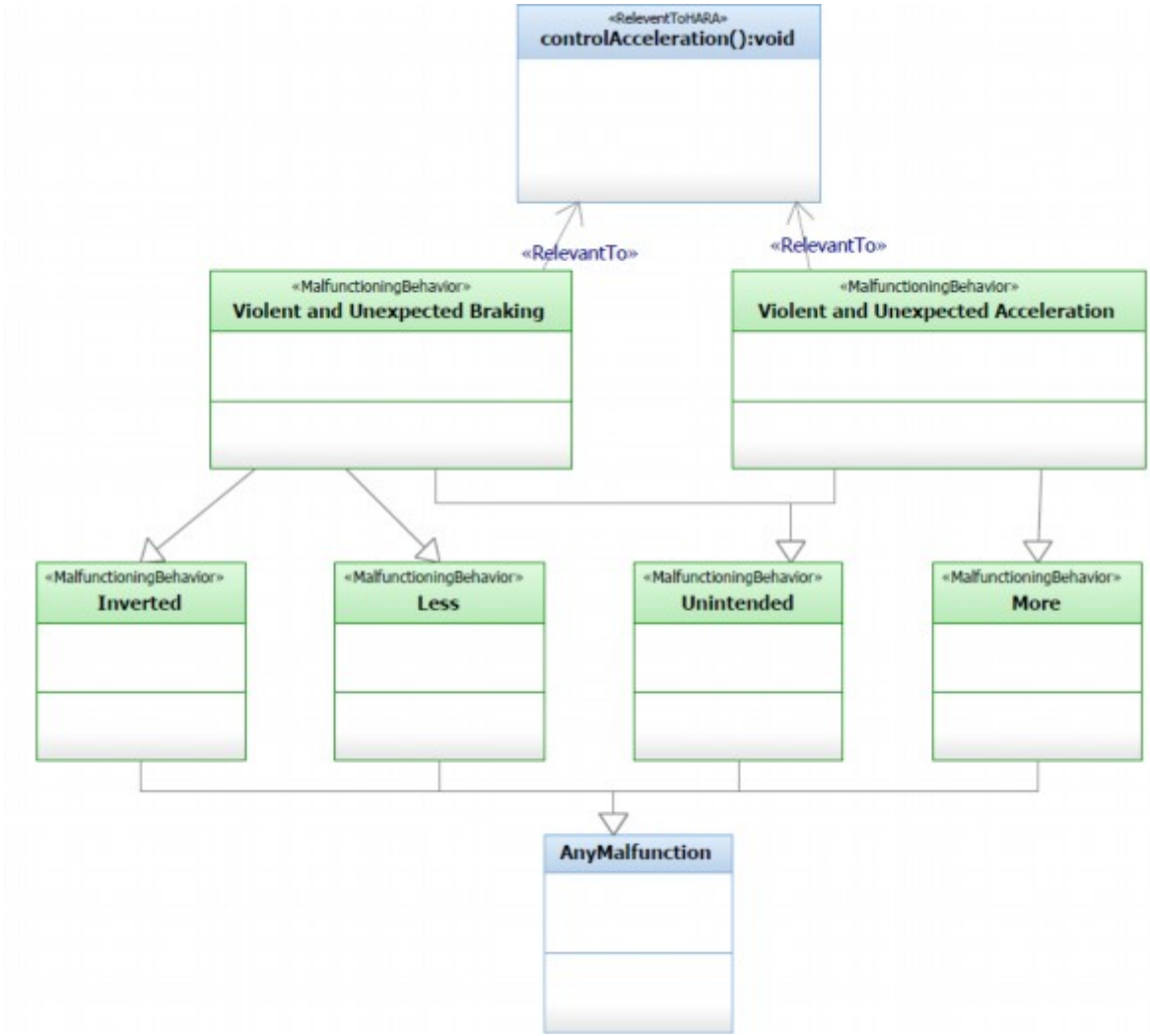

Рис. 2.11. Приклад діаграми RAAML HAZOP

Далі показано, як складається таблиця оперативних ситуацій (рис 2.12), таблиця сценаріїв аварій (рис. 2.13) і таблиця ефектів рівня автомобіля (рис. 2.14). Завдяки впливу, керованості та тяжкості, які були визначені у відповідних вищезгаданих таблицях, ASIL може бути автоматично розрахований для небезпечної події та відображений у підсумковій таблиці ASIL (рис. 2.15).

Name	Exposure	Justification of Exposure	Vehicle Usage
Motorway Driving at High Speed	E4	Very likely to occur in most day to day driving situations	Driving at Speed

Traffic and People	Location	Road Condition	Environmental Condition
High Speed Difference	Highway	AnyRoadCondition	AnyEnvironmentalCondition

Рис. 2.12. Приклад таблиці робочих ситуацій

Name	Description	Controllability	Justification of Controllability
Rear-end Collision		C3	Most drivers are unable to control the situation

Assumptions	Malfunctioning Behavior	Operational Situation
	Violent and Unexpected Braking	Motorway Driving at High Speed

Рис. 2.13. Приклад таблиці сценарію аварії

Name	Description	Severity	Justification of Severity
High Speed Collision		S3	

Рис. 2.14. Приклад таблиці ефектів (властивостей)

Hazardous Event	ASIL	Accident Scenario	Controllability
hazardous event_0	D	Rear-end Collision	C3

Situation	Exposure	Vehicle Level Effect	Severity
Motorway Driving at High Speed	E4	High Speed Collision	

Рис. 2.15. Приклад підсумкової таблиці

Окрім можливості створення таблиць і діаграм, описаних у підручнику для Rhapsody, профіль RAAML може бути застосований для управління вимогами в рамках робочого процесу, відповідного стандарту ISO 26262. Як показано на рисунку 2.16, профіль ISO 26262 RAAML містить бібліотеку для

управління вимогами, що включає стереотипи для моделювання цілей безпеки, вимог функціональної безпеки (FuSa), вимог тестування безпеки (TeSa), а також вимог до апаратного та програмного забезпечення з точки зору безпеки.

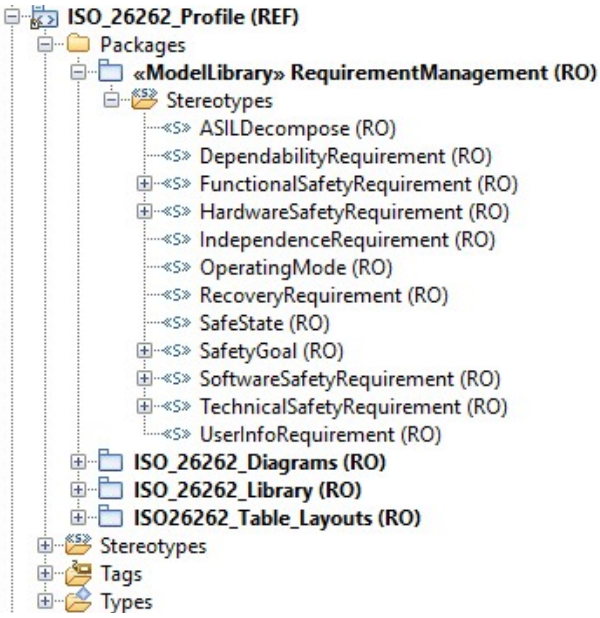

Рис. 2.16. Керування вимогами для бібліотеки ISO 26262

Висновки до розділу

В даному розділі було детально проаналізовано вибір методів, інструментів і мов моделювання, зроблений на основі їх переваг для забезпечення функціональної безпеки програмного забезпечення (FuSa). Основною причиною вибору мови SysML є її широке застосування у наукових дослідженнях, а також здатність підтримувати моделювання аспектів безпеки та надійності (S&R) за допомогою профілю RAAML. Це рішення дозволяє інтегрувати стандарти функціональної безпеки, зокрема ISO 26262, забезпечуючи відповідність вимогам безпеки в різних галузях, зокрема в автомобільній промисловості.

РОЗДІЛ 3. ІМПЛЕМЕНТАЦІЯ МОДЕЛЕЙ ТА АЛГОРИТМІВ ДЛЯ РЕАЛІЗАЦІЇ МЕТОДОЛОГІЇ ФУНКЦІОНАЛЬНОЇ БЕЗПЕКИ ПРОГРАМНИХ ДОДАТКІВ

3.1. Визначення термінології та термінів в методології

Першим етапом інтеграції стандарту ISO 26262 у методологію MagicGrid є узгодження термінології. Стандарт ISO 26262 допускає певну гнучкість у тлумаченні його елементів, що подано в таблиці 3.1.

Таблиця 3.1.

Терміни та визначення згідно ISO

Термін	**Визначення**
Пункт	Система або комбінація систем, до яких застосовується ISO 26262, яка реалізує функцію або частину функції на рівні транспортного засобу.
Система	Один з одним принаймні датчик, контролер і виконавчий механізм.
Компонент	Елемент несистемного рівня, який є логічно або технічно відокремленим і складається з більш ніж однієї апаратної частини або одного чи кількох програмних блоків.
Програмний блок	Програмний компонент атомарного рівня архітектури програмного забезпечення, який можна піддати автономному тестуванню.
Апаратна частина	Частина апаратного компонента на першому рівні ієрархічної декомпозиції .
Апаратна підчастина	Частина апаратної частини, яку можна логічно розділити та представляє другий або вищий рівень ієрархічної декомпозиції.
Підрозділ апаратного елемента	Найменша частина апаратної частини, яка розглядається в аналізі безпеки.

Зокрема, термін "Елемент" може відповідати як окремій системі, так і комбінації систем. Водночас система може складатися з набору компонентів або підсистем, проте підсистема не є чітко визначеним терміном у стандарті.

Компонент, у свою чергу, може бути технічно або логічно ізольованим, проте не обов'язково містити кілька програмних або апаратних частин; достатньо однієї одиниці або елемента. Незважаючи на неоднозначність у визначенні програмних та апаратних підчастин, стандарт надає чіткі та конкретні визначення цих понять.

Як видно з таблиці 3.2 методологія MagicGrid визначає свої терміни для елементів системи більше з точки зору моделювання порівняно зі стандартом ISO 26262. Крім того, зв’язок між елементами системи також є набагато чіткішим: система систем (SoS) може містити декілька систем. Ці системи складаються з підсистем, які потім знову складаються з компонентів. Нарешті, система, що аналізується, визначається як система інтересів (SoI).

Таблиця 3.2.

Терміни та визначення згідно MagicGrid

Термін	**Визначення**
Система систем	Системними елементами є самі системи. Як правило, це тягне за собою великомасштабні проблеми з кількома, гетерогенними, розподіленими системами.
Система відсотків	Система, життєвий цикл якої розглядається. Система, яка аналізується спочатку з точки зору чорного, а потім з точки зору білого ящика, щоб створити специфікацію системних вимог.
Система	Комбінація взаємодіючих елементів, які організовані в складне ціле для однієї чи кількох спільних цілей.
Підсистема	Другорядна або підпорядкована система в рамках більшої системи. Підсистема може бути зафіксована як блок у моделі.
Компонент	Елементарний об'єкт структури системи. Компоненти часто групуються в підсистеми системи; їх ідентифікують шляхом декомпозиції цих підсистем. Компонент може бути захоплений як блок у моделі.

Таблиця 3.3 показує результати відображення термінів ISO 26262 у MagicGrid. Було вирішено назвати цікаву систему елементом відповідно до

ISO 26262, щоб уникнути плутанини при використанні документів стандарту. Елемент може бути SoS або окремою системою; якщо це система, то елемент можна розглядати як концепцію системи. Для визначення системи було визначення MagicGrid вибраний тому, що він підкреслює, що це комбінація елементів, які організовані для спільної мети, таким чином залишаючи місце для визначення системи ISO 26262, яке потрібно скоригувати для визначення підсистеми: як системи в більшій системі, яка пов'язує принаймні одну датчик, контролер і виконавчий механізм один з одним. Подібно до підсистеми, компонент визначається скоригованим визначенням ISO 26262 як елемент у підсистемі, який є логічно або технічно відокремленим і складається з однієї або кількох апаратних частин або одиниць ПЗ.

Таблиця 3.3.

Зіставлення подібних концепцій між MagicGrid та ISO 26262

MagicGrid	**ISO 26262**	**Magic-V-Grid**	**Причина**
Система відсотків	Пункт	**Пункт**	Цікава система, до якої застосовано FuSa і яка реалізує одну функцію
Система	Елемент або система	**Система**	Комбінація взаємодіючих підсистем або - організованих для спільної мети.
Підсистема	Система або компонент	**Підсистема**	Система, яка пов'язує один з одним датчик, контролер і виконавчий механізм.
Компонент	Компонент	**Компонент**	Елемент, який логічно або технічно відокремлюється і складається з більш ніж однієї апаратної частини або одного чи кількох програмних блоків.

3.2. Інтеграція програмного життєвого циклу згідно ISO в методологію MagicGrid

Тепер, коли відома термінологія для комбінованого робочого процесу, робочий процес ISO 26262 можна зіставити з MagicGrid. Для початку пункти,

виділені на етапі концепції, можна зіставити з проблемною областю, оскільки обидва схожі у створенні концепції елемента або SoI. Подібним чином пункти розробки продукту можуть бути зіставлені з областю рішення. Результат цього показано на рис. 3.1.

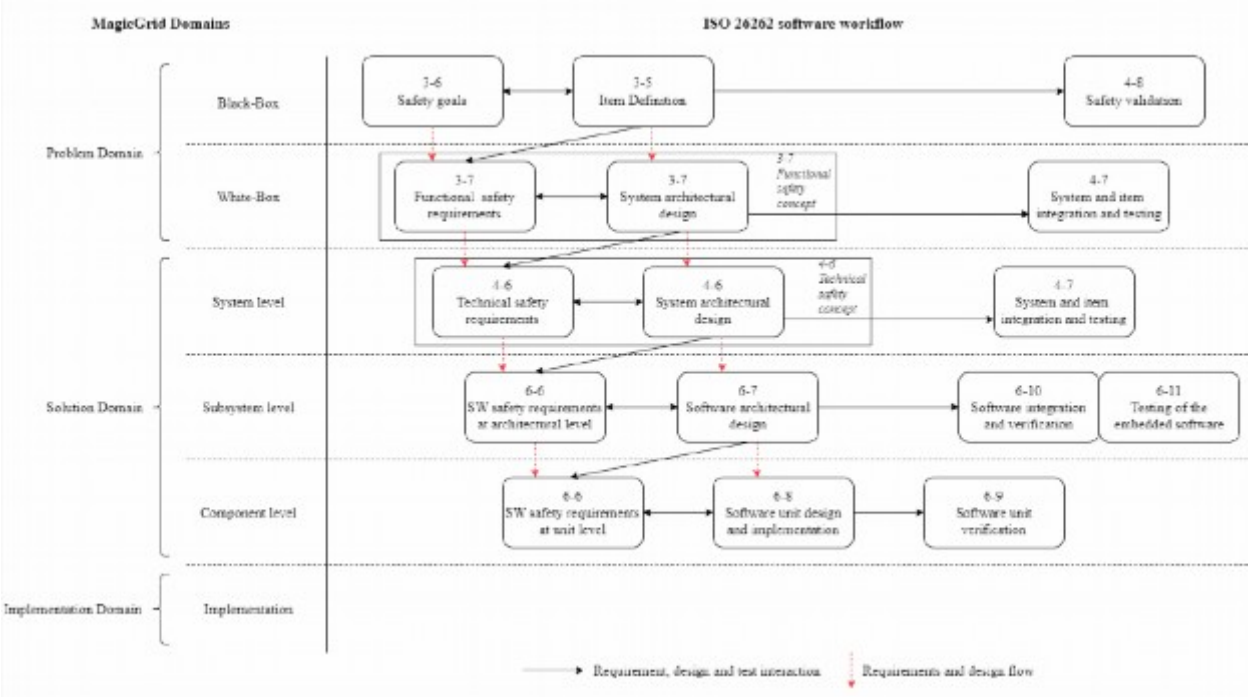

Рис. 3.1. Проектування програмного робочого процесу ISO 26262 на структуру доменів MagicGrid

У проблемній області визначення елемента та цілей HARA/безпеки розміщено на рівні "чорної скриньки", оскільки вони охоплюють лише вхідні та вихідні дані елемента та описують його взаємодію з навколишнім середовищем. Перевірка безпеки також віднесена до цього рівня, попри її походження з розробки продукту, оскільки вона спрямована на перевірку досягнення цілей безпеки.

Концепція функціональної безпеки (FuSa) відображена на рівні "білої скриньки", оскільки вона стосується внутрішньої структури елемента, включаючи попередню архітектуру. Інтеграція та тестування системи і компонентів розташовані як на рівні "білої скриньки", так і на системному

рівні проблемної області, оскільки вони перевіряють як концепцію FuSa, так і технічної безпеки (TeSa).

У рішенні концепція TeSa розміщена на системному рівні, відповідно до її визначення у стандарті ISO 26262. На рівні підсистеми відображено архітектуру програмного забезпечення разом із її перевіркою, інтеграцією та тестуванням, оскільки вона перебуває на рівні, нижчому за систему в рамках ISO 26262. Проектування програмних блоків розміщено на рівні компонентів, разом з архітектурним проектуванням та його перевіркою.

Останнім етапом інтеграції ISO 26262 у MagicGrid є спрощення робочого процесу шляхом використання уніфікованої термінології для кожного рівня. Це досягається двома шляхами: по-перше, використанням термінів, що охоплюють кілька пунктів стандарту, і по-друге, виключенням пунктів, які дублюються в MagicGrid. Результат цього процесу відображено на рисунку 3.2.

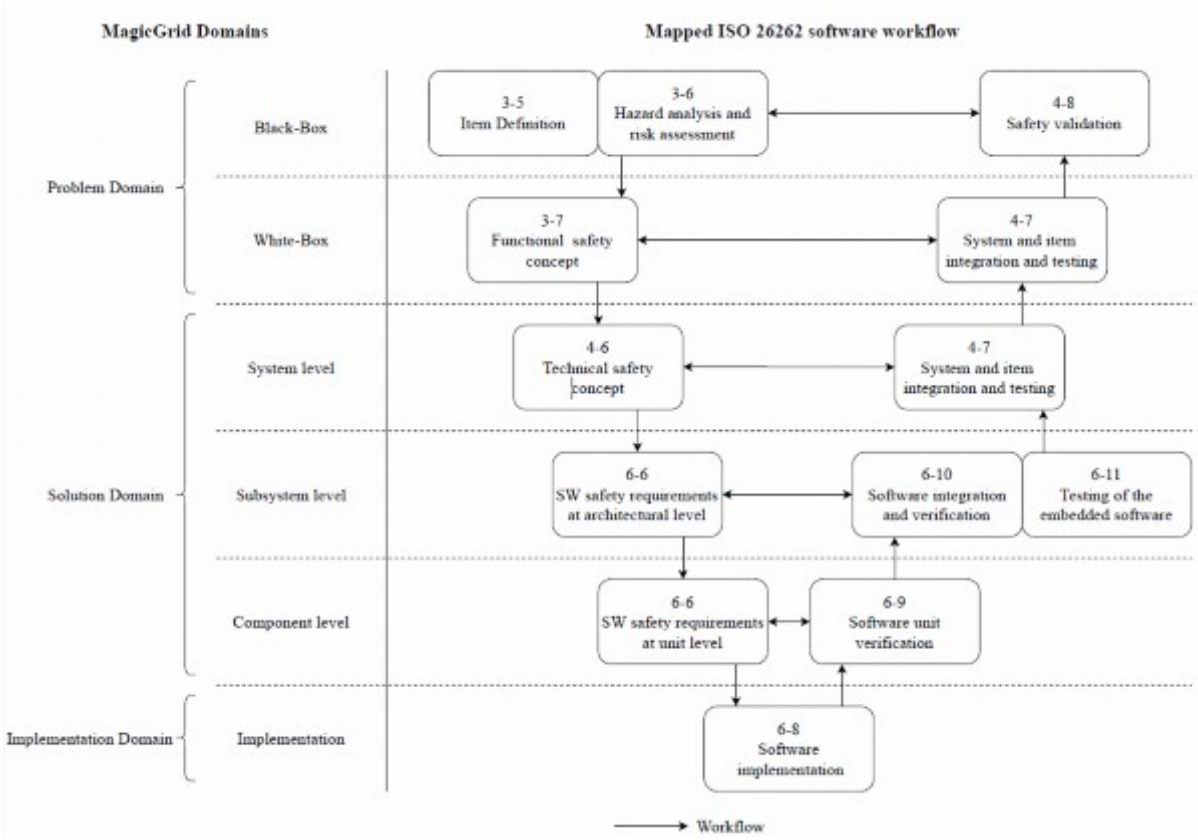

Рис. 3.2. Узагальнений робочий процес ISO 26262, зіставлений із доменами MagicGrid

Загалом, MagicGrid охоплює архітектурні конструкції та проектування елементів, тому було прийнято рішення не включати їхню оцінку до кінцевого методу. При цьому всі етапи на правій стороні V-моделі залишаються незмінними, як показано на рисунку 3.1.

Аналізуючи окремі домени MagicGrid, на рівні "чорної скриньки" визначення елемента розміщено перед етапом HARA, яке тепер замінює цілі безпеки. Такий порядок є більш логічним, оскільки визначення елемента слугує вхідними даними для HARA. Крім того, HARA точніше відображає процес визначення цілей безпеки, оскільки ці цілі формулюються на основі результатів HARA.

На рівні "білої скриньки" як вимоги FuSa, так і архітектурний дизайн системи представляються через концепцію FuSa, оскільки обидва ці аспекти разом формують дану концепцію, а MagicGrid вже охоплює архітектурний дизайн.

Подібним чином, на рівні системи вимоги TeSa та архітектурний дизайн системи відображаються через концепцію TeSa, оскільки остання включає обидва ці аспекти.

На рівнях підсистеми та компонентів відображені лише вимоги, оскільки MagicGrid вже охоплює архітектурні рішення та проектування елементів. Зрештою, етап реалізації програмного забезпечення було переміщено з рівня компонентів на рівень реалізації.

3.3. Реалізація структури методології для підвищення функціональної безпеки

Як показано на рисунку 3.3, запропонована структура Magic-V-Grid включає три домени та шість стовпів. У цьому розділі розглядаються комірки інфраструктури для кожного рівня субдомену, з урахуванням рекомендацій з MagicGrid Book of Knowledge та стандарту ISO 26262. Основними доменами є проблемна область, рішення та реалізація.

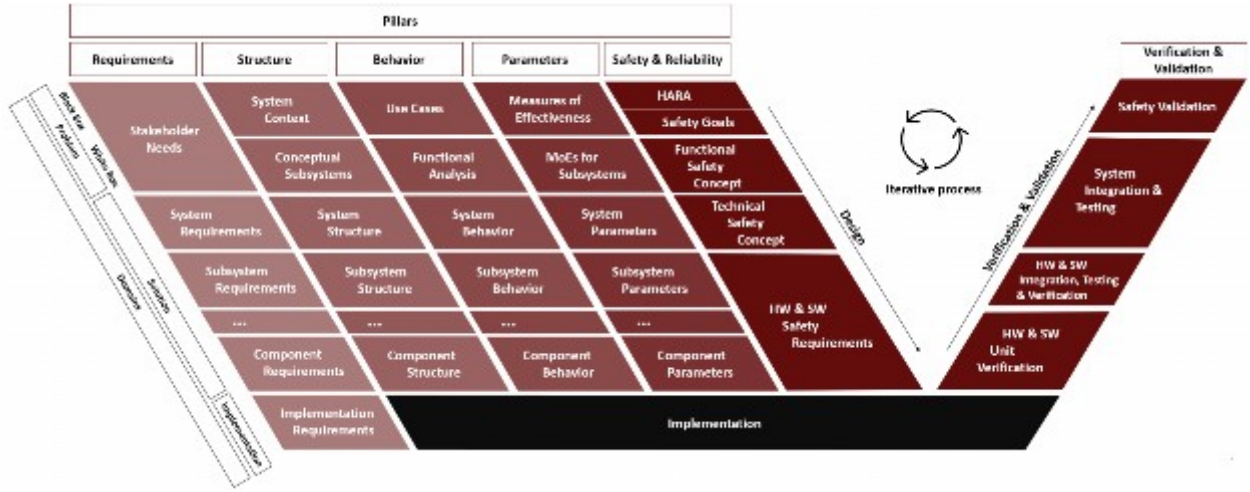

Рис. 3.3. Запропонована структура методології

Домен рішення спрямований на аналіз потреб зацікавлених сторін з метою чіткого визначення очікуваних функцій системи інтересу (SoI). Він поділяється на два субдомени: рівень "чорної скриньки" та рівень "білої скриньки". На рівні "чорної скриньки" моделюється взаємодія SoI із середовищем, зокрема з його вхідними та вихідними даними, а також з очікуваними функціями. Метою рівня "білої скриньки" є розуміння внутрішньої поведінки та структури SoI. Першим етапом цього рівня є визначення логічної архітектури через ідентифікацію концептуальних підсистем.

Домен рішення формує міждисциплінарну логічну архітектуру SoI для вирішення проблеми, визначеної у проблемній області. Оскільки одна проблема може мати кілька варіантів розв'язку, може бути створено декілька доменів рішення. Точність архітектури, розробленої в цьому домені, залежить від кількості ітерацій, що здійснюються на різних рівнях деталізації (від високого до низького). Домен рішення розподілений на три субдомени: рівень системи, рівень підсистеми та рівень компонентів, при необхідності можливе додавання додаткових рівнів.

У домені реалізації визначаються вимоги до фізичної реалізації SoI, що стосуються його фізичного дизайну.

3.3.1. Моделювання методом чорного ящика

Як показано на рисунку 3.4, процес MagicGrid розпочинається на рівні Black Box зі збору вимог зацікавлених сторін, після чого визначається контекст системи. У цьому контексті відображаються всі зовнішні елементи, що знаходяться поза межами об'єкта дослідження (SoI), включаючи їх взаємодію з SoI. Ідентифікація зовнішніх суб'єктів може бути здійснена шляхом аналізу потреб зацікавлених сторін та/або використанням діаграми, що базується на моделі з дев'яти блоків.

Рис. 3.4. Елементи моделювання чорного ящика

Далі, сценарії використання детальніше описують вимоги зацікавлених сторін, демонструючи, як SoI взаємодіє з зовнішніми елементами в межах визначеного контексту системи. Кількісні характеристики ефективності (Measures of Effectiveness, MoE) є уточненням нефункціональних вимог зацікавлених сторін і визначають ключові показники системи. Окрім цього, MoE можуть слугувати обмеженнями для перевірки відповідності дизайну системи. Етап Black Box завершується аналізом небезпек і оцінкою ризиків (HARA), що використовується для визначення цілей безпеки.

Як показано на рисунку 3.5, всі вихідні елементи рівня Black Box у MagicGrid сукупно формують визначення елемента відповідно до ISO. Ці елементи разом використовуються як вхідні дані для проведення HARA. Рисунок також демонструє, що робочий процес на цьому етапі є ітеративним: у разі виявлення недоліків на попередніх етапах, вони можуть бути додані та скориговані в подальшому процесі перевірки.

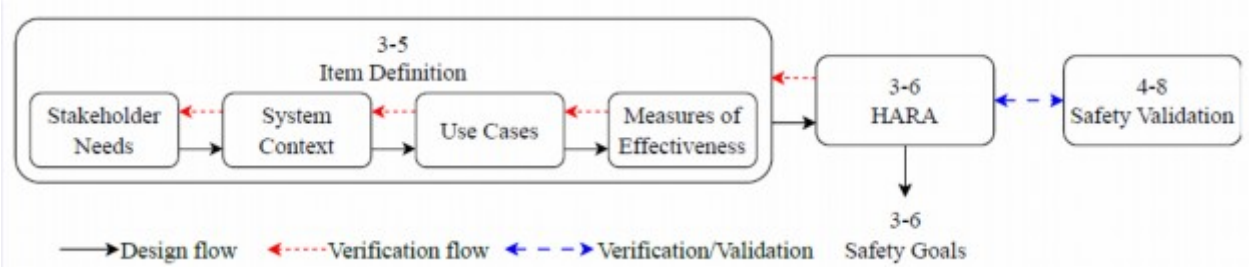

Рис. 3.5. Відношення під час моделювання методом чорного ящика

Перевірка безпеки має відбуватися після завершення інтеграції системи та елемента, щоб перевірити, чи відповідає розроблена система цілям безпеки. Водночас звіт про перевірку безпеки також є доказом того, що концепції FuSa та TeSa правильні для досягнення продукту функціональної безпеки. Перевірку можна виконати за допомогою численних тестів.

3.3.2. Моделювання методом білого ящика

На рівні білого ящика (рис. 3.6) модель системи будується шляхом декомпозиції функціональних вимог, ідентифікованих на рівні use case. Цей процес є ітеративним і триває доти, доки не буде досягнуто рівня деталізації, достатнього для подальшого аналізу та синтезу.

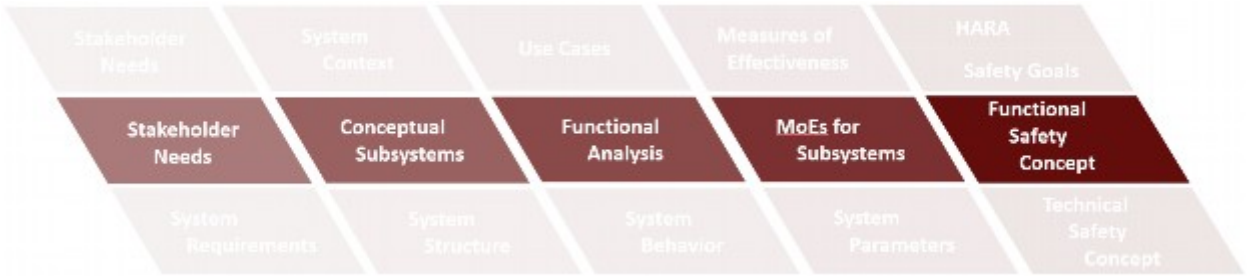

Рис. 3.6. Елементи моделювання білого ящика

Результатом є ієрархічна модель, яка відображає структуру системи у вигляді дерева, де кожен вузол представляє функціональний елемент. Зв'язки між вузлами відображають інформаційні та функціональні залежності між

елементами системи. Ця модель є основою для подальшого розроблення детальної архітектури системи.

Наступним кроком є моделювання концептуальних підсистем, інтегруючи дані з функціонального аналізу та аналізу контексту системи. Спочатку проводиться ідентифікація підсистем, до яких потім асоціюються функції, визначені в результаті функціонального аналізу. Взаємодії між підсистемами, а також їхні входи та виходи деталізуються на основі інформації з аналізу контексту системи, use case та функціонального аналізу.

Моделі поведінки (MoE) для підсистем є необов'язковими, якщо їх функціональність повністю визначається моделями на рівні чорного ящика.

Для кожної підсистеми формується концепція функціональної безпеки (FuSa), яка включає визначення нормальної та відхиленої поведінки, а також критеріїв для своєчасного виявлення та усунення можливих відмов. Як показано на рисунку 3.7, функціональна поведінка моделюється в рамках функціонального аналізу, а обмеження для цілей безпеки – в MoE підсистем. Таким чином, MoE можуть слугувати основою для формулювання вимог до функціональної безпеки та критеріїв їхньої верифікації. Для оцінки відповідності концепції FuSa цілям безпеки проводиться аналіз, який включає створення матриці простежуваності та виконання аналізу ризиків (HARA), подібного до FMEA, для елементів моделі білого ящика.

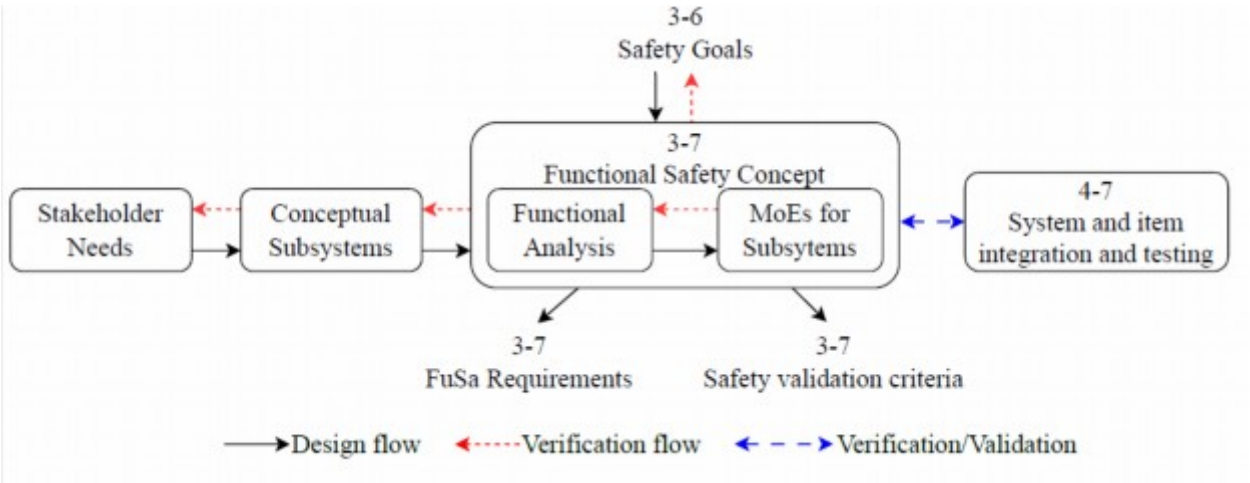

Рис. 3.7. Відношення під час моделювання методом білого ящика

Верифікація та валідація (V&V) системи на рівні білого ящика розпочинаються з інтеграції апаратних та програмних компонентів у функціональні блоки, системи та, зрештою, весь транспортний засіб. Мета V&V полягає у підтвердженні відповідності розробленої системи встановленим вимогам безпеки.

Для доведення виконання заходів безпеки застосовується комплекс тестових сценаріїв, які базуються як на вимогах безпеки, так і на моделюванні типових та аварійних ситуацій. Тестування включає в себе:

- Функціональне тестування: перевірка відповідності системи функціональним вимогам.

- Тестування безпеки: перевірка ефективності механізмів безпеки шляхом ін’єкції помилок та аналізу реакції системи.

- Тестування надійності: оцінка стійкості системи до збоїв та відмов шляхом проведення стрес-тестів та тестів на виснаження ресурсів.

- Тестування продуктивності: перевірка відповідності системи вимогам до швидкодії та часу відгуку.

Результати тестування документуються та використовуються для підтвердження відповідності транспортного засобу всім вимогам безпеки.

3.4. Моделювання системи та підсистеми на основі методології розробки програмних додатків Magic-V-Grid

Результати аналізу проблемної області підлягають формалізації у вигляді моделі вимог, яка документується в текстовому форматі (рис. 3.8). Між елементами моделі вимог та елементами проблемної області встановлюються трасування, що дозволяють відстежувати відповідність між вимогами та їхньою реалізацією в системі. Модель вимог є динамічним документом, який може уточнюватися та доповнюватися протягом усього життєвого циклу розробки системи. Після завершення розробки логічної

архітектури системи (LSA) проводиться верифікація відповідності між моделлю вимог та архітектурними рішеннями.

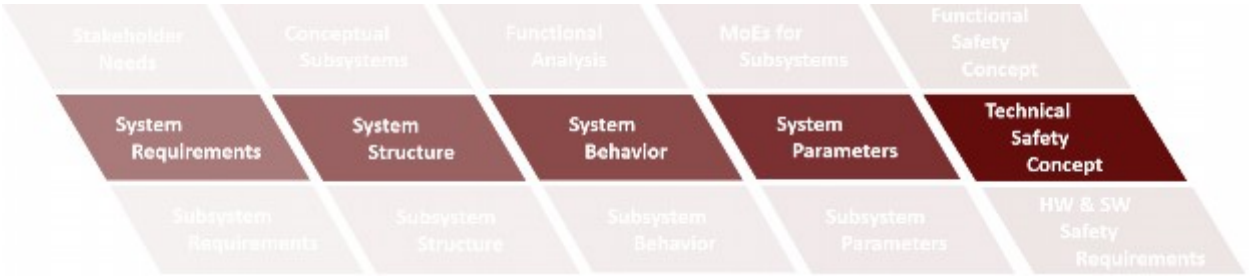

Рис. 3.8. Елементи моделювання системи

Логічна архітектура системи (LSA) визначає структуру системи на абстрактному рівні, розбиваючи її на логічні підсистеми. Слід зазначити, що логічні підсистеми не обов'язково корелюють з концептуальними підсистемами, визначеними на етапі аналізу проблемної області. Паралельно з визначенням логічних підсистем формується структура розподілу робіт між ними.

Поведінка системи в рамках LSA фіксується в окремій моделі. Однак, деталізація цієї моделі може бути відкладена до моменту уточнення логічної архітектури, оскільки передчасна деталізація може призвести до суперечностей з інтегрованою моделлю системи.

Системні параметри, які характеризують кількісні показники системи, використовуються для розрахунку показників ефективності (MoE). Ці параметри визначаються на основі аналізу проблемної області та використовуються для верифікації нефункціональних вимог. Для забезпечення можливості ранньої верифікації системи необхідно розробити математичні моделі, які дозволяють розрахувати MoE на основі значень системних параметрів.

Концепція технічної безпеки (TeSa) включає в себе як вимоги до безпеки, так і архітектурні рішення, спрямовані на забезпечення виконання цих вимог. Як показано на рисунку 3.9, концепція TeSa базується на

визначенні елементів системи, їхніх функцій безпеки (FuSa) та логічних підсистем.

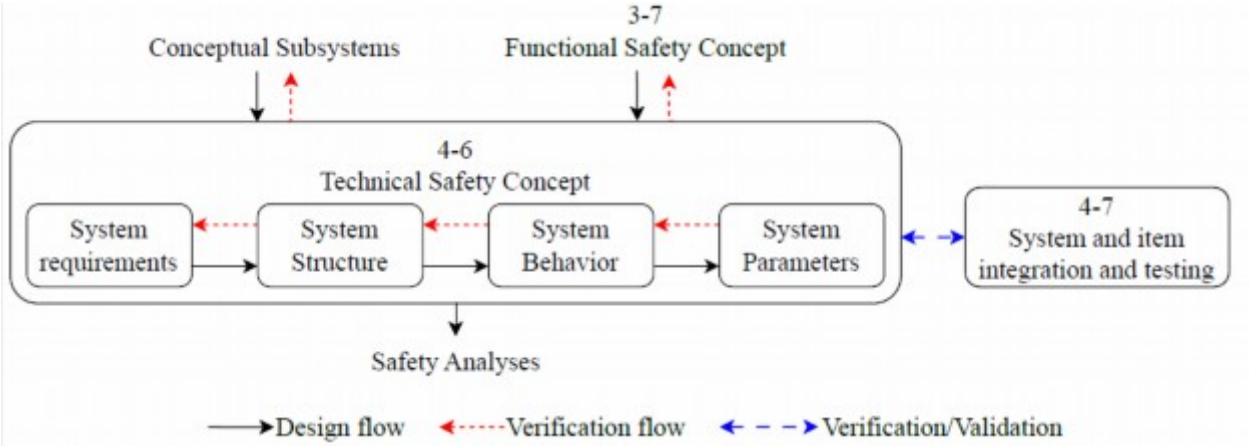

Рис. 3.9. Відношення в системі

Для верифікації концепції технічної безпеки (TeSa) необхідно провести детальний аналіз безпеки. Якісний аналіз дозволяє ідентифікувати потенційні відмови системи, тоді як кількісний аналіз оцінює ймовірність виникнення цих відмов. Вибір методу аналізу залежить від рівня безпеки ASIL. Для систем з високим рівнем ASIL рекомендується використовувати дедуктивний метод аналізу, такий як аналіз дерев відмов (FTA), а для систем з нижчим рівнем ASIL – індуктивний метод, наприклад, аналіз режимів відмов і їхніх ефектів (FMEA). Небезпеки, виявлені в процесі аналізу, які не були враховані на попередніх етапах, повинні бути додані до матриці аналізу ризиків (HARA) на рівні чорного ящика.

На етапі моделювання підсистеми за допомогою методу Magic-V-Grid специфікація вимог до підсистеми розробляється після завершення логічної архітектури системи (LSA) (рис. 3.10). Кожна вимога найнижчого рівня повинна бути трасована до системних вимог або елементів LSA. Після розробки логічної архітектури підсистеми (LSSA) встановлюються зв'язки між вимогами до підсистеми та елементами LSSA для забезпечення їхньої верифікації.

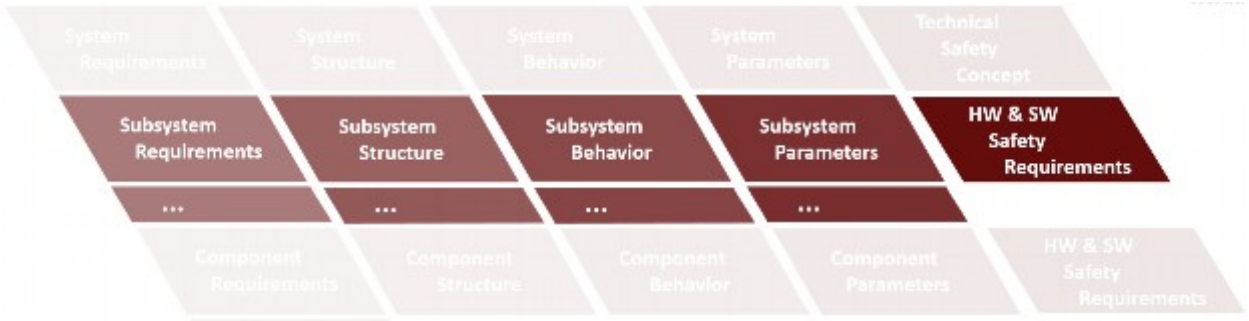

Рис. 3.10. Елементи моделювання підсистеми

У комірці структури підсистеми деталізується внутрішня структура підсистеми, включаючи визначення компонентів та їхніх взаємозв'язків. Інтерфейси між підсистемами, попередньо визначені на рівні логічної архітектури системи (LSA), можуть бути доповнені за необхідністю. Моделі поведінки підсистеми, які розробляються на цьому етапі, повинні відповідати функціональним вимогам, що висуваються до підсистеми. Після розробки моделей поведінки всіх підсистем проводиться їх інтеграція та верифікація. Комірка параметрів підсистеми містить кількісні характеристики, необхідні для розрахунку показників ефективності (MoE). Для забезпечення можливості ранньої верифікації системи розробляються математичні моделі, що пов'язують MoE з системними параметрами.

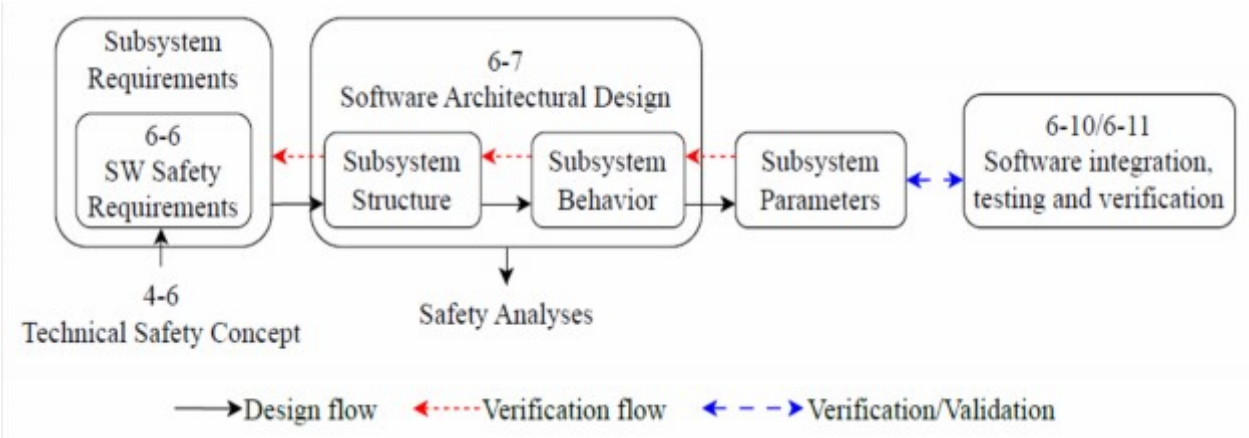

Рис. 3.11. Відношення в підсистемі

Як показано на рисунку 3.11, процес розробки програмного забезпечення, що забезпечує безпеку, починається з визначення вимог, включаючи специфікацію інтерфейсів апаратного та програмного забезпечення. Далі здійснюється проектування, оцінка, інтеграція та верифікація системи.

Для верифікації архітектурного проекту програмного забезпечення необхідно провести аналіз безпеки, зокрема, аналіз небезпек, ризиків та зменшення їх впливу (HARA). Додатково можуть застосовуватися інші методи, такі як моделювання та аналіз планування, детально описані в стандарті ISO.

Процес інтеграції та верифікації на рівні підсистеми аналогічний до процесу верифікації на рівні архітектури. Однак для тестування вбудованого програмного забезпечення рекомендується використовувати метод тестування "hardware-in-the-loop" (HIL), який дозволяє виконувати тести на основі вимог або проводити перевірку на виявлення помилок.

3.4.1. Моделювання компонентів

На рівні компонентного рішення логічна архітектура взаємопов'язана з логічним дизайном, що дозволяє її моделювання за допомогою підходу MBD (Model-Based Design) у поєднанні з MBSE (Model-Based Systems Engineering). Для цього можуть бути використані інструменти, такі як UML і MATLAB. Після завершення моделювання окремих компонентів здійснюється побудова конфігураційної моделі системи, яка інтегрує всі моделі об'єкта дослідження.

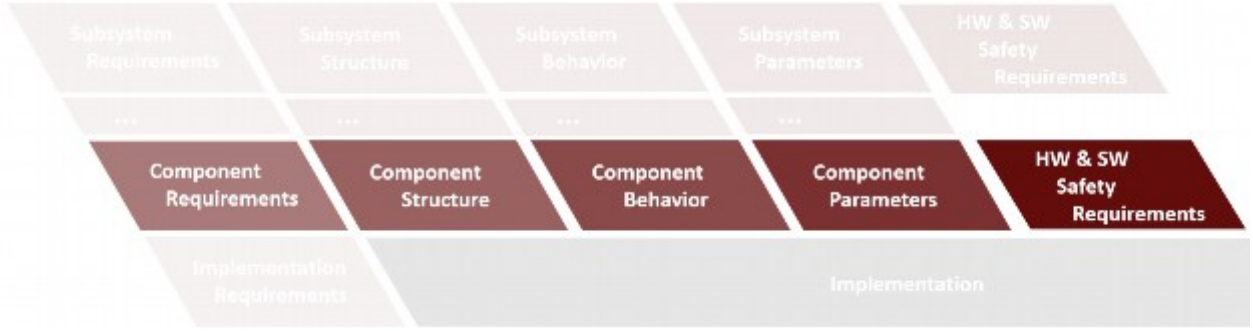

Рис. 3.12. Елементи моделювання компонентів

Рівень компонентів є подібним до рівня підсистем, оскільки допускається наявність необмеженої кількості рівнів. Відповідно, вміст усіх комірок MagicGrid на цьому рівні ідентичний вмісту на рівні підсистем. Під час верифікації та валідації програмних блоків (рис. 3.13), поряд із методами перевірки та тестами, зазначеними вище, необхідно також виконати тести покриття, щоб визначити, чи повністю вимоги охоплюють програмне забезпечення. У разі недостатнього охоплення слід додати нові вимоги та тестові випадки.

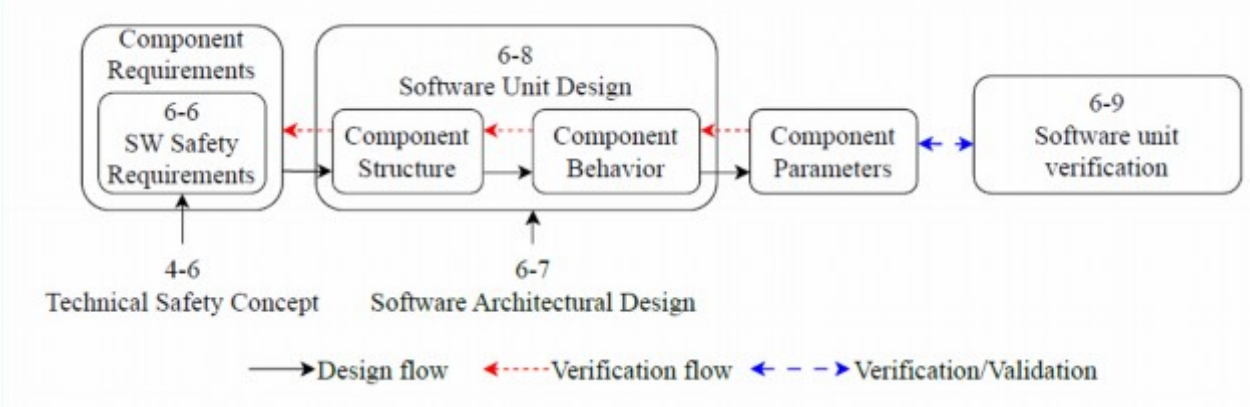

Рис. 3.13. Зв’язки компонентів

3.4.2. Моделювання імплементації

Вимоги до впровадження (рис. 3.14) включає детальні вимоги, що використовуються для фізичного проектування SoI на основі логічної архітектури. У випадку програмного забезпечення сама реалізація виконується шляхом генерації коду.

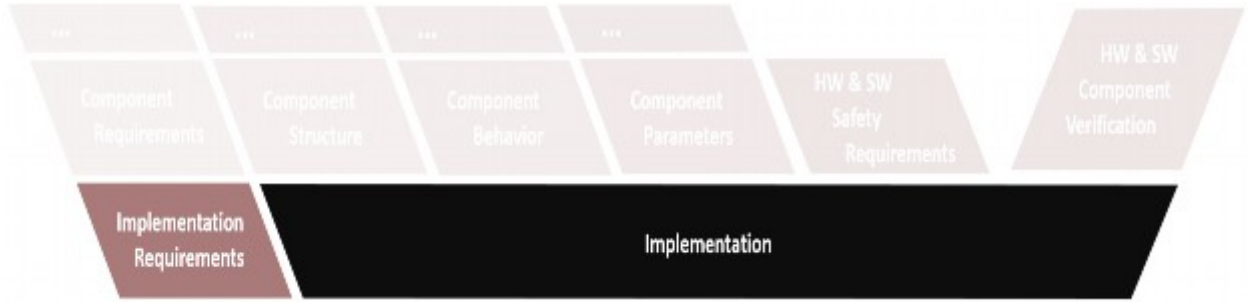

Рис. 3.14. Елементи моделювання реалізації

3.5. Алгоритмічна реалізація пропонованої методології

З метою демонстрації застосування блоків з бібліотеки Motar-toolbox була розроблена та реалізована модель системи автоматичного управління внутрішнім освітленням автомобіля.

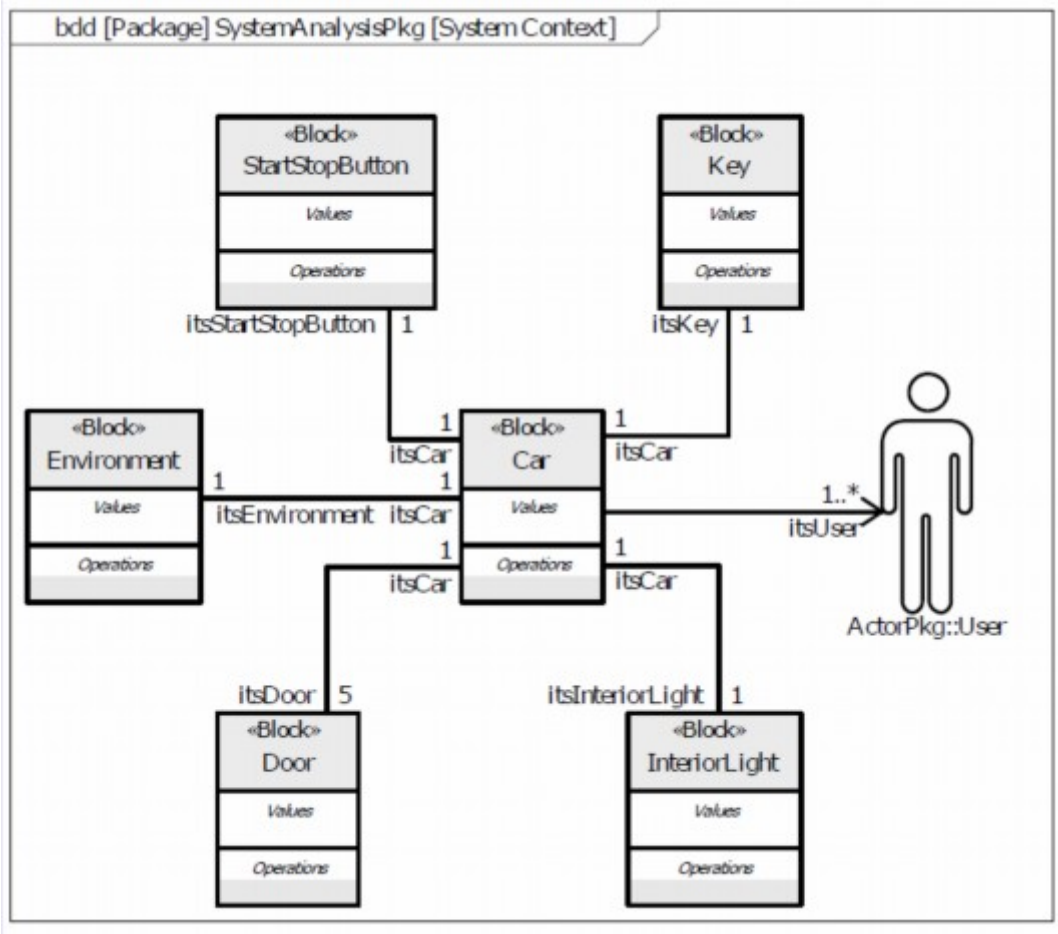

Рис. 3.15. Структурна схема для здійснення управління

На рисунку 3.15 представлена структурна схема розробленої системи. Модель включає наступні основні елементи:

- Кнопка «Пуск/Стоп»: Ініціює запуск або зупинку двигуна внутрішнього згоряння.

- Ключ запалювання: Замикає або розмикає електричний ланцюг автомобіля, забезпечуючи його живлення та функціонування.

- Двері: Можуть бути у двох станах: відкриті або закриті.

- Внутрішнє освітлення: Автоматично вмикається при відкриванні дверей та вимикається при їх закриванні або вимкненні двигуна.

Модель також відображає взаємодію користувача з системою та оточенням, в якому функціонує автомобіль.

Мета моделювання:

- Демонстрація можливостей бібліотеки Motar-toolbox для моделювання динамічних систем.

- Аналіз поведінки системи автоматичного управління внутрішнім освітленням в різних умовах експлуатації.

- Оптимізація алгоритмів управління для підвищення енергоефективності та комфорту користувачів.

Розроблена модель системи автоматичного внутрішнього освітлення є ефективним інструментом для дослідження та розробки нових рішень в області автомобільної електроніки.

Модель системи, представлена на рисунку 3.16, включає чотири основні сценарії використання: OpenDoor, OperateLock, OperateEngine та OperateLights. Останні три сценарії є розширеннями базового сценарію OperateLights, який описує базову функціональність системи управління внутрішнім освітленням.

Опис сценаріїв:

- OpenDoor: Сценарій моделює ситуацію відкривання дверей автомобіля. При активації цього сценарію система автоматично вмикає внутрішнє освітлення.

- OperateLock: Цей сценарій пов'язаний з операціями із замком запалювання. Активація цього сценарію може впливати на стан внутрішнього освітлення залежно від конкретної реалізації системи.

- OperateEngine: Сценарій моделює запуск та зупинку двигуна. Він також може впливати на стан внутрішнього освітлення, наприклад, автоматично вимикаючи його при вимкненні двигуна.

- OperateLights: Базовий сценарій, який описує пряму взаємодію користувача з вимикачем внутрішнього освітлення.

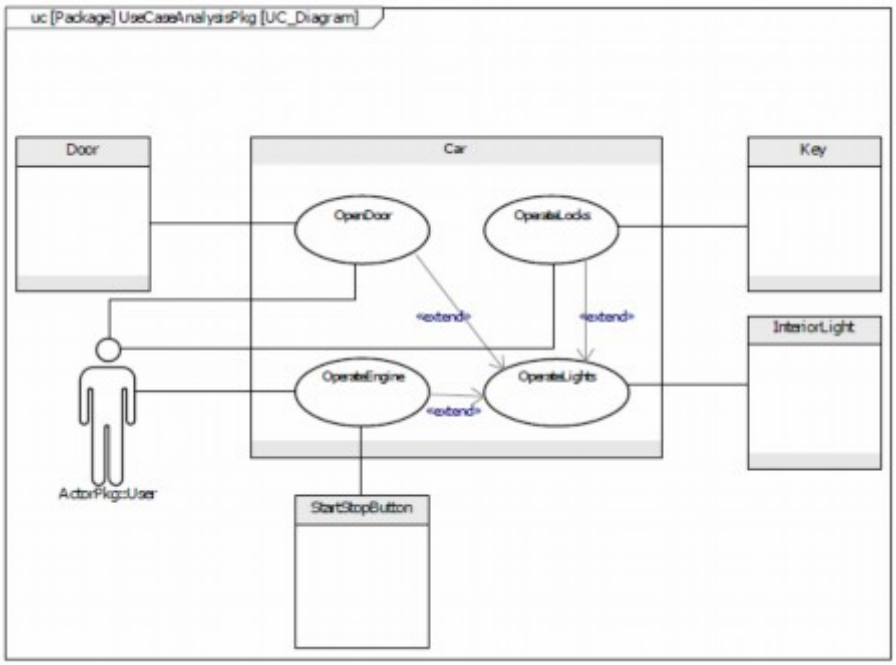

Рис. 3.16. Модель системи із сценаріями використання

Рисунок 3.16 наочно демонструє взаємозв'язки між різними блоками моделі та сценаріями використання. Він також відображає, з якими сценаріями безпосередньо взаємодіє користувач.

Після моделювання автоматичного внутрішнього освітлення в SysML система також була змодельована в Simulink за допомогою Motar-toolbox.

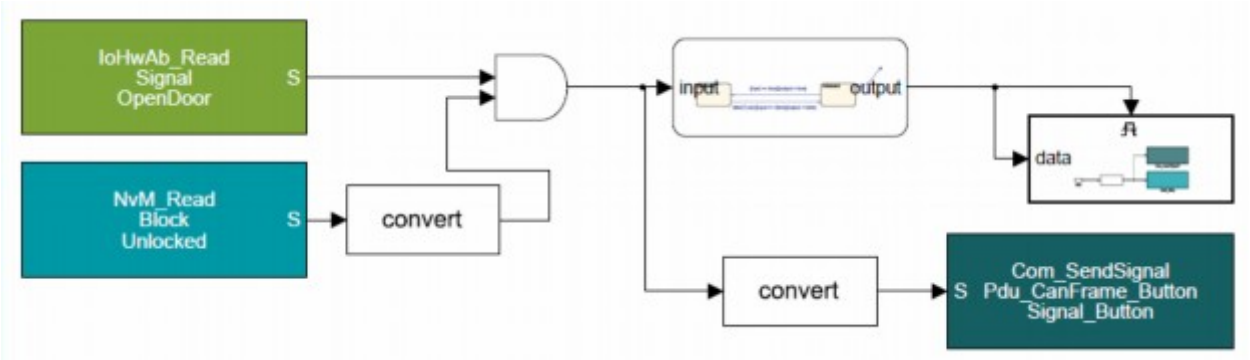

Рис. 3.17. Реалізація прецеденту (сценарію) OpenDoor

На рисунку 3.17 детально представлена реалізація сценарію відкривання дверей (OpenDoor). Даний сценарій передбачає зчитування сигналу про стан дверей з відповідного вхідного/вихідного пристрою. При відкритих дверях на вході фіксується високий рівень напруги, а при закритих – низький.

Паралельно з цим, зчитується інформація про стан замку автомобіля з блоку NvM (Non-Volatile Memory). Значення цього параметра (0 – розблоковано, 1 – заблоковано) зберігається у вигляді цілого числа та конвертується у логічний сигнал перед подальшою обробкою, оскільки блок NvM не підтримує логічні значення.

Отримані сигнали про стан дверей та замку об'єднуються за допомогою логічної операції І. Якщо обидва сигнали мають високий рівень (тобто двері відкриті і автомобіль розблокований), то на виході цієї операції формується сигнал, який подається на вхід блоку усунення дребезгу. Даний блок фільтрує короткочасні перешкоди (дребезг) у сигналі, забезпечуючи стабільну роботу системи. Якщо протягом 7 секунд на вході блоку усунення дребезгу зберігається високий рівень сигналу, то на його виході також формується високий рівень, який активує блок управління внутрішнім освітленням. В іншому випадку, світло вимикається.

Додаткові функціональні блоки:

- Com_SendSignal: Блоки цього типу використовуються виключно для цілей налагодження та відстеження стану системи. Вони дозволяють відправляти сигнали по шині CAN для подальшого аналізу.

- NvM LightState: Блок неперервної пам'яті, в який записується стан внутрішнього освітлення (включено/вимкнено) у вигляді цілого числа. Ця інформація може бути використана для діагностики системи та подальшого аналізу її роботи.

Розглянута реалізація сценарію OpenDoor демонструє типову структуру алгоритмів управління внутрішнім освітленням автомобіля. Використання

блоку усунення дребезгу дозволяє підвищити надійність системи та уникнути помилкових спрацювань.

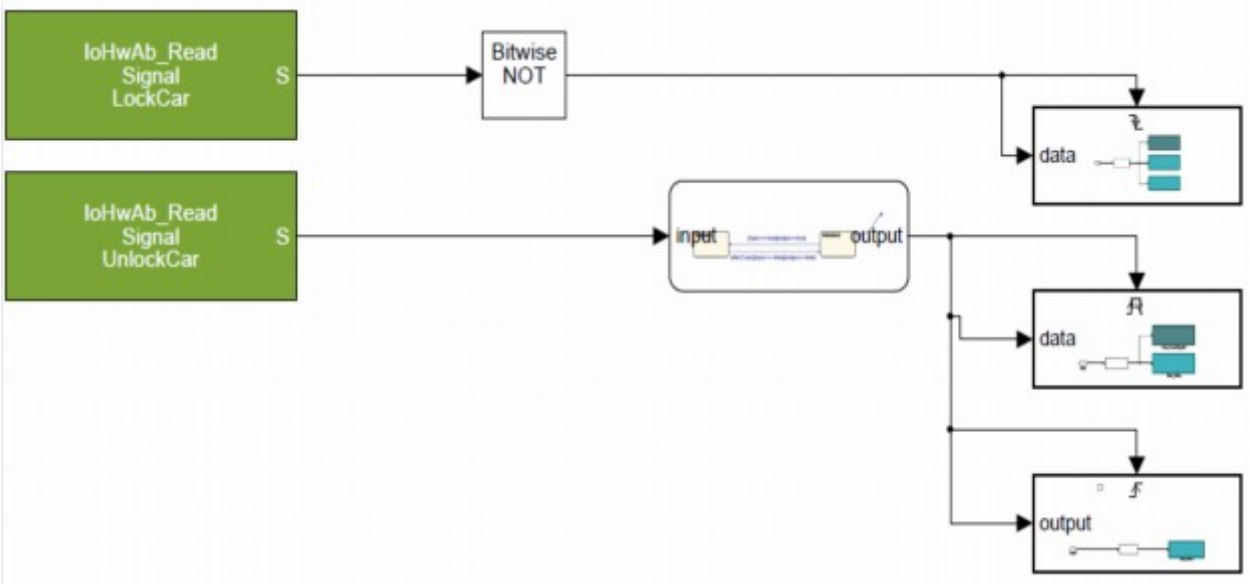

Рис. 3.18. Реалізація сценарію блокування/розблокування (OperateLock)

На рисунку 3.18 детально представлена реалізація сценарію блокування/розблокування автомобіля (OperateLock). Даний сценарій передбачає зчитування сигналу, що ініціює відповідну операцію.

Логіка роботи наступна:

- Блокування автомобіля: При отриманні сигналу блокування, він інвертується на логічне "0" (false). Це пов'язано з тим, що при блокуванні автомобіля сигнал, що відповідає за стан замку, зазвичай переходить у низький рівень. Отриманий сигнал записується в блок стану замку (LockState) та блок стану освітлення (LightState).

- Розблокування автомобіля: При отриманні сигналу розблокування, він пропускається через блок усунення дребезгу. Якщо сигнал стабільний протягом 7 секунд, то на виході блоку формується логічна одиниця (true), яка активує блок управління внутрішнім освітленням. Одночасно, значення в блоці LockState змінюється на "true" (розблоковано). Через 7 секунд після зняття сигналу розблокування, світло автоматично вимикається.

Функціональні блоки:

- LockState: Блок пам'яті, в якому зберігається інформація про стан замку автомобіля (заблоковано/розблоковано).

- LightState: Блок пам'яті, в якому зберігається інформація про стан внутрішнього освітлення (включено/вимкнено).

- Блок усунення дребезгу: Фільтрує короткочасні перешкоди в сигналі, забезпечуючи стабільну роботу системи.

Розглянута реалізація сценарію OperateLock демонструє взаємозв'язок між станом замку автомобіля та управлінням внутрішнім освітленням. Використання блоку усунення дребезгу забезпечує надійну роботу системи в умовах можливих перешкод.

Висновки до розділу

Отже, в цьому розділі представлено інтеграцію положень стандарту ISO 26262 щодо функціональної безпеки у методологію MagicGrid. Спочатку здійснено порівняння термінології та положень ISO з елементами MagicGrid. Далі продемонстровано взаємозв’язки між пунктами стандарту та іншими компонентами MagicGrid шляхом їх інтеграції. Розглянуто сценарії використання пропонованої методології.

ВИСНОВКИ

У магістерській роботі представлено методологію інтеграції функціональної безпеки програмного забезпечення в системну інженерію на основі моделі (MBSE). Для досягнення цієї мети дослідження розпочалося з огляду літератури, зосередженого на сучасному стані MBSE, функціональної безпеки програмного забезпечення (MBFS), а також загальних підходів до функціональної безпеки. В результаті виявлено, що визначення MBSE часто є нечіткими через велику кількість абревіатур. Крім того, у літературі недостатньо обґрунтовано використання MBSE, приклади не охоплюють повний життєвий цикл розробки або є занадто загальними, що вказує на потребу галузі у більшій адаптивності для досягнення успіху.

Щодо функціональної безпеки, було встановлено, що стандарт вже містить рекомендації щодо застосування моделі управління проектуванням (MBD) для генерації коду, особливо у випадку з платформами, такими як Simulink, яка широко використовується для розробки програмного забезпечення двигунів.

Після проведення літературного огляду був здійснений вибір інструментів та методів MBSE. Мовою моделювання було обрано SysML, оскільки RAAML підтримується виключно в поєднанні з цією мовою. Серед інструментів MBSE було обрано IBM Rhapsody, завдяки доступності ліцензії та попередньому досвіду роботи з цим інструментом.

Для полегшення інтеграції було створено профіль SysML, що спрощує моделювання діаграм і елементів в Rhapsody. Це забезпечило можливість інтеграції робочого процесу розробки програмного забезпечення, що призвело до створення нової методології. Цей процес передбачав зіставлення етапів розробки з доменами і консолідацію робочих етапів в меншу кількість комірок, що полегшило їх інтеграцію.

У процесі оцінювання Magic-V-Grid було порівняно з іншими методологіями, запропонованими в літературі, що підтвердило аналогічні

підходи в інших дослідженнях. Однак розроблену методологію поки що не було повністю протестовано в практичних умовах, що унеможливлює остаточний висновок щодо її ефективності та відповідності початковим цілям дослідження.

Таким чином, можна зробити висновок, що пропонована методологія демонструє можливість інтеграції функціональної безпеки у процес MBSE. Однак для її повної оцінки та порівняння з іншими підходами необхідні подальші дослідження та практичні випробування.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. O. Burkacky, J. Deichmann, M. Guggenheimer and M. Kellner, Outlook on the automotive software and electronics market through 2030, Jan. 2023. [Online]. Available: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/mapping-theautomotive-software-and-electronics-landscape-through-2030.
2. P. J. Younse, J. E. Cameron and T. H. Bradley, “Comparative analysis of a model-based systems engineering approach to a traditional systems engineering approach for architecting a robotic space system through knowledge categorization,” Systems Engineering, vol. 24, pp. 177–199, 3 May 2021, issn: 15206858. doi: 10.1002/sys.21573.
3. ISO 26262 | Rapita Systems Available: https://www.rapitasystems.com/iso26262
4. I. Group, Motar. verkort uw time-to-market. [Online]. Available: https://www.ict.eu/nl/producten/motar-verkort-uw-time-to-market.
5. I. Group, Ict group verlengt samenwerking met inmotion, Jan. 2021. [Online]. Available: https://www.ict.eu/nl/newsroom/nieuws/ict-group-verlengt-samenwerking-metinmotion.
6. I. Group, Aanzienlijk sneller ontwikkelen met motar, Dec. 2021. [Online]. Available: https://www.ict.eu/nl/projecten/aanzienlijk-sneller-ontwikkelen-met-motar.
7. I. Group, Gre dankzij motar binnen vijf maanden van diesel naar elektrisch, Jan. 2023. [Online]. Available: https://www.ict.eu/nl/projecten/gre-dankzij-motar-binnenvijf-maanden-van-diesel-naar-elektrisch.
8. Omg, Risk analysis and assessment modeling language (raaml) libraries and profiles, 2021. [Online]. Available: http://www.omg.org/legal/tm_list.htm..
9. A. Ahlbrecht and O. Bertram, “Evaluating system architecture safety in early phases of development with mbse and stpa,” ISSE 2021 - 7th IEEE

International Symposium on Systems Engineering, Proceedings, Sep. 2021. doi: 10.1109/ISSE51541.2021.9582542.

10. A. Berres, A. Armonas, T. Juknevičius, K. Post, M. Hecht and aeroorg Dave Banham, “Omg raaml standard for model-based fault tree analysis,” 2021.
11. K. Gadd, TRIZ for engineers: Enabling inventive problem solving. Wiley, 2011.
12. H. Sillitto, J. Martin, D. McKinney et al., Systems engineering and system definitions, Jul. 2019. [Online]. Available: https://www.incose.org/docs/ default- source/defaultdocument-library/final_-se-definition.pdf.
13. . Weilkiens, Definition of MBSE, Jan. 2022. [Online]. Available: https://mbse4u.com/2022/01/04/definition-of-mbse/.
14. L. J. Delligatti, SysML Distilled: A Brief Guide to the Systems Modeling Language. 2013.[Online]. Available: https://www.semanticscholar.org/ paper/60e2965d3e726cfe096fab0b9d9e47043bfe3466.
15. J. Shepard, What acronyms relate to mbse? Sep. 2022. [Online]. Available: https://www.analogictips.com/what-acronyms-relate-to-mbse-faq/.
16. E. Carroll and R. J. Malins, “Systematic literature review: How is model-based systems engineering justified?.,” 2016. doi: 10.2172/1561164. [Online]. Available: https://www.semanticscholar.org/paper/be565bf14cc 409712a4a8a0ececb1a78440d2582.
17. A. Madni and M. Sievers, “Model-based systems engineering: Motivation, current status, and research opportunities,” Systems Engineering, vol. 21, pp. 172–190, 2018. doi: 10 .1002 / sys . 21438. [Online]. Available: https://www.semanticscholar.org/paper/b5772c00a9a18f2099c9e5d9e58e9.
18. K. Henderson and A. Salado, “Value and benefits of model-based systems engineering (mbse): Evidence from the literature,” Systems Engineering, vol. 24, pp. 51–66, 2020. doi: 10 . 1002 / sys . 21566. [Online]. Available: https://www.semanticscholar.org/paper/2ca8f973591e9454095e61b40a9ce.

19. J. Maurandy, A. Helm, E. Gill and R. Stalford, “Cost-benefit analysis of sysml modelling for the atomic clock ensemble in space (aces) simulator,” INCOSE International Symposium, vol. 22, null, 2012. doi: 10.1002/j.2334-5837.2012.tb01433.x. [Online]. Available:https://www.semanticscholar.org/paper/a7c549f7e48c9394c173bc8d1695baabd0c703c4.
20. T. Bayer, “Is mbse helping? measuring value on europa clipper,” 2018 IEEE Aerospace Conference, vol. null, pp. 1–13, 2018. doi: 10.1109/AERO.2018 [Online]. Available: https://www.semanticscholar.org/paper/5b1ef840b25a2fd55963c9fe2a038f5d0e986ff8.
21. G. Liebel, M. Tichy and E. Knauss, “Use, potential, and showstoppers of models in automotive requirements engineering,” Software & Systems Modeling, pp. 1–21, 2019. doi:10.1007/s10270- 018- 0683- 4. [Online]. Available: https://www.semanticscholar.org/paper/af694c5aac48ffd4609d.
22. R. Maschotta, A. Wichmann, A. Zimmermann and K. Gruber, “Integrated automotive requirements engineering with a sysml-based domain-specific language,” 2019 IEEE International Conference on Mechatronics (ICM), vol. 1, pp. 402–409, 2019. doi: 10 . 1109 /ICMECH.2019.8722951. [Online]. Available: https://www.semanticscholar.org/paper/79a89f16b6bb.
23. P. Younse, J. E. Cameron and T. H. Bradley, “Comparative analysis of a model-based systems engineering approach to a traditional systems engineering approach for architecting a robotic space system through knowledge categorization,” Systems Engineering, vol. 24, pp. 177–199, 2021. doi:10.1002/sys.21573. Available: https://www.semanticscholar.org/paper/19c16167b8bf7918255900ea0522682a8aaf99e6.
24. J. Ma, G. Wang, J. Lu, H. Vangheluwe, D. Kiritsis and Y. Yan, “Systematic literature review of mbse tool-chains,” Applied Sciences, vol. null, null, 2022. doi: 10.3390/app12073431. [Online]. Available: https://www.semanticscholar.org/paper/5097d7a48b165423790209425ed689e8c3506e59.

25. Iso 21448:2022(en), road vehicles — safety of the intended functionality. [Online]. Available: https://www.iso.org/obp/ui/en/#iso:std:iso:21448:ed-1:v1:en.
26. Iso/sae 21434:2021(en), road vehicles— cybersecurity engineering. [Online]. Available: https: //www.iso.org/obp/ui/en/#iso:std:iso-sae:21434:ed-1:v1:en.
27. M. W. Group, Model quality objectives - embedded software development with matlab/simulink, [Online]. Available: https://www.mathworks.com/content/dam/mathworks/white-paper/mqo-paper-v1-0.pdf.
28. T. Erkkinen, How to use simulink for ISO 26262 projects, 2022. [Online]. Available: https://nl.mathworks.com/company/technical-articles/how-to-use-simulink-for-iso-26262-projects.html.
29. L. Rosqvist, An iso 26262 workflow for automated driving applications using matlab: Guidelines and best practices, 2020. [Online]. Available: https://nl.mathworks.com/company/technical-articles/an-iso-26262-workflow-for-automated-driving-applicationsusing-matlab-guidelines-and-best-practices.html.
30. B. Hendrix, T. Lewis, M. Emery and B. Rachele, “Model based functional safety – how functional is it?” Journal of System Safety, vol. null, null, 2022. doi: 10.56094/jss.v57i2.192. [Online]. Available: https://www.semanticscholar.org/paper/ab310781dfe16449a9cf0000a29f3d469e622c84.
31. D. Szymański, M. K. Scharrer, G. Macher, E. Armengaud and H. Schmidt, “Model-based functional safety engineering,” 2018. doi: 10.1007/978-3-319-57445-5_2. [Online]. Available: https://www.semanticscholar.org/paper/7effa2b958f438bc224a823770ee1c4debee6243.
32. F. R. Franco, M. Mauro, S. Stevan, A. B. Lugli and W. Torres, “Model-based functional safety for the embedded software of automobile power window system,” 2014 11th IEEE/IAS International Conference on Industry Applications, vol. null, pp. 1–8, 2014. doi: 10.1109/

INDUSCON.2014.7059430. Available: https://www.semanticscholar.org/paper/c84c4eab3b27ed0879a54210ff63de4c848b0edc.

33. Y. Luo, A. Saberi and M. Brand, “Safety-driven development and iso 26262,” 2019. doi:10.1007/978-3-030-12157-0_10. [Online]. Available: https://www.semanticscholar.org/paper/2355c8926293849096a590e375fe7172c1698b0c.
34. A. Buczacki and P. Piątek, “Proposal for an integrated framework for electronic control unit design in the automotive industry,” Energies, vol. null, null, 2021. doi: 10.3390/en14133816. [Online]. Available: https://www.semanticscholar.org/paper/72914d8a1c5f48050745328ba03725d797721f6a.
35. M. A. Skoglund, F. Warg and B. Sangchoolie, “In search of synergies in a multi-concern development lifecycle: Safety and cybersecurity,” 2018. doi: 10.1007/978-3-319-99229-7. Available: https://www.semanticscholar.org/paper/668258d8334f5a0fa018c3e0924f9180fe8e8fd7.
36. G. Biggs, T. Juknevičius, A. Armonas and K. Post, “Integrating safety and reliability analysis into mbse: Overview of the new proposed omg standard,” INCOSE International Symposium, vol. 28, null, 2018. doi: 10.1002/j.2334-5837.2018.00551.x. [Online]. Available: https://www.semanticscholar.org/paper/eb593aa2156aee657462e12e7420a4682f186c6a.
37. A. Berres, K. Post, A. Armonas, M. Hecht, T. Juknevičius and D. Banham, “Omg raaml standard for model-based fault tree analysis,” INCOSE International Symposium, vol. 31, null, 2021. doi: 10.1002/j.2334-5837.2021.00905.x. [Online]. Available: https://www.semanticscholar.org/paper/c205a4c02062152e51f9eeef557249bf516ff4e6.
38. A. Ahlbrecht and U. Durak, “Integrating safety into mbse processes with formal methods,” 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC), vol. null, pp. 1–9, 2021. doi: 10 . 1109 / dasc52595 . 2021. 9594315. [Online]. Available: https : / / www.semanticscholar.org/paper/622b7c442b477da40e6afb40736018e89bc6aaf6.

39. M. Schäfer, A. Berres and O. Bertram, “Integrated model-based design and functional hazard assessment with sysml on the example of a shock control bump system,” CEAS Aeronautical Journal, vol. 14, pp. 187–200, 2022. doi: 10.1007/s13272-022-00631-0. Available: https://www.semanticscholar.org/paper/98f6684cb1bfab876acf643030cc507f7e617e2a.
40. P. de Saqui-Sannes, R. Vingerhoeds, C. Garion and X. Thirioux, “A taxonomy of mbse approaches by languages, tools and methods,” IEEE Access, vol. 10, pp. 120 936–120 950, 2022. doi: 10.1109/ACCESS.2022.387.Available: https://www.semanticscholar.org/paper/e94dfdf74df68cd31d15fb558e229d150c0d5b35.
41. A. Khandoker, S. Sint, G. Gessl et al., “Towards a logical framework for ideal mbse tool selection based on discipline specific requirements,” J. Syst. Softw., vol. 189, p. 111 306, 2022. doi: 10.1016/j.jss.2022.111306. [Online]. Available: https://www.semanticscholar.org/paper/e5380104a897a1ff67051.
42. T. Weilkiens, A. Scheithauer, M. D. Maio and N. Klusmann, “Evaluating and comparing mbse methodologies for practitioners,” 2016 IEEE International Symposium on Systems Engineering (ISSE), vol. null, pp. 1–8, 2016. doi: 10.1109/SYSENG.2016.7753174. [Online]. Available: https://www.semanticscholar.org/paper/f888015551ece044120ef7e91b67708d638581fe.
43. S. Alai, “Evaluating arcadia/capella vs. oosem/sysml for system architecture development,” 2019. doi: 10.25394/PGS.8301050.V1. [Online]. Available: https://www.semanticscholar.org/paper/1b32d6bfa18e295a8bba6a720482c828ef05ca22.
44. M. D. Maio, T. Weilkiens, O. Hussein et al., “Evaluating mbse methodologies using the femmp framework,” 2021 IEEE International Symposium on Systems Engineering (ISSE), vol. null, pp. 1–8, 2021. doi: 10.1109/ISSE51541.2021.9582465. [Online]. Available: https://www.semanticscholar.org/paper/2d3dea023e95c9010f53cae3b04579debea8aee2.

45. C. Ponsard and V. Ramon, “Applying and extending femmp to select an adequate mbse methodology,” 2022. doi: 10.5220/0011312200003266. [Online]. Available: https://www.semanticscholar.org/paper/ec6e9194c91f.
46. L. Brisacier-Porchon, O. H. Ensta and R. Boutemy, “Modeling a uav in practice: A comparison between rhapsody and capella,” 2021 IEEE International Symposium on Systems Engineering (ISSE), vol. null, pp. 1–8, 2021. doi: 10.1109/ISSE51541.2021.9582553. [Online]. Available: https://www.semanticscholar.org/paper/082ce805c7723566f3da61636c013f6be94bb636.
47. D. Mazeika, A. Morkevicius and A. Aleksandraviciene, “Mbse driven approach for defining problem domain,” 2016 11th System of Systems Engineering Conference (SoSE), vol. null, pp. 1–6, 2016. doi: 10.1109/SYSOSE.2016.7542911. [Online]. Available: https://www.semanticscholar.org/paper/d5e19304dc1dd48731364a6db389e46e2ae6d27f.
48. A. Morkevicius, A. Aleksandraviciene, D. Mazeika, L. Bisikirskiene and Z. Strolia, “Mbse grid: A simplified sysml-based approach for modeling complex systems,” INCOSE International Symposium, vol. 27, null, 2017. doi: 10.1002/j.2334-5837.2017.00350.x. [Online]. Available: https://www.semanticscholar.org/paper/40dbcc2c56e9f5c339f988bb13aa606b9c504dc3.
49. A. Morkevicius, A. Aleksandraviciene, A. Armonas and G. Fanmuy, “Towards a common systems engineering methodology to cover a complete system development process,” INCOSE International Symposium, vol. 30, null, 2020. doi: 10.1002/j.2334-5837.2020.00713.x. Available: https://www.semanticscholar.org/paper/a655bd14ecbed8d68e75bb9cd1a92846501b46e5.