Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет Комп'ютерні технології та системи
Кафедра Комп'ютерні інформаційні технології

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «Дослідження методів валідації railML документів в середовищі .NET»

за освітньою програмою **Інженерія програмного забезпечення**
зі спеціальності: **121 Інженерія програмного забезпечення**

Виконав: студентка групи «ПЗ2321»

_____ /Маргарита ВИСКАРКА/

Керівник: _____ /доц. Олександр ІВАНОВ/

Нормоконтролер: _____ /доц. Світлана ВОЛКОВА/

Науковий консультант: _____ /Milan Wölke/

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань
Студентка _____

Дніпро – 2025 рік

Ministry of Education and Science of Ukraine

Ukrainian State University of Science and Technologies

Faculty Computer technologies and systems
Department Computer information technology

Explanatory Note

to Master's Thesis

on the topic: «Research of the validation methods of railML documents in the .NET environment»

according to educational curriculum **12 software engineering**
in the Speciality: **121 software engineering**

Done by the student of the group PZ2322: /Marharyta VYSKARKA/

Scientific Supervisor: /Oleksandr IVANOV/

Normative controller: /Svitlana VOLKOVA/

Research consultant: / Milan Wölke /

Dnipro – 2025

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Комп'ютерних технологій і систем
Кафедра: Комп’ютерні інформаційні технології
Рівень вищої освіти: магістр
Освітня програма: Інженерія програмного забезпечення
Спеціальність: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ____КІТ
_____ Вадим ГОРЯЧКІН
_____ грудня 2023 р.

ЗАВДАННЯ

На кваліфікаційну роботу ____Магістр____
студенту Вискарці Маргариті Юріївній____.
1. Тема дипломної роботи: «Дослідження методів валідації railML документів в середовищі .NET».
Керівник роботи: Іванов Олександр Петрович
затверджені наказом 1187 ст від 26.12.2023 року
2. Строк подання студентом роботи 10.01.2025 року
3. Вихідні дані до дипломної роботи:
_____.
4. Зміст пояснювальної записки (перелік питань до розробки):
4.1. Аналітична частина: Огляд предметної галузі;
4.2. Основна частина: Дослідження правильності роботі методів валідації та розробка рішення;
5. Перелік демонстраційного матеріалу:
5.1. презентація;
5.2. демонстраційне відео.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	01.07.24 – 01.08.24	
2	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	01.08.24 – 15.08.24	10%
3	Постановка задачі, технічне завдання	15.08.24 – 01.09.24	
4	Розробка інструментальних засобів дослідження	01.09.24 – 01.10.24	30%
5	Виконання досліджень	01.10.24 – 01.11.24	
6	Оформлення пояснювальної записки	01.11.24 – 31.12.24	60%
7	Розробка демонстраційних матеріалів	06.01.25 – 12.01.25	100%
8	Подання кваліфікаційної роботи до кафедри	13.01.25	
9	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	20.01.25 – 26.01.25	

Студент: _____ Маргарита ВИСКАРКА

Керівник роботи: _____ доц. Олександр ІВАНОВ

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра складається з 164 с., 65 рис. , 4 табл., 3 додатків, 27 джерел.

Об'єктом дослідження є валідація railML документів у середовищі .NET.

Мета роботи: дослідити доступні методи валідації railML документів у середовищі .NET. Провести аналіз особливостей їх валідації, оцінити їх переваги й недоліки та визначити або розробити найбільш відповідний вимогам railML.org метод.

Методика дослідження: теоретичний та практичний аналіз аналогів та методів валідації XML та railML, тестування за допомогою Unit-тестів та розробленого застосунку.

Перелік ключових слів: валідація, railML, XML, XSD, сформованість, розширення.

ЗМІСТ

ВСТУП

Залізничні операції містять безліч даних у різних категоріях, які часто окремо обробляються різними програмами. Використання кількох інтерфейсів для формату даних часто ускладнює обмін та може змушувати задіяні компанії мати значні витрати часу та ресурсів. Універсальний формат обміну даних, railML, має на меті полегшити процес шляхом надання лише одного інтерфейсу для кожної програми, який є легким, зручним та близьким до існуючих стандартів.

railML (Railway Markup Language) був створеним на основі XML, і, як і будь-який існуючий формат, він має дотримуватись певних правил, які визначають, яка саме інформація та у якому вигляді у ньому допускається. Для розробки та успішного використання railML має існувати можливість перевірки документів даного формату, як на сформованість, очікувану структуру, так і на зміст, тобто валідація. Хоч валідація railML теоретично можлива за допомогою стандартних валідаторів даних на основі XML, цей формат даних все ж таки має свої особливості, через що не надійно спиратись на їх перевірку шляхом сторонніх валідаторів, функціонування яких може постійно змінюватись незалежно від розвитку railML. Через це виникає потреба у власному валідаторі для цього формату обміну даних.

Для вирішення такої задачі у 2015 році був створений настільний застосунок railVIVID 1, розроблений мовою Java на основі валідатора Xerces. Однак шляхом його аналізу в 2024 році було визначено, що продовжувати роботу над ним зараз є недоцільним через застарілість використаних технологій та складність розробленого програмного забезпечення. Через що у 2024 році була виявлена потреба в розробці валідатора для railML у середовищі .NET, яке наразі все ще є постійно оновлюваним та може підтримувати, як веб, так і настільні застосунки, що було важливим через попит на подвійну версію для нової імплементації.

Так як попередній розвиток railVIVID 1 був виконаний лише в середовищі Java, стає необхідним дослідження та перевірка методів валідації railML документів саме в середовищі .NET. Пошук рішення цієї задачі має вирішальне

значення для розробки надійного та правильного валідатора саме для цього формату обміну даних в необхідному середовищі.

РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Валідація XML файлів

Валідація XML файлів є процесом перевірки документу на сформованість («well-formedness») та відповідність визначеній структурі. Сформованість визначається базовими синтаксичними правилами XML. Коректність файлу також залежить від відповідності документу визначеним для нього правилам, наприклад, DTD (Document Type Definition) або схемі (XSD) [1]. В межах цього дослідження буде розглядатись лише перевірка коректності файлів за допомогою XSD, так як вони використовуються при валідації railML документів.

1.1.1 Перевірка на сформованість (Well-formedness)

Сформованість документу, як зазначено вище, визначає правильність файлу XML відповідно до низки синтаксичних правил, які регулюють XML. Ці правила також поширюються і на документи railML, так як цей формат даних був створений на основі XML [2]. Ці правила включають визначення таких помилок, як пропущена кутова дужка або закриваючий тег, декілька однаково названих атрибутів або вихід некореневого елементу за межі батьківського елементу.

1.1.2 Валідація відповідно до XML схем (XSD)

Схема – це файл із правилами, які визначають, що може та не може містити файл даних XML [3]. Такі файли зазвичай мають розширення .xsd. Валідація за допомогою схеми або схем є процесом перевірки чи є документ правильний відповідно до визначних правил у файлах схем. Схеми у файлах XML зазвичай визначаються у «xsi:schemaLocation», як пара простір назви та місцезнаходження схеми, яке може бути локальне або віртуальне. Однак визначення схем також може бути здійснено і за допомогою їх включення або імпортування шляхом «xs:include» або «xs:import» [4].

1.2 railML: особливості формату та валідації

railML (Railway Markup Language) — це поширений формат обміну, який використовує систематику XML для опису даних, що стосуються залізниці. railML забезпечує обмін залізничними даними між внутрішніми та зовнішніми

залізничними застосунками. railML розроблено в рамках так званого «консорціуму railML» від railML.org.

railML базується на XML, а під-області використовують інші існуючі XML-схеми, такі як MathML і GML. railML складається з підсхем. У версії 2.4 продуктивно використовуються три підсхеми:

- Графік руху поїздів для моделювання графіку руху поїздів.
- Інфраструктура для (в основному топологічного) моделювання колій і сигнального обладнання.
- Рухомий склад для моделювання транспортних засобів.

Починаючи з railML версії 3.1, на вимогу спільноти було введено додаткову підсхему:

- Блокування для моделювання маршрутів руху поїздів [2].

Основною особливістю, яку необхідно врахувати в railML для задачі валідації, є присутність можливості розширення railML за допомогою сторонніх схем. Як і з XML, є велика кількість способів розширення railML. Вони існують на той випадок, якщо користувачу для особистих потреб не вистачає елементів, визначених у railML, або якщо необхідно доповнити вже існуючі.

“У railML 2 та railML 3.1 запропонованими засобами розширення railML для користувацьких потреб були <xs:any>-елемент для визначення нових елементів, <xs:anyAttribute>-елемент для визначення нових атрибутів і загальний тип tOtherEnumerationValue для визначення нових значень переліку у визначених місцях.

У railML 3.2 і вище необхідним рішенням є використання атрибута xsi:type для введення розширень у випущене ядро railML. Оскільки екземпляри <xs:any> можуть порушити процес перевірки, усі випадки було видалено під час переходу з railML 3.1 на railML 3.2. Розширення через xs:anyAttribute і tOtherEnumerationValue все ще можливі, якщо вони існують.”[5].

Особливість для валідації у випадку розширення є у тому, що валідація тоді має відбуватися з фокусом саме на схеми railML, навіть при перевірці на відповідність документу користувацьким схемам. Тобто будь-які проблеми, що

стосуються власних схем користувача та які не порушують схеми railML, мають розглядатись як попередження, а от проблеми відносно до схем railML, мають розглядатись як помилки або попередження в залежності від типу проблеми.

1.3 Аналоги програмного забезпечення для валідації XML та railML

На даний момент існує безліч різних валідаторів для XML, як для перевірки сформованості документів, так і для валідації відповідно до зазначених схем. Їх функціонування та результати можуть мати відмінності, розглянемо декілька доступних прикладів таких застосунків.

1.3.1 Інструмент валідації railML файлів – railVIVID 1

railVIVID 1 — це “вдосконалений інструмент для перегляду та перевірки файлів railML. Універсальність railVIVID дозволяє користувачам візуалізувати та перевіряти низку залізничних даних, включаючи дані про розклад, інфраструктуру, рухомий склад” [6].

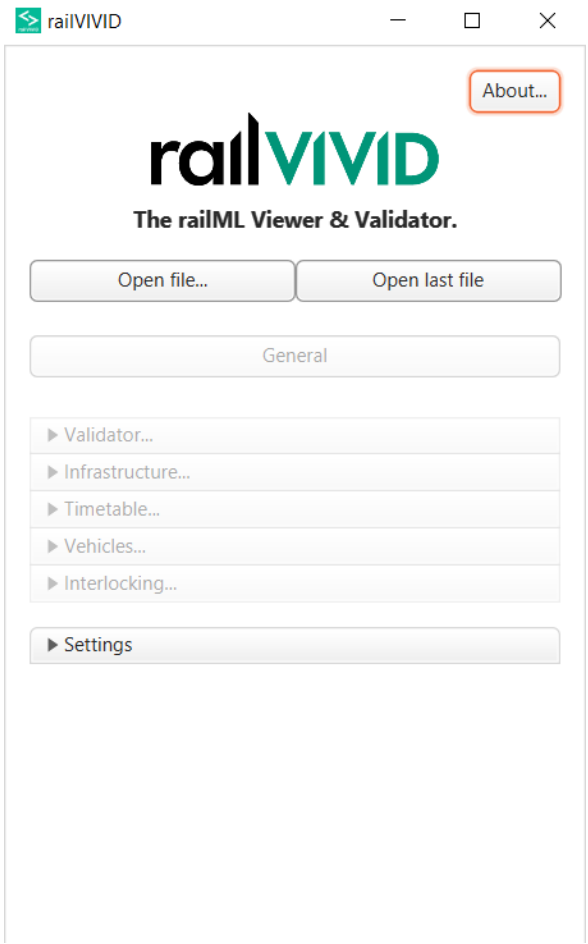


Рисунок 1.1 – Вигляд застосунку railVIVID 1 (зображення взято з railVIVID v.1.3.1)

«Остання офіційна версія railVIVID V1.3.1 була опублікована на 45-й конференції railML.org 6 червня 2024 року. Останні вдосконалення у версії 1.3 містять семантичні перевірки для різних обмежень railML 2.5, таких як періоди роботи, покажчики та дані про обіг. Крім того, оновлення включає виправлення помилок, які були перевірені спільнотою railML і може використовуватися всіма користувачами.» [6].

У лютому та березні 2024 року проводився аналіз даного програмного забезпечення, на основі чого було визначено, що по ряду причин, таких як складність коду, застарілість технологій і т.д. необхідне створення оновленого застосунку з самого початку. Через це під час 45-ї railML Конференції [7] було анонсовано заплановану розробку нової імплементації – railVIVID 2 для середовища .NET, робота над якою почалась у серпні 2024 року. Велика частина розробки офіційного railVIVID 2 включала результати створення інструментальних засобів для даної дипломної роботи.

1.3.2 Стороннє програмне забезпечення для валідації XML файлів

LiquidStudio XML Editor від Liquid Technologies

«Редактор XML Editor на основі Rich Text забезпечує підсвічування синтаксису, перевірку на сформованість, перевірку з урахуванням схеми, автозаповнення, розпізнавання схеми, перехід до визначення XSD, перевірку орфографії та багато іншого» [8].

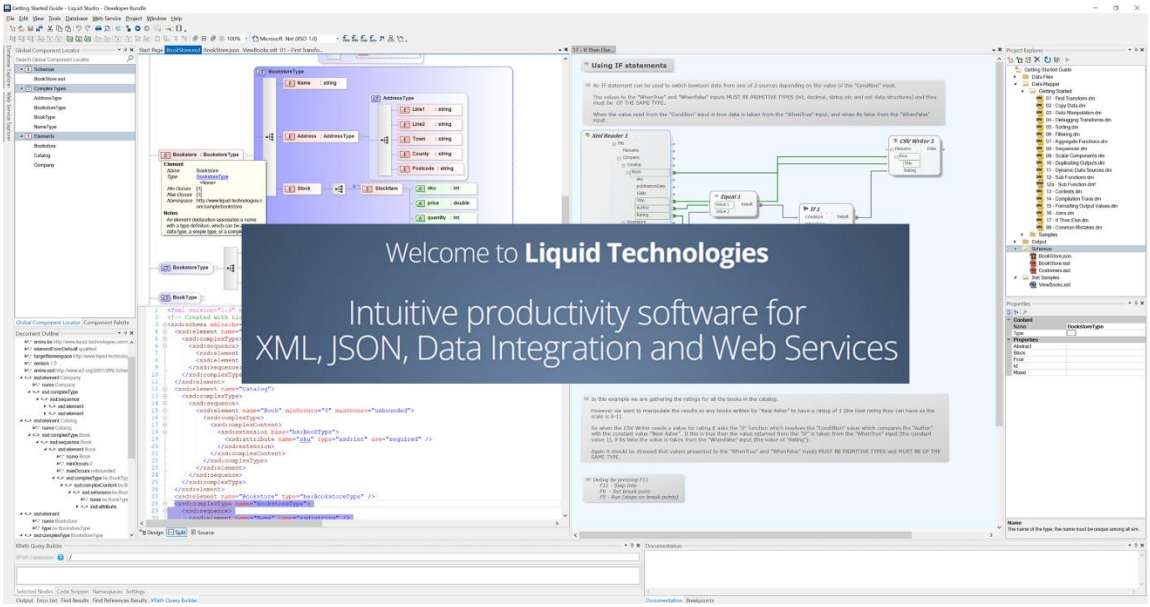


Рисунок 1.2 – Початок роботи з Liquid Studio (Зображення взято з сайту Liquid Technologies)

З сайту Liquid Technologies про роботу XML Editor: «Редактор XML перевіряє, чи ваш XML-документ правильно сформований і дійсний за пов'язаною схемою XML. Підтримуються такі формати XML-схеми:

– DTD - Microsoft .Net XML Validator & Xerces.
– XSD 1.0 – Microsoft .Net XML Validator & Xerces.
– XSD 1.1 – Microsoft .Net XML Validator.
– RelaxNG – Моно.
– Schematron – працює через Saxon XSLT Engine.

Помилки повідомляються в режимі реального часу під час введення та відображаються всередині документа XML і у вікні помилок.» [8].

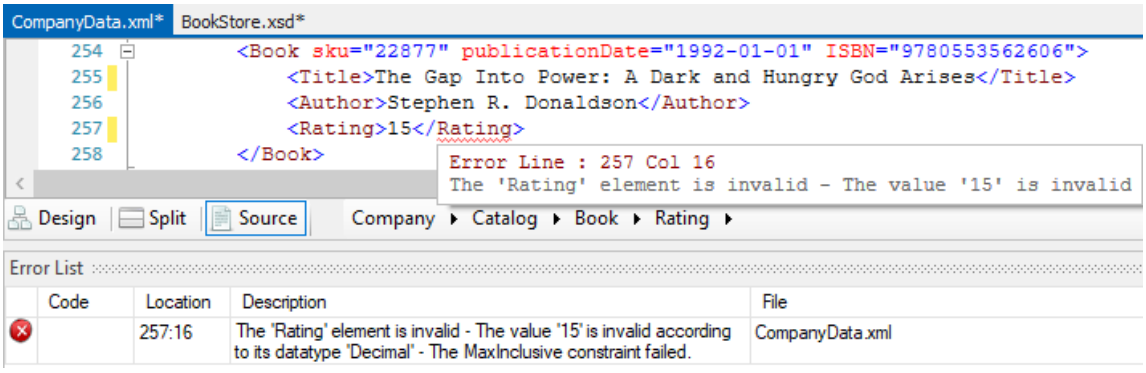


Рисунок 1.3 – Валідація документу за допомогою XML Editor (Зображення взято з сайту Liquid Technologies)

XMLSpy SmartFix XML Validator від Altova

«Розробникам потрібен редактор JSON і XML, який додає цінність, крім перевірки закритості дужок і базової перевірки валідації. XMLSpy надає повний набір функцій … і включає графічні представлення, генератори коду, майстри та інші інтелектуальні функції редагування JSON і XML, які допомагають вам виконувати роботу швидше, ніж будь-коли.» (див. Рис. 1.4 для прикладу роботи даного редактору) [9].

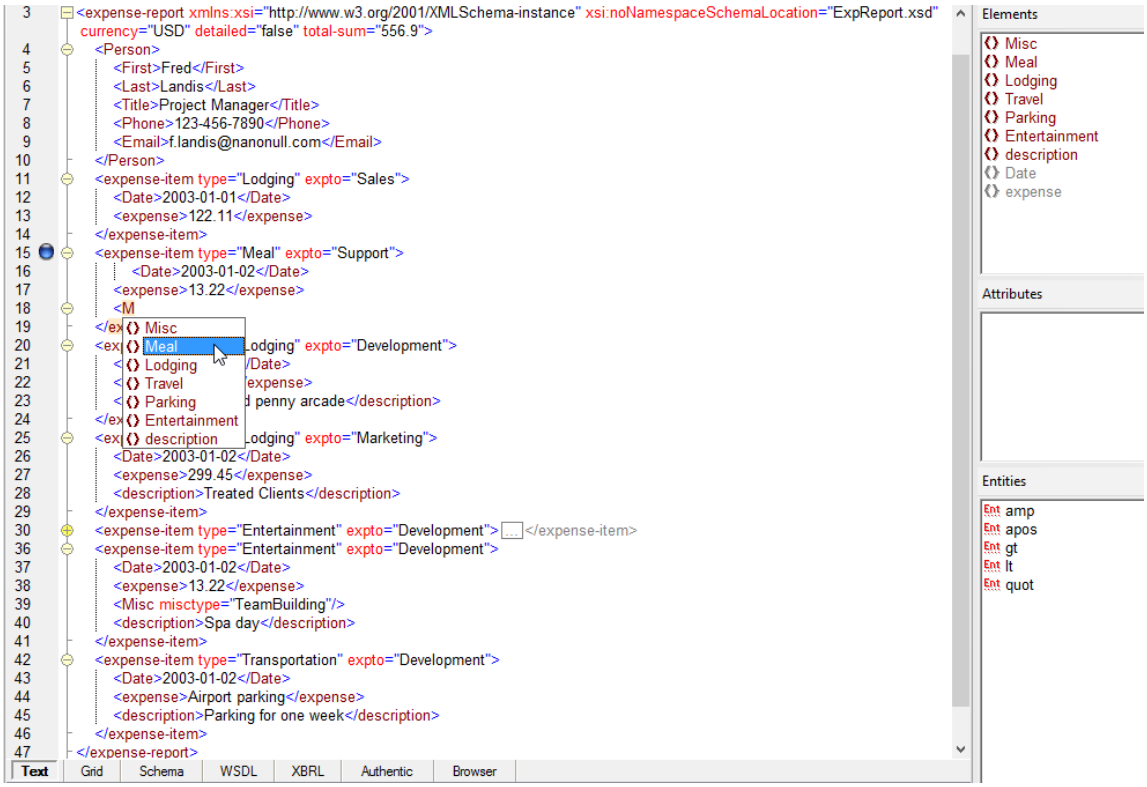


Рисунок 1.4 – Приклад роботи з XMLSpy (Зображення взято з Altova)

«XMLSpy працює на базі механізму перевірки Altova RaptorXML, який був створений з нуля, щоб забезпечити якомога точнішу відповідність стандартам у поєднанні з надвисокою швидкістю перевірки.» [10]. Тобто на відміну від Liquid Studio цей редактор використовує не вже існуючі валідатори з таких середовищ, як .NET та Java, а має свій власний механізм.

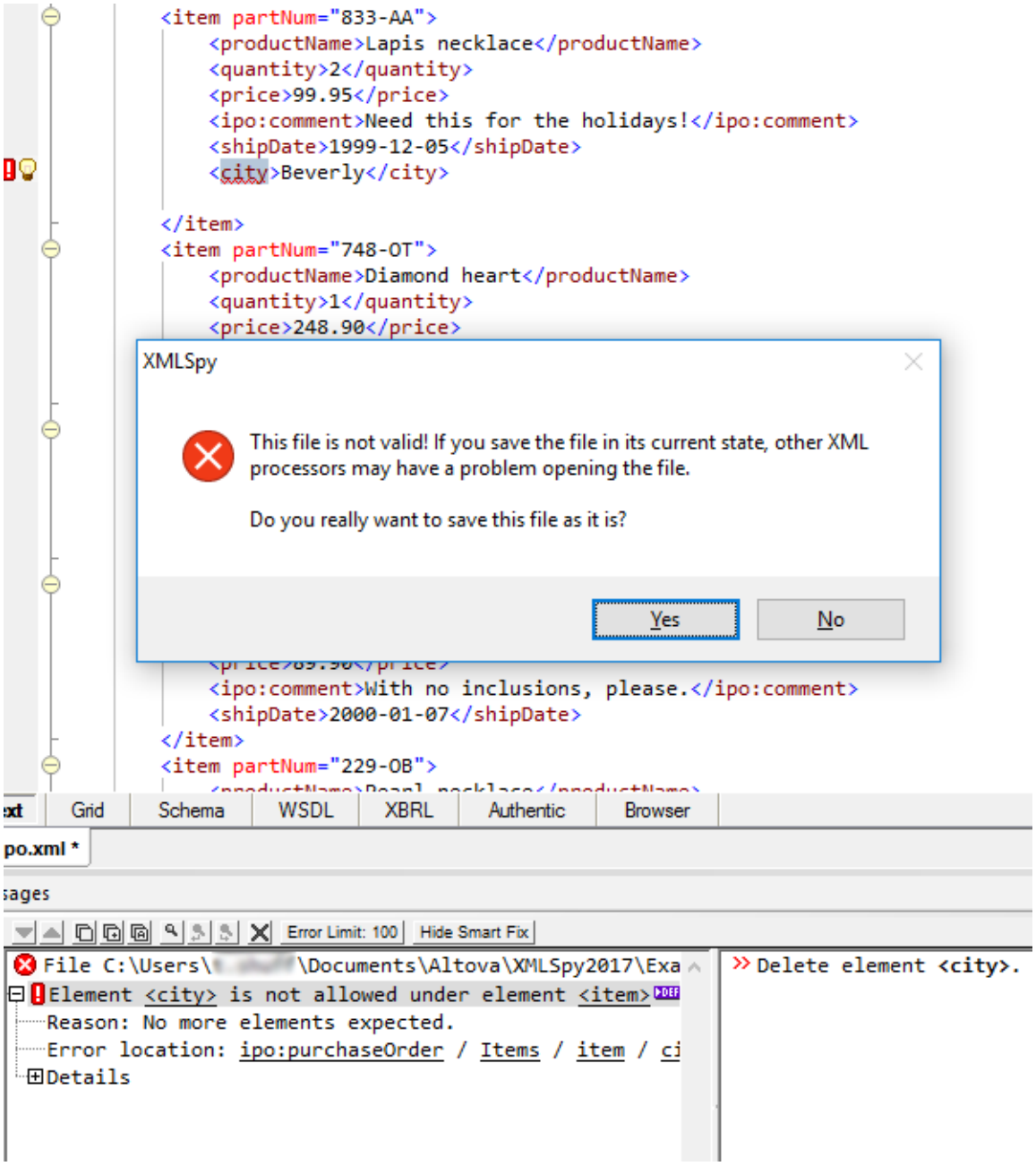


Рисунок 1.5 – Приклад валідації документу за допомогою XMLSpy (Зображення взято з сайту Altova)

«Окрім SmartFix, програмне забезпечення містить численні функції, які допомагають створювати правильні XML-документи та швидко й легко вирішувати проблеми. Вікно перевірочних повідомлень із кількома вкладками пропонує детальну інформацію про будь-які помилки у вашому документі чи XML-проекті, а зручна панель інструментів дозволяє легко переміщатися, шукати та копіювати інформацію про помилки за потреби. Після підтвердження

та перевірки правильності XMLSpy надає докладні відомості про всі помилки, зокрема:

– Причину помилки.

– Посилання на помилку в робочому XML-файлі.

– Посилання на відповідні визначення(я) у пов’язаному файлі схеми.

– Посилання на відповідну інформацію у відповідній специфікації W3C» (див. наведений вище Рис. 1.5 для прикладу валідації файлу з XMLSpy) [10].

1.4 Постановка задачі

Існує велика кількість різних застосунків (як веб, так і настільних) для валідації XML та за асоціацією railML, створених у різних середовищах, однак багато з них навіть для простих файлів можуть надавати різні результати в залежності від особливостей їх функціонування. Через це потрібне визначення вимог до правильної валідації railML на основі аналізу особливостей цього формату даних та попередньо створеного валідатора саме для цього формату. Так як railVIVID 1 є застарілим, та використовує ресурси середовища Javи, для вирішення поставленої задачі валідації railML документів в .NET необхідно на основі визначених вимог дослідити доступні методи валідації в цьому середовищі для знаходження найкращого рішення.

Висновки до розділу 1

Отже у цьому розділі будо виконано ознайомлення з поняттям валідації XML документів та проаналізовано схожість і відмінності валідації основаного на XML форматі обміну даних – railML. В процесі огляду предметної області було розглянуто декілька аналогів програмного забезпечення для валідації, такі як railVIVID 1 від railML.org, XML Editor від LiquidStudios та XMLSpy від Altova. Було зроблено висновок, що основною відмінністю обох прикладів стороннього програмного забезпечення від офіційного валідатора railVIVID 1 є надання можливості редагування вмісту документу, в той час як застосунок від railML.org надає можливість лише перевіряти правильність файлу та переглядати його візуалізований вміст, що і очікується від розроблюваного програмного забезпечення. Також в процесі дослідження було виявлено потребу в розробці нової імплементації railVIVID 2 в середовищі .NET. В результаті даного аналізу було визначено необхідність дослідження методів валідації XML (та за схожістю базової структури railML) в цьому середовищі для виявлення їх особливостей та відповідності вимогам створеними railML.org для виявлення найкращого способу валідації.

РОЗДІЛ 2 ОБҐРУНТУВАННЯ ЕКСПЕРИМЕНТАЛЬНОГО МЕТОДУ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ МЕТОДІВ ВАЛІДАЦІЇ

Аналіз доступних бібліотек в середовищі .NET є важливим для перевірки відповідності їх валідації вимогам railML.org, для чого має відбуватися ретельний пошук методів для вирішення поставленої задачі.

2.1 Загальна методика проведення наукових досліджень

Для визначення підходящих бібліотек у .NET для валідації railML файлів відповідно до вимог необхідно провести їх попереднє тестування через те, що поверхневого аналізу методів буде недостатньо, так як опис методів не надає інформації про їх поведінку у специфічних для railML випадках. Головна увага при тестуванні має звертатись на можливість валідації файлів з наданням переваги валідації за допомогою схем railML, а не користувацьких розширень, дозволених у деяких випадках.

Також для попереднього тестування буде розроблено декілька railML файлів з ситуаціями, розпізнання яких є критичним для вирішення задачі валідації:

Таблиця 2.1 – Критичні ситуації для валідації файлів з розширеннями

№	Ситуація	Версія railML	Схема розширень	Дозвіл за схемою railML	Класифікація
1	Файл має розширення у вигляді елементів	2	Відсутня/у ній відсутнє визначення елементів	Розширення дозволене за схемою railML (<xs:any>)	Попередження
2	Файл має розширення у вигляді елементів	2	Відсутня/у ній відсутнє визначення елементів	Розширення не дозволене за схемою railML (лише визначені елементи)	Помилка
3	Файл має розширення у	2	Відсутня/у ній відсутнє	Розширення не дозволене за	Помилка

	вигляді елементів		визначення елементів	схемою railML (ніяких визначених елементів)	
4	Файл має розширення у вигляді атрибутів	2	Відсутня/у ній відсутнє визначення елементів	Розширення дозволене за схемою railML (<xs:anyAttribute>, <xs:any>)	Попередження
5	Файл має розширення у вигляді атрибутів	2	Відсутня/у ній відсутнє визначення елементів	Розширення не дозволене за схемою railML (лише визначені елементи)	Помилка
6	Файл має розширення у вигляді атрибутів	2	Відсутня/у ній відсутнє визначення елементів	Розширення не дозволене за схемою railML (ніяких визначених елементів)	Помилка
7	Файл має розширення	3	Відсутня/у ній відсутнє визначення елементів	Розширення дозволене за схемою railML (<xsi:type>) (лише де немає any)	Помилка

Приклади файлів для кожної ситуації (приклади несуть лише синтаксичний характер, доречність розширень та їх зміст не береться до уваги):

1) Дозволений під-елемент «something» у елементі «category» (має <xs:any> у своїй схемі [11]):

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<railml xmlns="http://www.railml.org/schemas/2011"
        xmlns:my="https://www.my-company.com/my-department/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.railml.org/schemas/2011
http://www.railml.org/schemas/2011/railML-2.1/railML.xsd"
        version="2.1">
  <timetable id="tt1">
    <categories>
      <category id="c1" code="IC" name="InterCity"
description="Fast large city connection" trainUsage="passenger">
        <my:something id="s1"/>
      </category>
      <category id="c2" code="RE" name="RegionalExpress"
description="Fast small city connection" trainUsage="passenger"/>
    </categories>
  </timetable>
</railml>
```

2) Не дозволений під-елемент «something» у елементі «categories» (визначає лише «category» як можливий під-елемент [12]):

```
<?xml version="1.0" encoding="UTF-8"?>
<railml xmlns="http://www.railml.org/schemas/2011"
        xmlns:my="https://www.my-company.com/my-department/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.railml.org/schemas/2011
http://www.railml.org/schemas/2011/railML-2.1/railML.xsd"
        version="2.1">
  <timetable id="tt1">
    <categories>
      <category id="c1" code="IC" name="InterCity"
description="Fast large city connection" trainUsage="passenger"/>
      <category id="c2" code="RE" name="RegionalExpress"
description="Fast small city connection" trainUsage="passenger"/>
      <my:something id="s1"/>
    </categories>
  </timetable>
```

```
</railml>
```

3) Не дозволений елемент «something» у елементі «additionalName» (не визначає жодних під-елементів [13]):

```
<?xml version="1.0" encoding="UTF-8"?>
<railml xmlns="http://www.railml.org/schemas/2011"
        xmlns:my="https://www.my-company.com/my-department/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.railml.org/schemas/2011
http://www.railml.org/schemas/2011/railML-2.1/railML.xsd"
        version="2.1">
    <timetable id="tt1">
      <categories>
        <category     id="c1"     code="IC"     name="InterCity"
trainUsage="passenger">
          <additionalName name="IC1" description="Intercity train
with fast large city connection">
            <my:something id="s1"/>
          </additionalName>
        </category>
        <category   id="c2"   code="RE"   name="RegionalExpress"
description="Fast small city connection" trainUsage="passenger"/>
      </categories>
    </timetable>
</railml>
```

4) Дозволений атрибут «maximumWait» в елементі «times» та атрибут «length» в елементів «trainPart» (перший елемент має <xs:anyAttribute> у своїй схемі, другий має <xs:any> [14] [15])

```
<?xml version="1.0" encoding="UTF-8"?>
<railml xmlns="http://www.railml.org/schemas/2013"
        xmlns:my="https://www.my-company.com/my-department/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.railml.org/schemas/2013
http://www.railml.org/schemas/2013/railML-2.2/railML.xsd"
        version="2.2">
```

```
    <infrastructure id="in1">
      <operationControlPoints>
        <ocp id='ocp_A'/>
        <ocp id='ocp_B'/>
      </operationControlPoints>
    </infrastructure>
    <timetable id="tt1">
      <trainParts>
        <trainPart id="tp_1" my:length="60">
          <ocpsTT>
            <ocpTT ocpRef='ocp_A' ocpType='stop'>
              <times scope='earliest' departure='16:30:11'/>
              <times    scope='scheduled'    departure='16:31:18'
my:maximumWait="22"/>
              <times scope='published' departure='16:30:11'/>
              <times scope='actual' departure='16:39:10'/>
            </ocpTT>
            <ocpTT  ocpRef='ocp_B' ocpType='pass'>
              <times   scope='scheduled'   departure='16:38:02.46'
my:maximumWait="12"/>
              <times scope='actual' departure='16:45:27'/>
            </ocpTT>
          </ocpsTT>
        </trainPart>
      </trainParts>
      <trains>
        <train id="trc_1" type="commercial" trainNumber="0111">
          <trainPartSequence sequence="1">
            <trainPartRef ref="tp_1" position="1"/>
          </trainPartSequence>
        </train>
      </trains>
    </timetable>
  </railml>
```

5) Не дозволений атрибут «description» в елементі «trackBegin» (елемент не має визначає <xs:anyAttribute> або <xs:any> у своїй схемі [16]):

```
<?xml version="1.0" encoding="UTF-8"?>
<railml xmlns="http://www.railml.org/schemas/2011"
        xmlns:my="https://www.my-company.com/my-department/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.railml.org/schemas/2011
http://www.railml.org/schemas/2011/railML-2.1/railML.xsd"
        version="2.1">
  <infrastructure id="in1">
    <tracks>
     <track id='tr_1' name='Track1' type='mainTrack'>
       <trackTopology>
         <trackBegin id='trn_begin_1' pos='23450' absPos='3800'
my:description="Information about track's begining">
         <openEnd id="oEb26301"/>
         </trackBegin>
         <trackEnd id='trn_end_1' pos='24070' absPos='4420'>
         <openEnd id="oEb26302"/>
         </trackEnd>
         <crossSections/>
       </trackTopology>
     </track>
    </tracks>
    <trackGroups>
      <line id='ln_1' name='6228'>
        <trackRef ref='tr_1'/>
      </line>
    </trackGroups>
    <operationControlPoints>
      <ocp id='ocp_A'/>
    </operationControlPoints>
  </infrastructure>
</railml>
```

6) Не дозволений атрибут «my:colour» в елементів «categories» (елемент не визначає ніяких атрибутів [12]).

```
<?xml version="1.0" encoding="UTF-8"?>
<railml xmlns="http://www.railml.org/schemas/2011"
        xmlns:my="https://www.my-company.com/my-department/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.railml.org/schemas/2011
http://www.railml.org/schemas/2011/railML-2.1/railML.xsd"
        version="2.1">
  <timetable id="tt1">
    <categories  my:colour="red">
      <category     id="c1"     code="IC"     name="InterCity"
description="Fast large city connection" trainUsage="passenger"/>
      <category   id="c2"   code="RE"   name="RegionalExpress"
description="Fast small city connection" trainUsage="passenger"/>
    </categories>
  </timetable>
</railml>
```

7) Дозволене розширення за допомогою «ExtendedOperationalPoint» та «ExtendedTrack» елементів «operationalPoint» та «track», однак через відсутність схеми створюється помилка (тип має обов’язково бути визначеним [17] [18]).

```
<?xml version="1.0" encoding="UTF-8"?>
<railML
   xmlns="https://www.railml.org/schemas/3.2"
   xmlns:example="https://www.somedomain.net/2022/exampleExtensi
on"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation=""
   version="0.1">

   <infrastructure id="is01">
        <functionalInfrastructure>
             <operationalPoints>
```

```
                    <operationalPoint
xsi:type="example:ExtendedOperationalPoint" id="op01">
                        <!-- You can use all elements and
attributes of the original operational point in addition to the ones
we added -->
                        <name name="Praha" language="cz"/>
                        <name name="Prag" language="de"/>
                        <name name="Prague" language="en"/>
                        <example:priceCategory
example:categoryNumber="2" example:priceMultiplier="2.1">
                            <example:name name="Tourist fare"
language="en"/>
                        </example:priceCategory>
                    </operationalPoint>
                    <example:name      name="Tourist      fare"
language="en"/>
                </operationalPoints>

                <!-- The same applies to the extended tracks -->
                <tracks>
                    <track      xsi:type="example:ExtendedTrack"
id="tr01" type="mainTrack"
                        example:additionalExampleComment="This
attribute was added just to show how to extend a type with an
additional attribute.">
                        <!-- You can use all elements and
attributes of the original track in addition to the ones we added -
->
                        <designator          register="_Tracks"
entry="1234"/>
                        <length value="500" type="physical"/>
                        <example:priceCategory
example:categoryNumber="1" example:priceMultiplier="1.0">
                            <example:name name="Normal fare"
language="en"/>
```

```
                        </example:priceCategory>
                        <example:priceCategory
example:categoryNumber="3" example:priceMultiplier="1.5">
                              <example:name name="Express fare"
language="en"/>
                        </example:priceCategory>
                  </track>
                  <track id="tr02" type="sidingTrack">
                        <!-- You can also use the original type-
->
                        <designator          register="_Tracks"
entry="4321"/>
                        <length value="1200" type="physical"/>
                  </track>
            </tracks>

      </functionalInfrastructure>
   </infrastructure>
</railML>
```

Всі описані ситуації підібрані з фокусом на відсутність схем розширень при валідації, тому що в такому випадку пріоритет має надаватись відповідності файлу схемі railML, а будь-які проблеми з розширеннями мають вважатись лише попередженнями для користувача, при умові що вони не є порушеннями правил railML. Також доповнюючи цей факт, будь-які проблеми, пов’язані з відсутністю схеми, мають також вважатись попередженнями (якщо вони не порушують правил railML або XML, як з <xsi:type> у ситуації №7), так як при валідації завжди використовуються доступні схеми railML.

Усього в результаті поверхневого аналізу доступних безкоштовних бібліотек у .NET було виявлено 3 основні стандартні методи для валідації XML файлів за допомогою XSD, які можливо буде використати для тестування:

1) XDocument.Validate(),
2) XMLDocument.Validate(),
3) XMLReader.Create().

Усі інші розглянуті бібліотеки або використовували один з вище перерахованих методів у своєму коді, або їх імплементація була неуспішною, як, наприклад, IKVM [19]. Відповіді на публікацію питання на платформі StackOverflow [20] лише підтвердили результати: запропоновані рішення або вже були відкинуті, або були на комерційній основі.

2.2 Результати аналізу та виявлена проблема стандартних методів .NET

Після визначення описаних вище ситуацій було проведено попереднє тестування обраних трьох стандартних функцій (більш детальний розгляд буде проведено у Розділі 4). У підсумках цього тестування було виявлено, що будь-які проблеми, пов’язані з користувацькими схемами, які використовувались для розширення railML, розглядалися в процесі валідації як помилки. Це вимагало власноручної обробки цих проблем як попереджень у випадку недоступності визначень для їх елементів/атрибутів, так як це не порушує правил схеми railML.

У ситуаціях 1, 2, 3, 4 та 7 результат стандартних функцій загалом збігається з очікуваним при врахуванні власної обробки ситуацій не оголошених елементів/атрибутів як попереджень. Тобто у ситуаціях 1 та 4 при помилці «елемент/атрибут не оголошений» через відсутність схеми, в якій він мав бути визначений , виконується власна обробка помилки як попередження, в той час як повідомлення для 2, 3 та 7 є більш детальними з інформацією про недозволенність елементу або неправильний тип («element is not allowed»/«invalid child», «invalid xsi:type» і т.д.).

Особливий результат був помічений у ситуаціях 5 та 6: не звертаючи увагу на порушення правил схеми railML (відсутності <xs:any>/<xs:anyAttribute> в елементі «trackBegin» або відсутності визначення будь-яких атрибутів для елементу «categories»), при валідації з’являється єдина помилка з текстом «атрибут не оголошений» («attribute is not declared»). Додатково було проведено перевірку майже ідентичних файлів для даних ситуації з єдиною зміною: доступністю XSD схеми розширення та наявністю визначення атрибуту у даній схемі:

Таблиця 2.2 – Приклади для ситуацій №5 та 6 зі схемою розширень

Ситуація	Файли	Схема
5	<?xml version="1.0" encoding="UTF-8"?> <railml xmlns="http://www.railml.org/schemas/2011" xmlns:my="https://www.my-company.com/my-department/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.railml.org/schemas/2011 http://www.railml.org/schemas/2011/railML-2.1/railML.xsd **https://www.my-company.com/my-department/ SimpleExtensionSchema.xsd**" version="2.1"> <infrastructure id="in1"> <tracks> <track id='tr_1' name='Track1' type='mainTrack'> <trackTopology> <trackBegin id='trn_begin_1' pos='23450' absPos='3800' **my:color="yellow"**> <openEnd id="oEb26301"/>	<?xml version="1.0" encoding="UTF-8"?> <xs:schema targetNamespace="https://www.my-company.com/my-department/" xmlns="https://www.my-company.com/my-department/" xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0"> <xs:attribute name="colour" type="colourType"> <xs:annotation> <xs:documentation>Provide the colour of the model</xs:documentation> </xs:annotation> </xs:attribute> <xs:simpleType name="colourType">

	</trackBegin> <trackEnd id='trn_end_1' pos='24070' absPos='4420'> <openEnd id="oEb26302"/> </trackEnd> <crossSections/> </trackTopology> </track> </tracks> <trackGroups> <line id='ln_1' name='6228'> <trackRef ref='tr_1'/> </line> </trackGroups> <operationControlPoints> <ocp id='ocp_A'/> </operationControlPoints> </infrastructure> </railml>	<xs:restriction base="xs:string"> <xs:enumeration value="blue"/> <xs:enumeration value="red"/> <xs:enumeration value="yellow"/> </xs:restriction> </xs:simpleType> </xs:schema>
6	<?xml version="1.0" encoding="UTF-8"?> <railml xmlns="http://www.railml.org/schemas/2011" xmlns:my="https://www.my-company.com/my-department/"	

	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.railml.org/schemas/2011 http://www.railml.org/schemas/2011/railML-2.1/railML.xsd **https://www.my-company.com/my-department/ SimpleExtensionSchema.xsd**" version="2.1"> <timetable id="tt1"> <categories **my:colour="red"**> <category id="c1" code="IC" name="InterCity" description="Fast large city connection" trainUsage="passenger"/> <category id="c2" code="RE" name="RegionalExpress" description="Fast small city connection" trainUsage="passenger"/> </categories> </timetable> </railml>	

Однак при тестуванні повідомлення не змінилось, з чого можна зрозуміти, що воно не спричинено відсутністю схеми розширень або визначення елементу у ній (атрибут «colour» є присутнім у наведеній вище схемі). Тобто такий текст повідомлення є реакцію на порушення схеми railML.

З цього можна визначити, що існує три різні сценарії, в яких валідація надає помилку з однаковим текстом «елемент/атрибут не оголошений»:

– відсутність або недоступність схеми для елемента/атрибута;

– відсутність визначення атрибута/елемента у наданій схемі;

– присутність не очікуваного атрибуту в елементі, який не визначає <xs:any> або <xs:anyAttribute> в своїй схемі.

Перші два сценарії мають оброблюватись як попередження, так як вони не порушують схеми railML і стосуються лише розширень, тому що валідація відбувається завжди за доступною офіційною схемою і елементи схеми railML тому є визначеними. Серйозна проблема виникає при поєднанні перший двох сценарій з третім, тобто коли атрибут одночасно є невизначеним і недозволеним за схемою, що і є ситуаціями № 5 та 6. В такому випадку неможливо визначити, чи повідомлення викликано, наприклад, відсутністю схеми розширення чи тому що атрибут існує в елементі railML, який не визначає жодних атрибутів, і тому не є зрозумілим, як саме така ситуація має оброблюватись – як жорстка помилка чи лише попередження. Таким чином було визначено проблему повідомлень при валідації стандартними функціями .NET, ситуації №5 та 6 можна визначити як проблематичні при майбутньому аналізі.

2.3 Тестування railVIVID 1 та стороннього програмного забезпечення

Для порівняння результатів було розглянуто вже доступне програмне забезпечення, щоб проаналізувати реакцію різних валідаторів на проблематичні ситуації.

Спершу можна перевірити попередника розроблюваного застосунку, тобто railVIVID 1 (версія 1.3.1), валідація якого відбувається на основі результатів Xerces середовща Java:

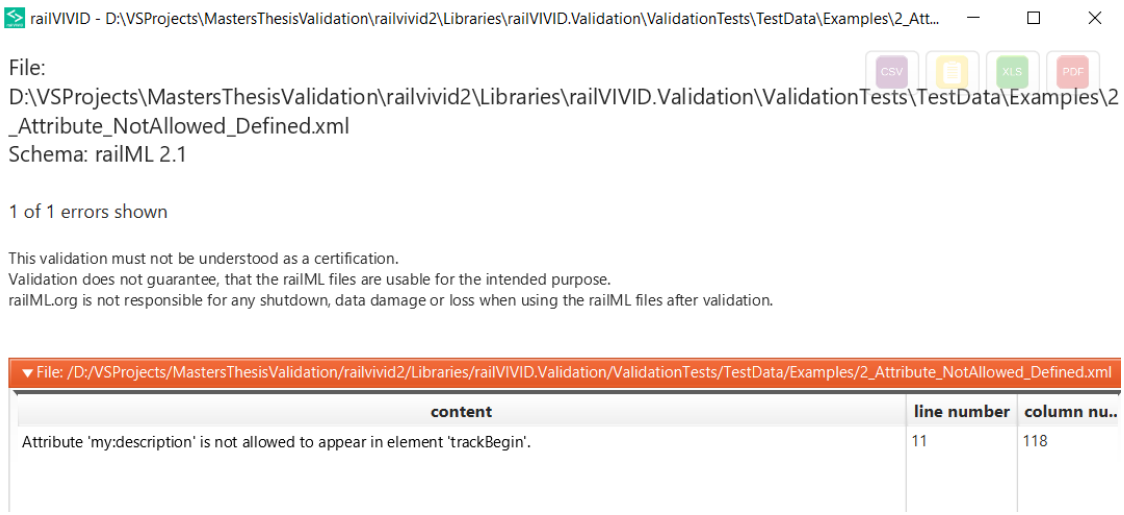


Рисунок 2.1 – Результати валідації railVIVID 1 файлу для ситуації №5 (не дозволений атрибут в елементі, який визначає деякі атрибути в своїй схемі)

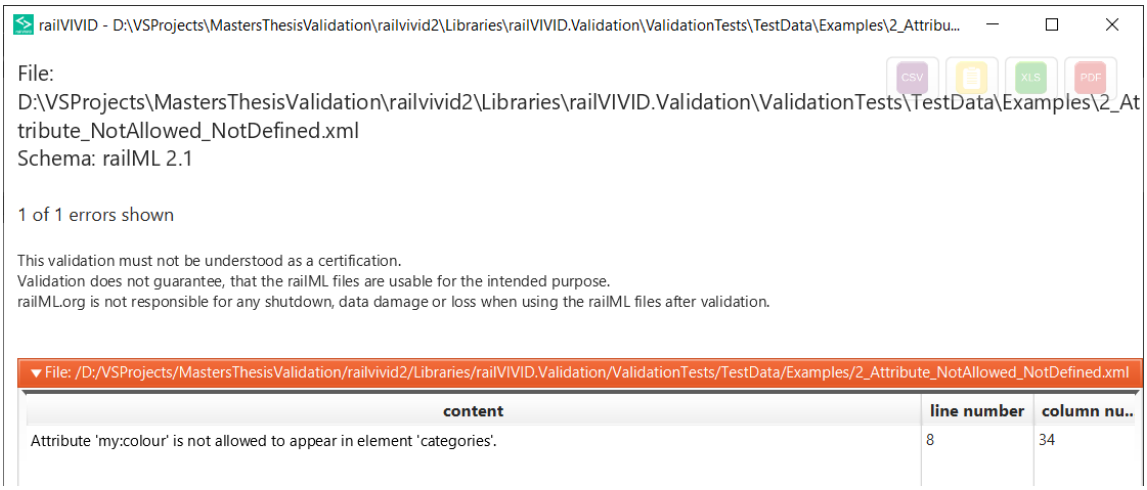


Рисунок 2.2 – Результати валідації railVIVID 1 файлу для ситуації №6 (не дозволений атрибут в елементі, який не визначає атрибути в своїй схемі)

Як можна побачити на наведений рисунках, railVIVID 1 розпізнає обидві ситуації як помилки і надає інформацію про недозволенність розглянутих атрибутів, тобто такий результат має бути очікуваним і у випадку нового валідатора.

Також було протестоване описане у першому розділі стороннє програмне забезпечення, наприклад, LiqiudStudio також надає можливість порівняти результати валідації за допомогою бібліотек Microsoft .NET та Xerces:

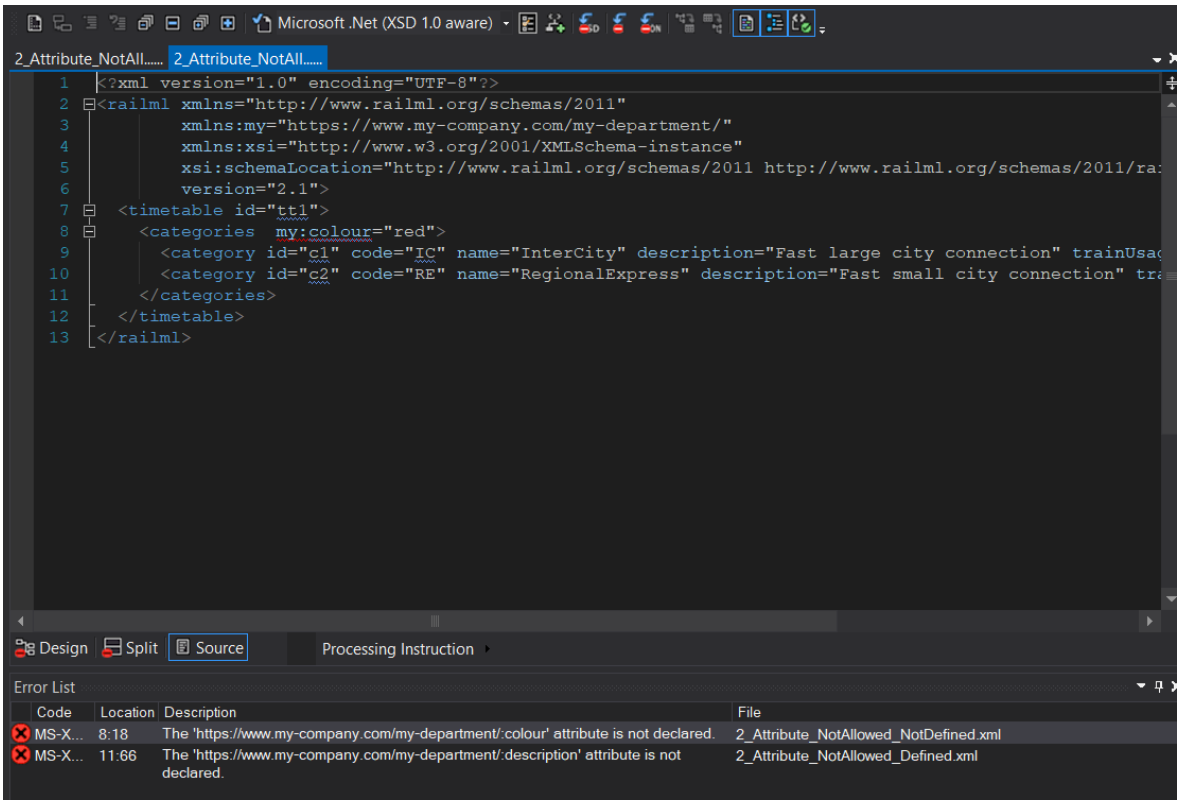


Рисунок 2.3 – Результати валідації Microsoft .Net (XSD 1.0 aware) файлів для ситуацій №5 та 6 в застосунку Liquid Studio

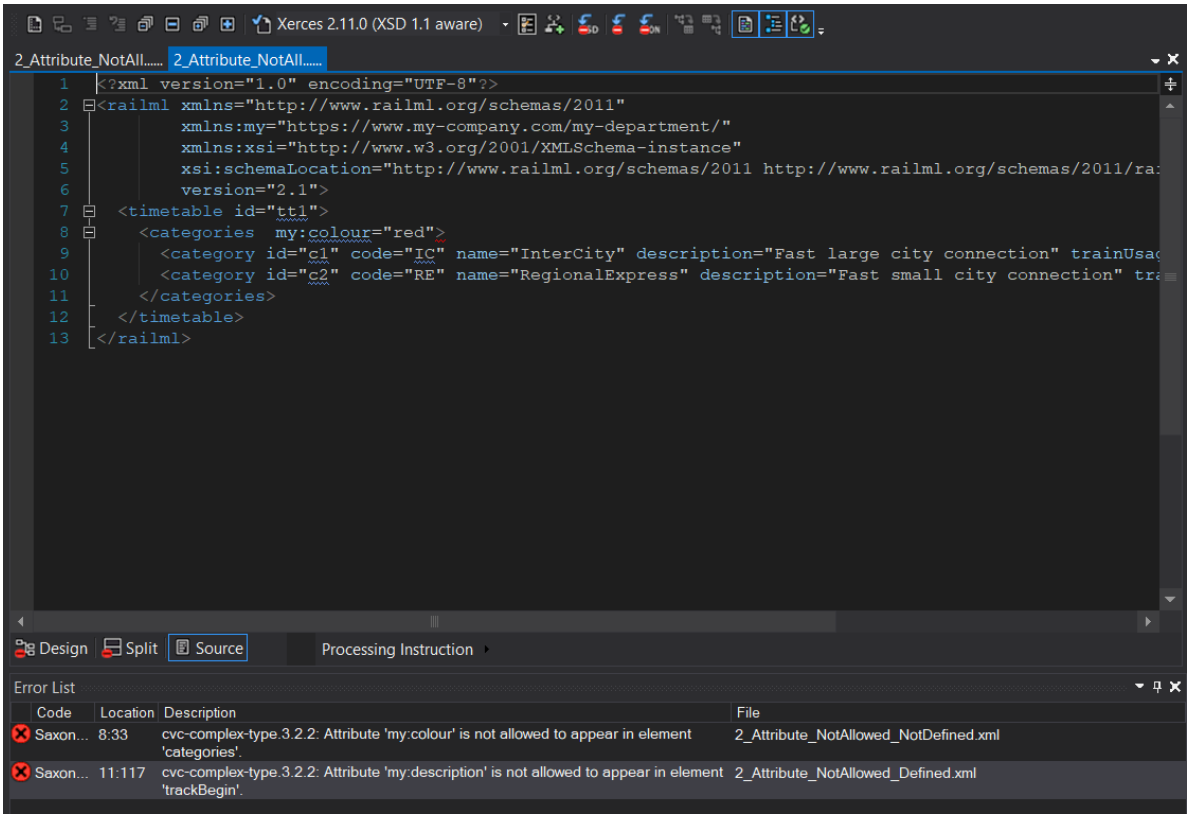


Рисунок 2.4 – Результати валідації Xerces 2.11.0 (XSD 1.1 aware) файлів для ситуацій №5 та 6 в застосунку Liquid Studio

Як можна помітити, хоч обидва валідатори знаходять помилку у файлі, результати валідатора Microsoft .NET повністю збігаються з результатами валідації стандартних бібліотек .NET, тобто текст помилки зазначає, що «атрибут не є оголошеним», однак неможливо зрозуміти, що саме її спричиняє. В цей час валідатор Xerces надає помилку з більш точним описом, а саме «атрибуту не дозволено з'являтись в елементі».

Для більш різноманітного порівняння також можна розглянути комерційний валідатор Altova XMLSpy, який працює відмінно від валідаторів .NET та Xerces:

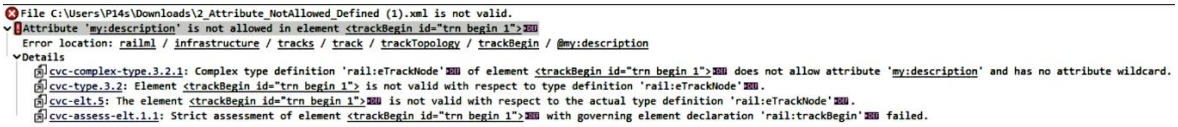


Рисунок 2.5 – Результати валідації файлу для ситуацій №5 в застосунку Altova XMLSpy

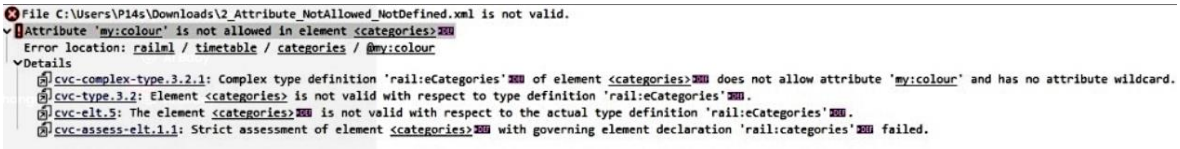


Рисунок 2.6 – Результати валідації файлу для ситуацій №6 в застосунку Altova XMLSpy

Як можна побачити вище, даний валідатор надає правильний очікуваний результат з набагато детальнішою інформацію, як, наприклад, пряме зазначення відсутності «wildcards», тобто елементів як <xs:any> та <xs:anyAttribute>, через що і виникає помилка при присутності неочікуваного атрибуту.

2.4 Гіпотеза №1 про присутність додаткової інформації в коді методу XDocument.Validate() та причини проблеми

В процесі аналізу роботи валідаторів було зроблено припущення, що код валідатора XDocument може мати інформацію, яку можна використати для створення власного більш точного повідомлення користувачу. При дебагінгу було знайдено головну в визначенні статусу атрибуту функцію:

```
internal SchemaAttDef? GetAttributeXsd(SchemaElementDecl? ed,
XmlQualifiedName qname, XmlSchemaObject? partialValidationType,
out AttributeMatchState attributeMatchState)
```

```
   {
       SchemaAttDef? attdef = null;
       attributeMatchState =
AttributeMatchState.UndeclaredAttribute;
       if (ed != null)
       {
           attdef = ed.GetAttDef(qname);
           if (attdef != null)
           {
               attributeMatchState =
AttributeMatchState.AttributeFound;
               return attdef;
           }
           XmlSchemaAnyAttribute? any = ed.AnyAttribute;
           if (any != null)
           {
               if (!any.NamespaceList!.Allows(qname))
               {
                   attributeMatchState =
AttributeMatchState.ProhibitedAnyAttribute;
               }
               else if (any.ProcessContentsCorrect !=
XmlSchemaContentProcessing.Skip)
               {
                   if (_attributeDecls.TryGetValue(qname, out
attdef))
                   {
                       if (attdef.Datatype.TypeCode ==
XmlTypeCode.Id)
                       { //anyAttribute match whose type is ID
                           attributeMatchState =
AttributeMatchState.AnyIdAttributeFound;
                       }
                       else
                       {
```

```
                            attributeMatchState =
AttributeMatchState.AttributeFound;
                        }
                    }
                    else if (any.ProcessContentsCorrect ==
XmlSchemaContentProcessing.Lax)
                    {
                        attributeMatchState =
AttributeMatchState.AnyAttributeLax;
                    }
                }
                else
                {
                    attributeMatchState =
AttributeMatchState.AnyAttributeSkip;
                }
            }
            else if (ed.ProhibitedAttributes.ContainsKey(qname))
            {
                attributeMatchState =
AttributeMatchState.ProhibitedAttribute;
            }
        }
        else if (partialValidationType != null)
        {
            XmlSchemaAttribute? attr = partialValidationType as
XmlSchemaAttribute;
            if (attr != null)
            {
                if (qname.Equals(attr.QualifiedName))
                {
                    attdef = attr.AttDef;
                    attributeMatchState =
AttributeMatchState.AttributeFound;
                }
```

```
                else
                {
                    attributeMatchState =
AttributeMatchState.AttributeNameMismatch;
                }
            }
            else
            {
                attributeMatchState =
AttributeMatchState.ValidateAttributeInvalidCall;
            }
        }
        else
        {
            if (_attributeDecls.TryGetValue(qname, out attdef))
            {
                attributeMatchState =
AttributeMatchState.AttributeFound;
            }
            else
            {
                attributeMatchState =
AttributeMatchState.UndeclaredElementAndAttribute;
            }
        }
        return attdef;
    }
```

Дослідивши та протестувавши наведену функцію було знайдено причину не коректного повідомлення, для цього було проаналізовано ключову частину алгоритму для проблематичних ситуацій №5 та 6:

1) спершу атрибуту надається статус «не оголошений»;

2) якщо атрибут знайдено в оголошенні елементу схеми, то цей елемент вважається дозволеним;

3) якщо ні, то перевіряється наявність елементу <xs:any> або <xs:anyAttribute>, якщо знайдено:

a. перевіряється визначення атрибуту у його схемі, якщо знайдено:

i. атрибут набуває статусу «дозволений»;

b. якщо ні, то атрибут повертається зі статусом «не оголошений»;

4) якщо ні, то перевіряється чи атрибут входить до безпосередньо визначених «заборонених» атрибутів за схемою;

5) інакше атрибут повертається зі статусом «не оголошений»

Проблема з даним алгоритмом є у тому, що статус за замовченням атрибуту надається «не оголошений», що є основою для тексту повідомленням для ситуацій №5 та 6, так як у проблематичних випадках атрибути не знайдено в основній схемі railML, елемент <xs:any> для них також відсутній, і вони не є безпосередньо визначеними в схемі як «заборонені».

Недолік цього алгоритму є у тому, що якщо атрибут є дозволеним елементом <xs:any>, але його визначення у схемі не знайдено (наприклад, схема розширення для атрибуту відсутня), то тоді його статус також залишиться «не оголошений» за замовчуванням, що і створює однакове повідомлення у досить різних ситуаціях.

Можливим рішенням такої проблеми було б надання іншого статусу за замовченням – «не дозволений», і лише у випадку відсутності схеми для атрибуту змінювати його на «не оголошений», що гарантувало б більш точне повідомлення у таких ситуаціях.

2.5 Гіпотеза №2 про використання схожого алгоритму всіма стандартними бібліотеками

Виходячи з результатів отримання однакового повідомлення всіма трьома методами у ситуаціях №5 та 6 було зроблено припущення, що методи XDocument.Validate(), XMLDocument.Validate() та XMLReader використовують однаковий алгоритм для валідації файлів, який було розглянуто вище. Під час процесу дебагінгу було досліджено внутрішній код всіх трьох функцій. Було підмічено, що, хоч на перший погляд вони використовували різні функції,

перевірка на основі аналізу коду XDocument показала, що всі три методи в якийсь момент валідації зупиняються на методі «GetAttributeXsd», в якому і створюється не повністю коректне оброблення ситуації та повідомлення.

Висновки до розділу 2

В другому розділі відбувався поверхневий огляд та тестування доступних бібліотек для валідації railML файлі у .NET. В результаті попереднього тестування особливих для railML ситуацій було знайдено проблему у роботі обраних функцій. В різних розглянутих ситуаціях було отримано повідомлення «атрибут не є оголошеним», хоча в деяких випадках атрибут був не дозволеним за схемою railML, що є жорсткою помилкою в порівнянні з відсутністю визначення атрибутів (у випадку розширення railML користувацькою схемою), що має вважатись лише попередженням. Таким чином було визначено, що при однаковому повідомленні та при поєднанні різних ситуацій неможливо визначити реальну причину проблеми та метод її оброблення.

Для порівняння можливих результатів при такій ситуації було розглянуто різне програмне забезпечення та валідатори: railVIVID 1 (Xerces), Liquid Studio (Microsoft .NET, Xerces) та Altova XMLSpy. Особлива увага звертається на railVIVID 1, тому що його результати є обов'язковим вимогою для нового розроблюваного застосунку, так як розроблюване програмне забезпечення не має в своєму функціонуванні поступатись своєму попереднику. В процесі аналізу було визначено, що більш точне повідомлення про недозволенність атрибуту для проблематичних ситуацій можливе (результати Xerces для railVIVID 1 та Liquid Studio; Altova XMLSpy), однак результати Liquid Studio для Microsoft .NET збігались з результати попереднього тестування знайдених бібліотек.

Було зроблено спробу в пошуку рішення даної проблеми – аналіз бібліотек для XDocument, а саме функції, яка визначає статус атрибутів. Було також визначено, що ця функція використовується всіма стандартними бібліотеками валідації. В процесі аналізу алгоритму функції було знайдено причину не коректного статусу атрибуту – надання йому статусу «не оголошений» за замовчуванням. Було підмічено, що функція могла мати інший алгоритм, однак розробники не врахували розрізнення різних ситуацій оголошення та дозволеності атрибуту.

В результаті було визначено, що дана інформація не несе користі для власного визначення ситуації та створенні власного повідомлення. Таким чином можна розглянути два рішення: зміна алгоритму бібліотеки для валідації .NET або розробка власного алгоритму перевірки статусу атрибуту. Так як зміна методу не дозволить легко отримувати нові оновлення для цієї бібліотеки в .NET, було визначено, що необхідна розробка власного рішення для цієї проблеми.

РОЗДІЛ 3 РОЗРОБКА ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ДЛЯ ДОСЛІДЖЕННЯ ВАЛІДАЦІЇ RAILML ФАЙЛІВ

Метою даної розробки є створення бібліотек, що надають можливість порівняння валідації railML документів за допомогою різних стандартних бібліотек середовища .NET, а також розробка рішення проблеми пов’язаної з цими бібліотеками. Для цього необхідна розробка Unit-тестів для перевірки роботи коду й аналізу роботи функцій, а також простого настільного застосунку для легкого перегляду результатів.

3.1 Формалізація задачі

Основними задачами розробки є розробка якісної архітектури програмного забезпечення для подальшого повторного використання без дуплікації коду та порівняння доступних стандартних функції для валідації в .NET.

Добре спроектована архітектура є дуже важливим етапом, так як вона визначає можливість подальшої підтримки системи, внесення необхідних змін у неї та читабельності створеного на її основі коду. Як показує досвід аналізу railVIVID 1 (результати якого були представлені на 45-й railML Конференції [7]), чим складніше розроблено програмне забезпечення та чим менше воно було попередньо сплановано, тим більше зростає дуплікація коду, кількість залежностей, використання анти-патернів та недоцільних практик, що ускладнює розуміння коду, будь-які його виправлення та роботу над ним у майбутньому [21]. Для правильного проектування необхідно розробити UML діаграми системи та компонентів, що надають можливість попередньо побачити та усунути небажані залежності. Правильно розроблені бібліотеки можливо буде використати для створення офіційної версії настільного застосунку railVIVID 2, його майбутньої веб-версії або іншого програмного забезпечення.

Розроблені бібліотеки та програмне забезпечення мають надавати користувачу можливість валідації файлів різними бібліотеками (див. Рис 3.1), також переглядати результати даної валідації (див. Рис. 3.2).

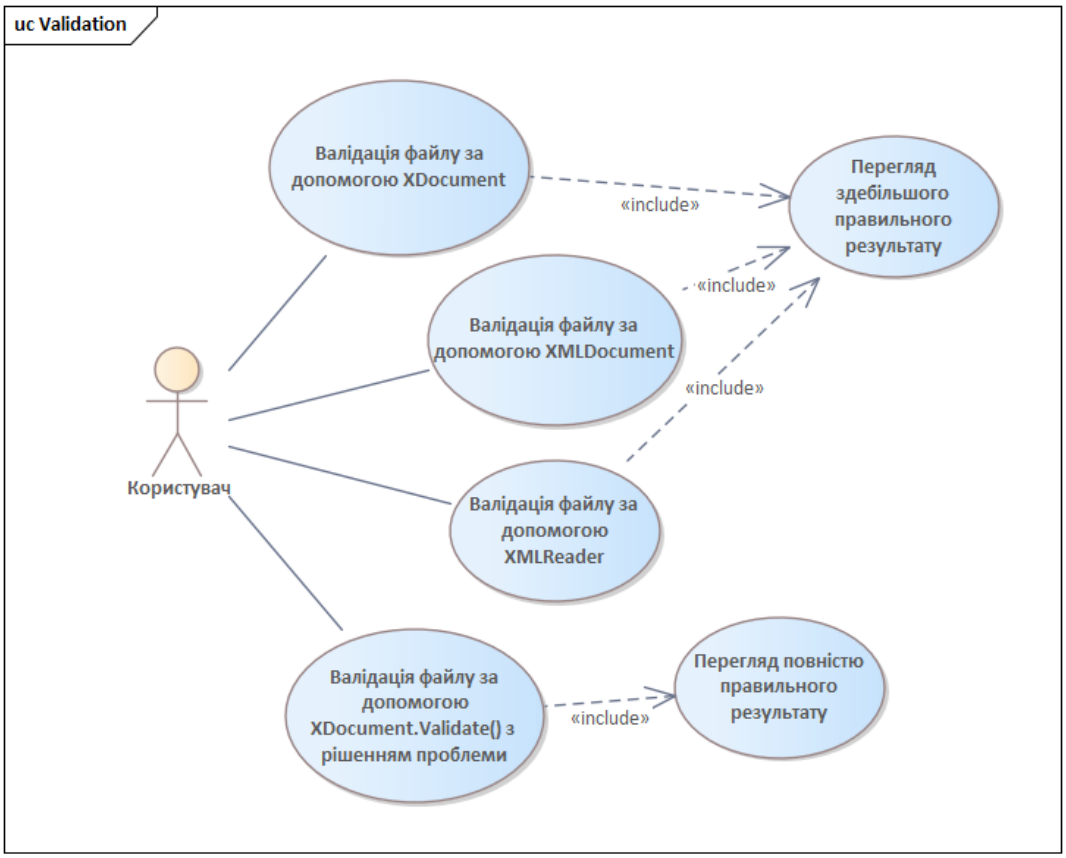


Рисунок 3.1 — Діаграма діяльності валідації railML файлу

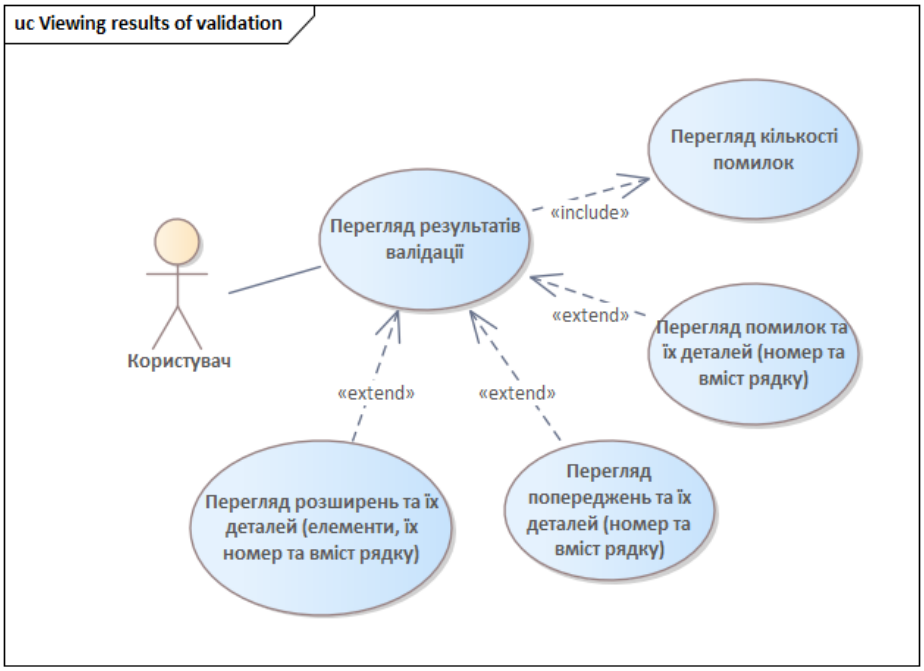


Рисунок 3.2 — Діаграма діяльності перегляду результатів валідації railML файлу

3.2 Базова архітектура системи

Як верхній рівень архітектури було спершу розроблено діаграму компонентів системи, щоб мати можливість уявити взаємодію її складових (див. Рис. 3.3):

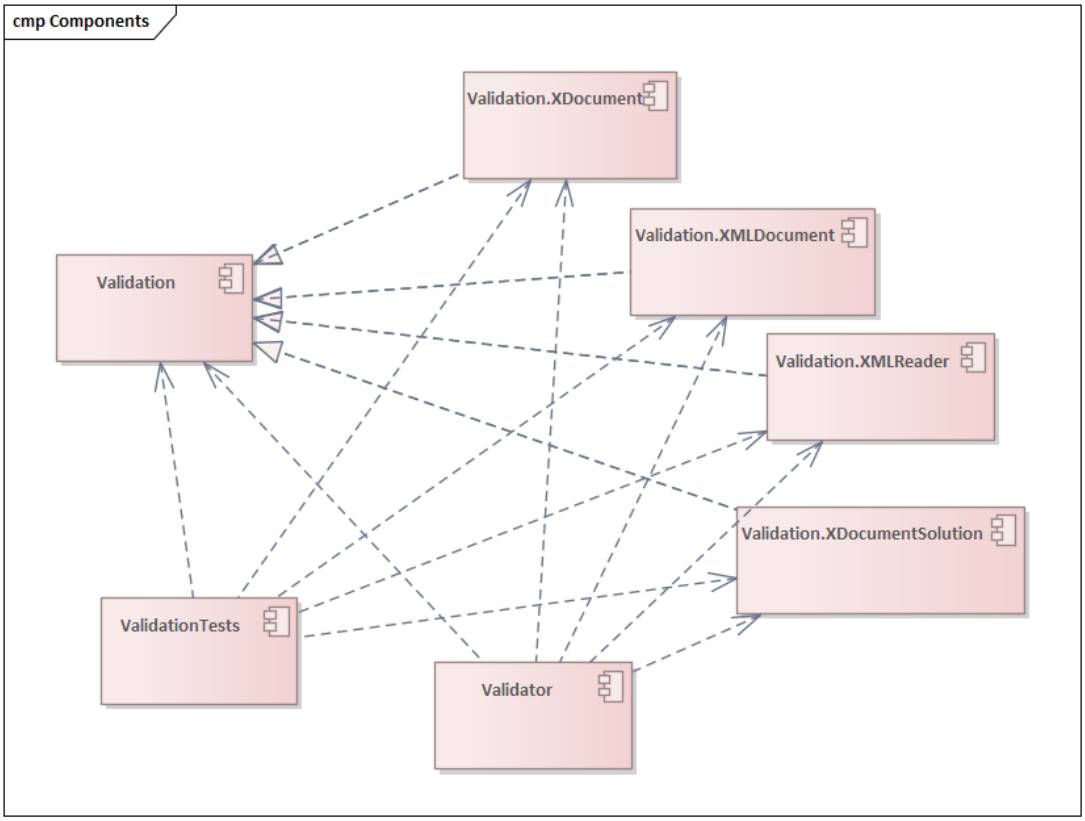


Рисунок 3.3 — Діаграма компонентів системи

На ній можна побачити, що під час розробки планується створити 4 основі бібліотеки та два проекти, які будуть їх використовувати. Таким чином проект має такі основні модулі та зв’язки:

Таблиця 3.1 – Основні модулі та зв’язки проекту

Номер	**Назва**	**Використовує**
1	Validation	–
2	Validation.XDocument	Validation
3	Validation.XMLDocument	Validation
4	Validation.XMLReader	Validation
5	Validation.XDocumentSolution	Validation

6	ValidationTests	Validation, Validation.XDocument, Validation.XMLDocument, Validation.XMLReader, Validation.XMLReader
7	Validator	Validation, Validation.XDocument, Validation.XMLDocument, Validation.XMLReader, Validation.XMLReader

1. Validation – відповідає за попередню обробку файлу (перевірка структури, формату), завантажує файл. Містить інтерфейс класу Validator та допоміжні функції для валідації.

2. Validation.XDocument – перший стандартний метод валідації за допомогою XDocument.Validate()[22] (має проблему валідації). Використовує дані отримані під час попередньої валідації; реалізує інтерфейс класу Validator.

3. Validation.XMLDocument – другий стандартний метод валідації за допомогою XMLDocument.Validate()[23] (має проблему валідації). Використовує дані отримані під час попередньої валідації; реалізує інтерфейс класу Validator.

4. Validation. XMLReader – третій стандартний метод валідації за допомогою XMLReader.Create()[24] (має проблему валідації). Використовує дані отримані під час попередньої валідації; реалізує інтерфейс класу Validator.

5. Validation.XDocumentSolution – доповнений перший метод валідації XDocument.Validate() з рішенням проблеми нечіткого повідомлення у трьох різних ситуаціях. Використовує дані отримані під час попередньої валідації; реалізує інтерфейс класу Validator.

6. ValidationTests – тести для перевірки справності програми та отримання результатів для дослідження.

7. Validator – настільний застосунок з графічним інтерфейсом, що дозволяє переглядати результати валідації.

3.3 Внутрішнє проектування

3.3.1 Вибір мови програмування та середовища розробки

Для вибору мови програмування та середовища розробки було враховано декілька факторів. Серед них зверталась увага на активну та велику спільноту для отримання відповідей та інформації, зрілість екосистеми, поновлювана підтримка нових версій та можливість повторного використання коду для настільного застосунку і веб версій, а також було отримано відгук від замовників щодо проведеного аналізу та рекомендацій. За різноманітними критеріями було розглянуто такі мови та технології, як Java та JavaFX/Swing, .NET з C# , QT з C++ та Python. В результаті цього аналізу було визначено, що платформа .NET 8 підходить для розробки найбільше, так як в порівнянні з іншими мовами та технологіями, вона має більше загальних переваг. Середовище .NET надає можливість легкого використання бібліотек для розробки настільних та веб застосунків завдяки єдності платформи, можливість легкої розробки інтерфейсу за допомогою Windows Forms для настільної версії, має Common Language Runtime, який забезпечує керування пам’яттю замість покладення відповідальності на розробника, як C++, має активну підтримку та спільноту, а також велику кількість доступних для використання бібліотек. C# є об’єктно-орієнтованою мовою, що сприяє повторному використанню коду, гнучкості і масштабованості розробки [25]. Для середовища розробки було обрано Visual Studio, яке часто використовується для розробки застосунків у .NET.

3.3.2 Ієрархія та взаємодія класів системи

Для більш детального планування системи програмного забезпечення було створено діаграму класів (див. Рис. 3.4):

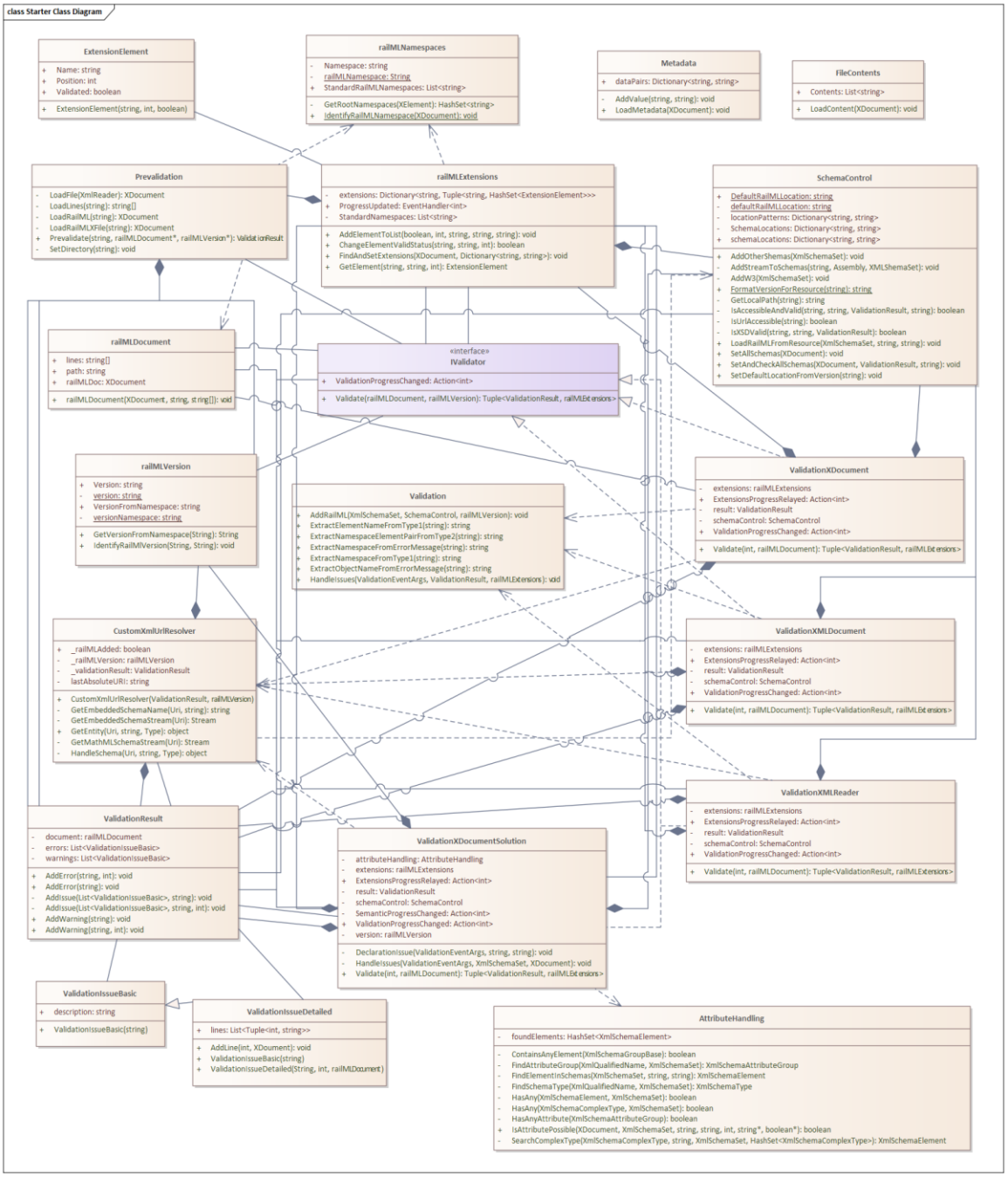


Рисунок 3.4 — Діаграма класів системи

3.3.3 Використані принципи та шаблони проектування

Для вирішення поставленої задачі, були використані наступні методи та алгоритми:

1. Стандартні методи валідації та роботи з XML файлами у .NET:

- XDocument.Load()
- XDocument.Validate();
- XMLDocument.Validate();
- XMLReader.Create().

2. Власний алгоритм перевірки невизначених атрибутів.

Загальний алгоритм роботи бібліотеки доповненого методу XDocument.Validate():

1. Додання всіх відповідних XSD схем до XMLSchemaSet, який буде використовуватись для валідації.
2. Знаходження всіх елементів, що відносяться до розширень.
3. Валідація.
4. Обробка проблем:
    a. Збереження в результат проблем з статусом попереджень.
    b. Якщо це помилка про «невизначений елемент/атрибут»:
        i. Якщо це атрибут – рекурсивно перевірити чи визначає елемент, до якого належить атрибут, в своїй схемі елемент <xs:any> або <xs:anyAttribute>. Якщо ні – обробити як помилку; якщо так – обробити як попередження (так як проблема невизначених елементів стосується лише розширень).
        ii. Якщо це не атрибут – додати попередження (так як проблема невизначених елементів стосується лише розширень).
    c. Збереження в результат інших проблем з статусом помилок.

3.4 Особливості та деталі імплементації railML валідації

Під час імплементації загальної валідації та обраних бібліотек .NET було виявлено деякі труднощі та особливості роботи з railML для задачі валідації.

Наприклад, при попередній валідації необхідно було врахувати не лише розширення файлів xml та railml, а і railmlx, який необхідно оброблювати як архів [26]. Під час цієї стадії валідації необхідна була також перевірка присутності простору імен railML, так як без нього файл не може вважатись відповідним даному формату даних.

Також під час основної валідації при завантаженні XSD до XsdSchemaSet з UrlResolver було підмічено, що він аналізує початкові додані схеми та також

завантажує всі імпортовані через них XSD. Через це виникає проблема, так як при валідації має використовуватись «set» саме з офіційними схемами railML, тобто схеми railML не мають додаватись через сторонні схеми. Було помічено, що при такому доданні іноді стає неможливим повне видалення цих схем з набору стандартними функціями, в такому випадку при доданні офіційних схем виникає помилка повторюваності схем. Рішення цієї ситуації було знайдено через модифікацію UrlResolver та додання офіційних схем до набору першими, та перевірки їх присутності в подальшому обробленні, щоб не допускати сторонні схеми railML.

Також було виявлено проблему з непостійним доступом до схем mathML, які використовуються під-областями railML, через що було зроблено рішення завжди використовувати локальні версії цих схем під час їх завантаження.

Під час валідації також потребувався пошук розширень, тобто всіх елементів та атрибутів, які не належать до простору імен railML.

І одною з найголовніших особливостей обробки знайдених проблем була згадана в Розділі 2 обробка повідомлень не оголошених елементів та атрибутів як попереджень, а не помилок, якими їх вважають стандартні валідатори. Це пояснюється існуванням таких проблем лише для елементів та атрибутів зі схем розширень, так як елементи railML завжди є оголошеними через обов’язкову доступність їх схем під час валідації.

Також особливого підходу потребувало рішення проблеми не чітких повідомлень для невизначених та недозволених атрибутів. Було виявлено, що для перевірки присутності <xs:any> або <xs:anyAttribute> для елементу не достатньо перевірити напряму визначені для нього атрибути. В деяких випадках в railML елемент може мати складний тип, визначений в іншому місці схеми, який, наприклад, також може посилатись на інший тип, який далі може визначати групу атрибутів, в якій можуть описуватись самі атрибути для початкового елементу. Тобто для знайдення атрибутів в такій ситуації необхідно прослідувати за «ланцюгом посилань», щоб знайти визначення атрибутів. Такі ситуації були враховані під час розробки алгоритму перевірки атрибутів для

нечітких повідомлень, в якому відбувається рекурсивна перевірка елементів складного типу для пошуку їх атрибутів.

3.5 Тестування та налагодження програми

В процесі тестування було проведено порівняння різноманітних методів валідації railML документів, в тому числі й з урахуванням розширення railML за допомогою сторонніх схем.

Було виконано перевірку вибірки тестових файлів для кожного методу окремо, а також протестовано деякі файли з проблемою нечіткого повідомлення «атрибут не визначено» в ситуації його недозволенності.

Результати загального тестування файлів для методу XDocument.Validate():

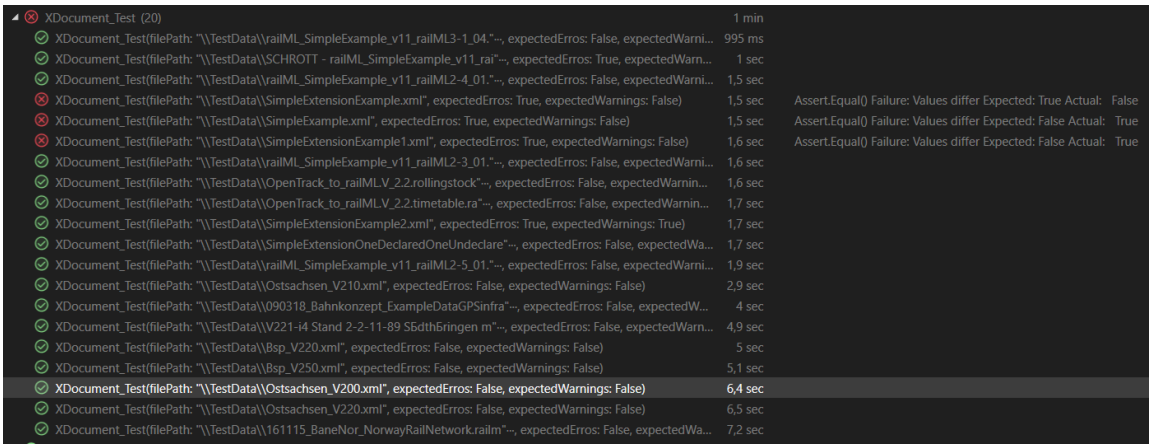


Рисунок 3.5 — Загальне тестування методу XDocument.Validate()

Результати загального тестування файлів для методу XMLDocument.Validate():

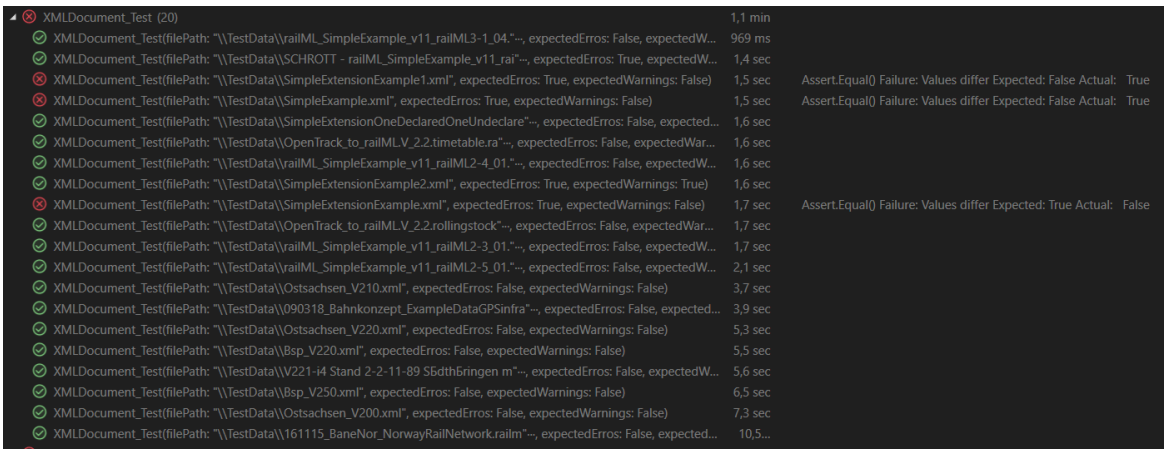


Рисунок 3.6 — Загальне тестування методу XMLDocument.Validate()

Результати загального тестування файлів для методу XMLReader.Create():

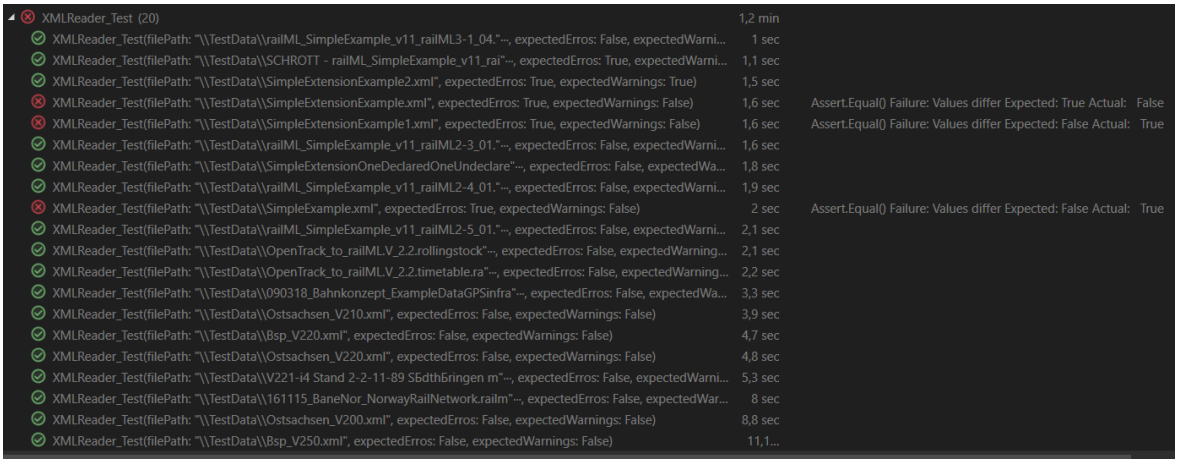


Рисунок 3.7 — Загальне тестування методу XMLReader.Create()

Результати загального тестування файлів для методу XDocument.Validate() з додатковим рішенням проблеми:

Рисунок 3.8 — Загальне тестування методу XDocument.Validate() з додатковим рішенням проблеми

3.6 Практичне застосування розробленого рішення у railVIVID 2

Розроблені основні бібліотеки та рішення на основі XDocument було імплементовано при розробці офіційного застосунку для валідації та вузуалізації файлів railML від railML.org – railVIVID 2 (див. Рис. 3.9), перша презентація якого відбулась під час 46-ї railML Конференції [27].

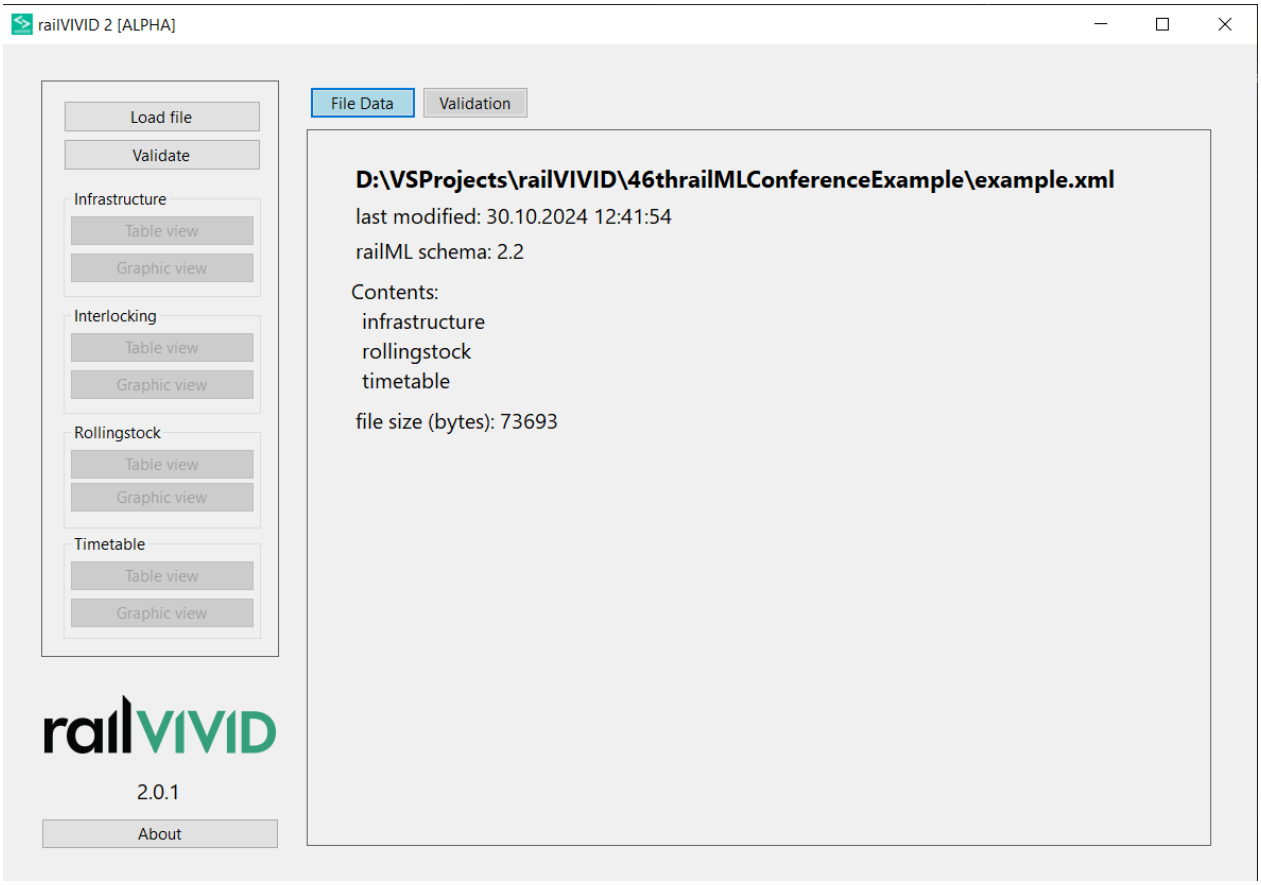


Рисунок 3.9 — Приклад railVIVID 2 (демонстраційна версія)

Дані бібліотеки також можливо буде використати при розробці майбутньої веб-версії цього застосунку, так як їх архітектура розроблювалась саме з розрахунком на такий сценарій повторного використання.

Висновки до розділу 3

В даному розділі відбувалось проектування, розробка та тестування програмного застосунку для отримання та перевірки результатів стандартних бібліотек для валідації XML файлів та рішення знайденої проблеми валідації на основі одного з методів. В процесі роботи було створено декілька бібліотек мовою C#, які можливо використати для створення застосунків в середовищі .NET, деякі з котрих були використані для розробки офіційного настільного застосунку railVIVID 2, який було презентовано під час 46-ї railML Конференції у листопаді 2024 року. Також варто зауважити, що дані бібліотеки також будуть застосовані при запланованій розробці веб-версії railVIVID 2. Під час створення програми для даної дипломної роботи було виявлено ряд особливостей імплементації валідації у середовищі .NET та знайдено рішення зустрітих проблем. В результаті загального тестування застосунку було підтверджено, що при тестуванні стандартних функцій для файлів з недозволеними атрибутами результат відрізняється від очікуваного, в той час як результати розробленого рішення є правильними, що можна детальніше розглянути у Розділі 4 для оглянутих у Розділі 2 ситуацій.

РОЗДІЛ 4 ДОСЛІДЖЕННЯ КОРЕКТНОСТІ МЕТОДІВ ВАЛІДАЦІЇ RAILML ДОКУМЕНТІВ

Для дослідження валідації було виконано більш детальне тестування трьох стандартних функцій та розробленого рішення проблеми на основі результатів XDocument.Validate(). Для цього було створено Unit-тести, а також розроблено настільний застосунок з графічним інтерфейсом для перегляду результатів.

4.1 Підготовка для експерименту

Частина підготовки до експерименту була виконана у Розділі 2 цієї роботи, а саме було створено документи, валідація яких буде тестуватись різними бібліотеками. Для перевірки було обрано 7 критичних ситуацій (див. Таблиця 2.1), правильна обробка яких є ключовою у правильному функціонуванні бібліотеки для валідації railML файлів. Для наглядної перевірки результатів функцій було створено Unit-тести для 3 бібліотек та власного розробленого рішення з особливим зверненням уваги на проблематичні сценарії. Також в Розділі 3 було створено застосунок, в якому можливо перевірити деталі результатів.

4.2 Опис підходу для визначення правильності роботи функцій

Перевірка бібліотек буде відбуватись на основі розглянутих у Розділі 2 критичних ситуацій (див. табл. 2.1) та розроблених для них файлів. Таким чином можна створити таблицю очікуваних результатів для всіх ситуацій:

Таблиця 4.1 – Очікувані результати валідації для основних ситуацій

Ситуація	Очікуваний тип проблеми	Результат валідації
1	Попередження	Документ правильний
2	Помилка	Документ не правильний
3	Помилка	Документ не правильний
4	Попередження	Документ правильний
5 (проблематична)	Помилка	Документ не правильний

5 (проблематична, зі схемою розширень)	Помилка	Документ не правильний
6 (проблематична)	Помилка	Документ не правильний
6 (проблематична, зі схемою розширень)	Помилка	Документ не правильний
7	Помилка	Документ не правильний

Кожен файл буде протестовано за допомогою трьох стандартних функцій, а також за допомогою розробленого рішення проблеми для ситуацій №5 та 6, при тестуванні через застосунок з графічним інтерфейсом можливо буде переглянути деталі валідації, а саме переглянути та порівняти повідомлення отримані або створені під час валідації.

4.3 Порівняння результатів за допомогою Unit-тестів

Для ситуації №1 виконується перевірка відсутності помилок та отримання попередження з текстом про не оголошеність елементу через відсутність схеми:

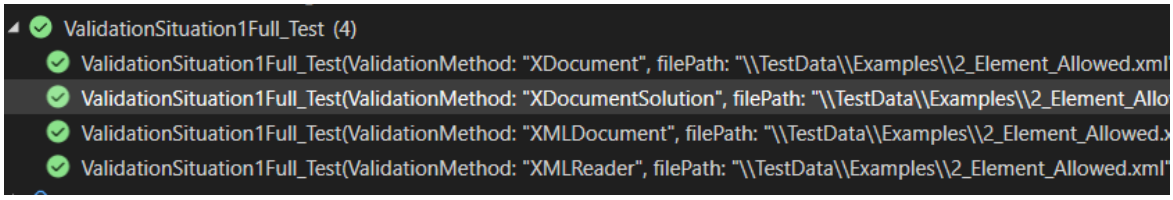


Рисунок 4.1 – Результати тестування файлу для ситуації №1 трьома стандартними бібліотеками та розробленим рішенням

Для ситуацій №2 та №3 виконується перевірка присутності помилок з текстом про недозволенність елементу через порушення схеми railML:

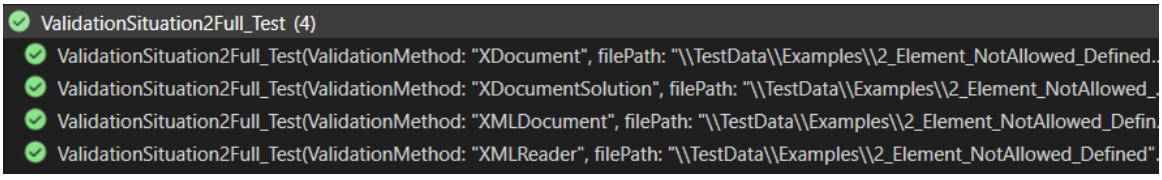


Рисунок 4.2 – Результати тестування файлу для ситуації №2 трьома стандартними бібліотеками та розробленим рішенням

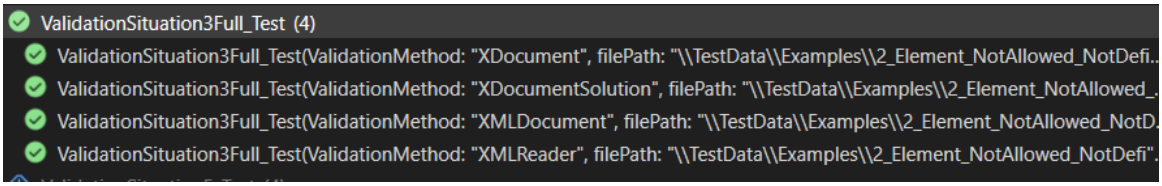


Рисунок 4.3 – Результати тестування файлу для ситуації №3 трьома стандартними бібліотеками та розробленим рішенням

Для ситуації №4 виконується перевірка відсутності помилок та отримання попередження з текстом про не оголошеність атрибуту через відсутність схеми:

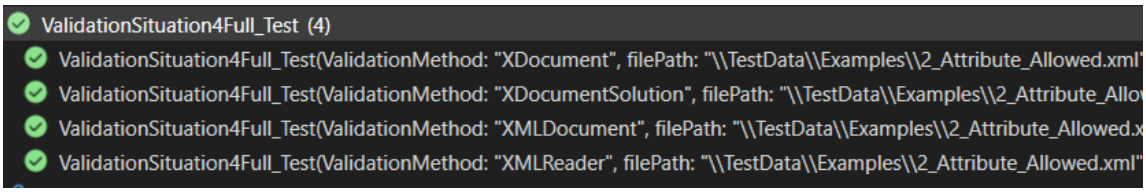


Рисунок 4.4 – Результати тестування файлу для ситуації №4 трьома стандартними бібліотеками та розробленим рішенням

Для ситуації №5 та №6 (проблематичні випадки) також виконується перевірка присутності помилок з текстом про недозволенність атрибуту через порушення схеми railML.

Спершу переглянемо чи знайдені були помилки:

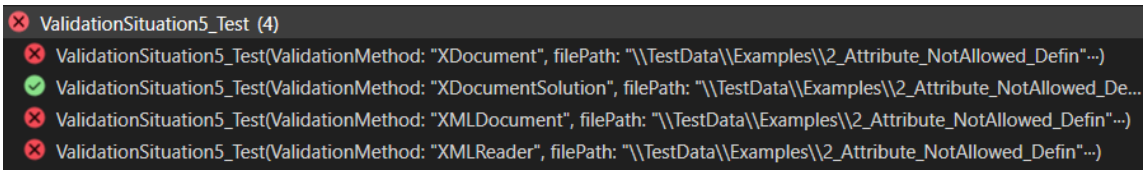


Рисунок 4.5 – Результати тестування файлу для ситуації №5 трьома стандартними бібліотеками та розробленим рішенням (перевірка кількості помилок)

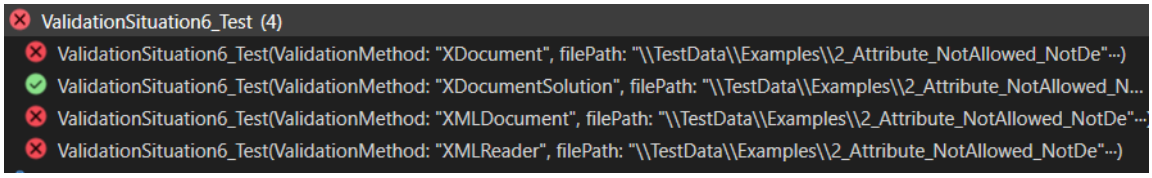


Рисунок 4.6 – Результати тестування файлу для ситуації №6 трьома стандартними бібліотеками та розробленим рішенням (перевірка кількості помилок)

Такі результати означають, що всі стандартні методи не визначили ніяких помилок для ситуацій №5 та №6 (враховуючи власну обробку проблем

невизначених атрибутів та елементів як попереджень через їх відношення до проблем розширень). Також можна помітити, що метод з доданням власного рішення дійсно дозволяє знайти помилку.

Детальніше це можна перевірити за допомогою пошуку тексту «не дозволено» в повідомленнях всіх знайдених проблем:

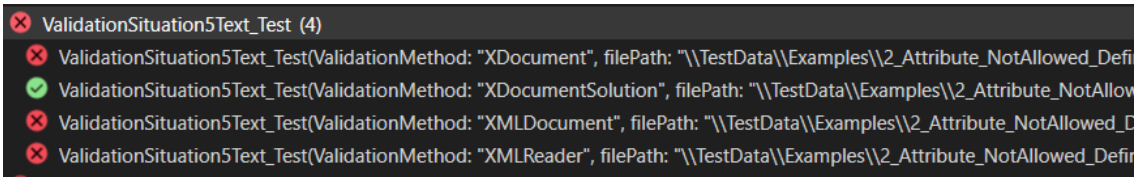


Рисунок 4.7 – Результати тестування файлу для ситуації №5 трьома стандартними бібліотеками та розробленим рішенням (перевірка тексту повідомлень)

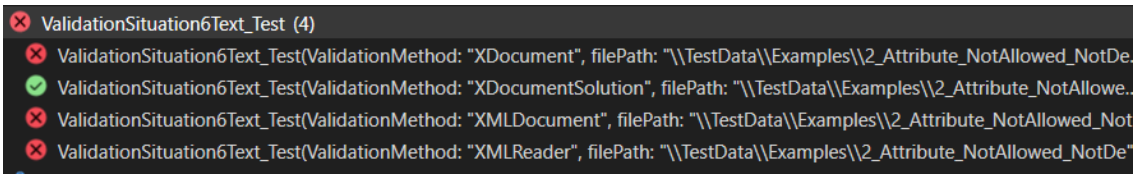


Рисунок 4.8 – Результати тестування файлу для ситуації №6 трьома стандартними бібліотеками та розробленим рішенням (перевірка тексту повідомлень)

Тобто з перевіркою тексту виникає та ж сама ситуація: результати всіх стандартних методів не містять тексту про заборону атрибуту, в той час рішення дозволяє визначити, що атрибут є недозволеним за схемою railML.

Щоб переконатися, що ситуація виникає не через відсутність визначення атрибутів через відсутність їх схеми, а саме через порушення схеми railML, розглянемо сценарії №5 та 6 доповнені місцезнаходженням схеми та чи буде все ще виникати повідомлення про оголошення:

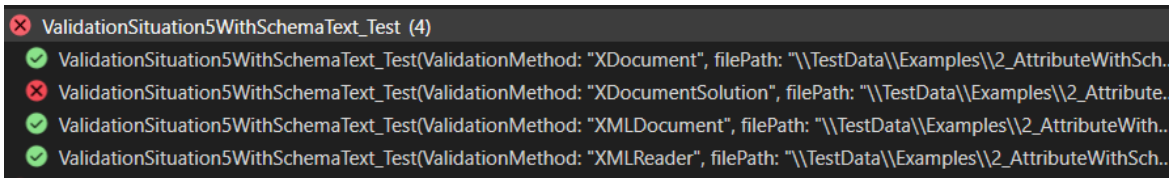


Рисунок 4.9 – Результати тестування файлу для ситуації №5 трьома стандартними бібліотеками та розробленим рішенням (з доданням схеми розширень)

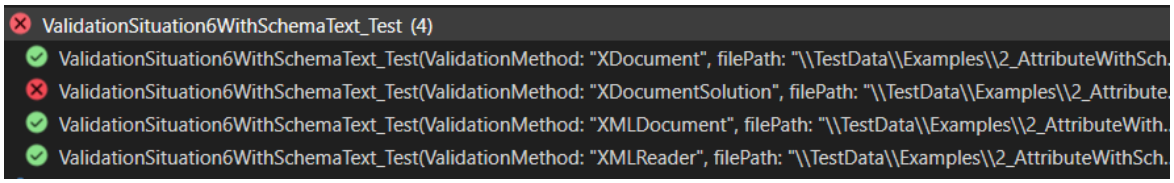


Рисунок 4.10 – Результати тестування файлу для ситуації №6 трьома стандартними бібліотеками та розробленим рішенням (з доданням схеми розширень)

Тобто при присутності схеми атрибутів виникає зворотна перевірка на текст «не оголошено», в якій стандартні методи все ще мають таке повідомлення як реакція на недозволенність атрибутів, в той час як метод з рішенням проблеми має інше повідомлення. Це означає, що такий текст дійсно спричинений атрибутом в неналежному місці, а не через реальну відсутність визначення у схемі.

Останньою є ситуація №7, в якій виконується перевірка присутності помилок, так як хоч <xsi:type> дозволяється використовувати для будь-яких елементів, якщо не можливо визначити схему для типу, це вважається жорсткою помилкою:

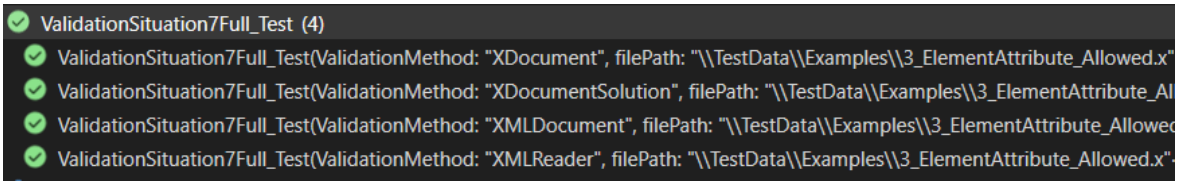


Рисунок 4.11 – Результати тестування файлу для ситуації №7 трьома стандартними бібліотеками та розробленим рішенням

4.4 Порівняння результатів за допомогою розробленого застосунку

В Розділі 3 було створено застосунок, який дозволяє переглянути деталі валідації всіх трьох стандартних бібліотек та розробленого рішення проблеми, тож можливо розглянути його результати для всіх описаних важливих ситуацій.

Ситуація №1.

Валідація стандартними бібліотеками:

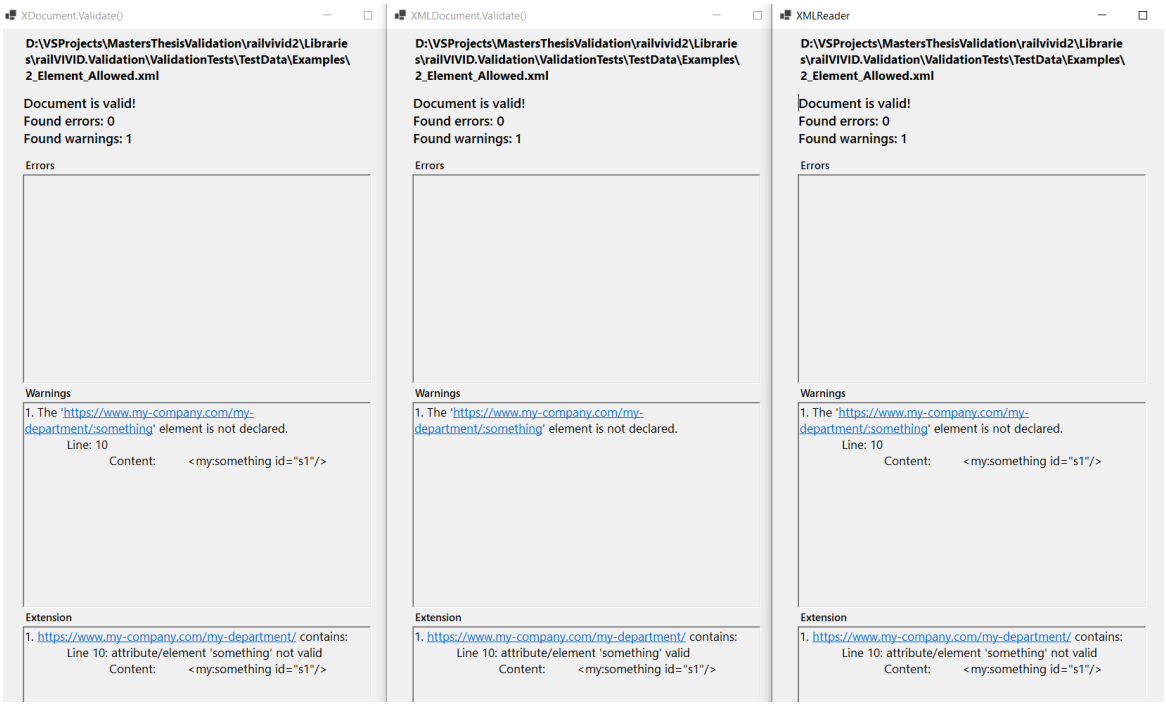


Рисунок 4.12 – Результати тестування файлу для ситуації №1 трьома стандартними бібліотеками

Валідація розробленим рішенням на основі XDocument.Validate():

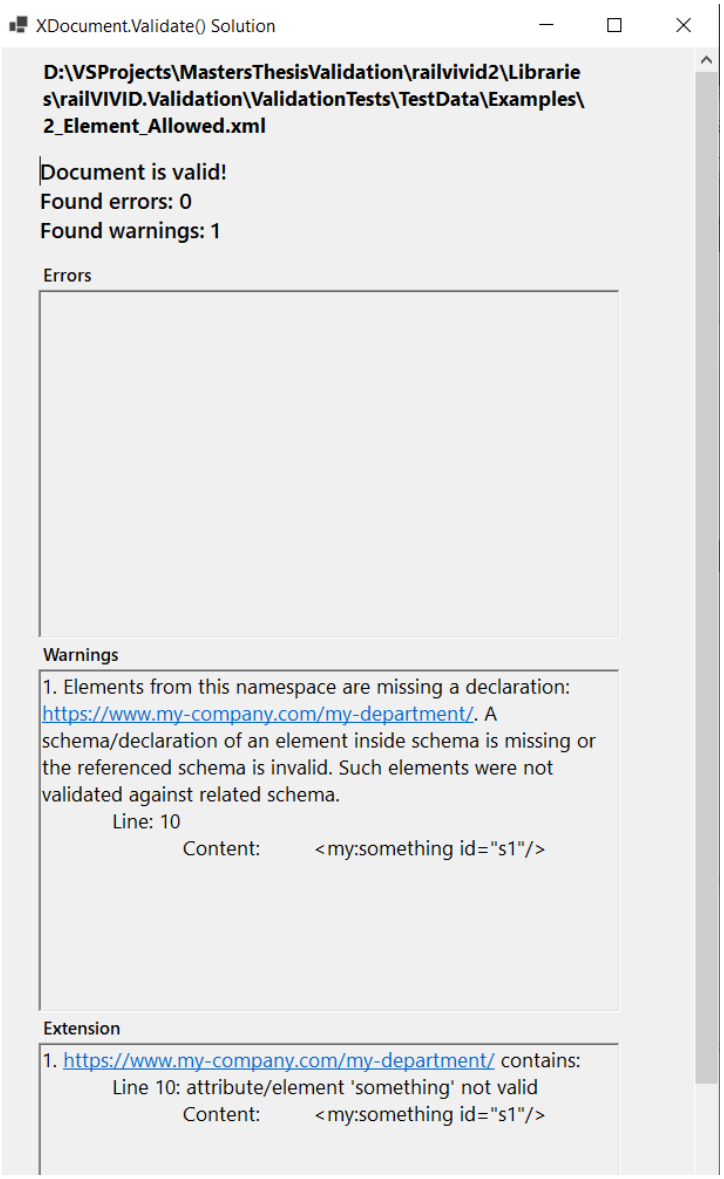


Рисунок 4.13 – Результати тестування файлу для ситуації №1 розробленим рішенням

Ситуація №2.

Валідація стандартними бібліотеками:

Рисунок 4.14 – Результати тестування файлу для ситуації №2 трьома стандартними бібліотеками

Валідація розробленим рішенням на основі XDocument.Validate():

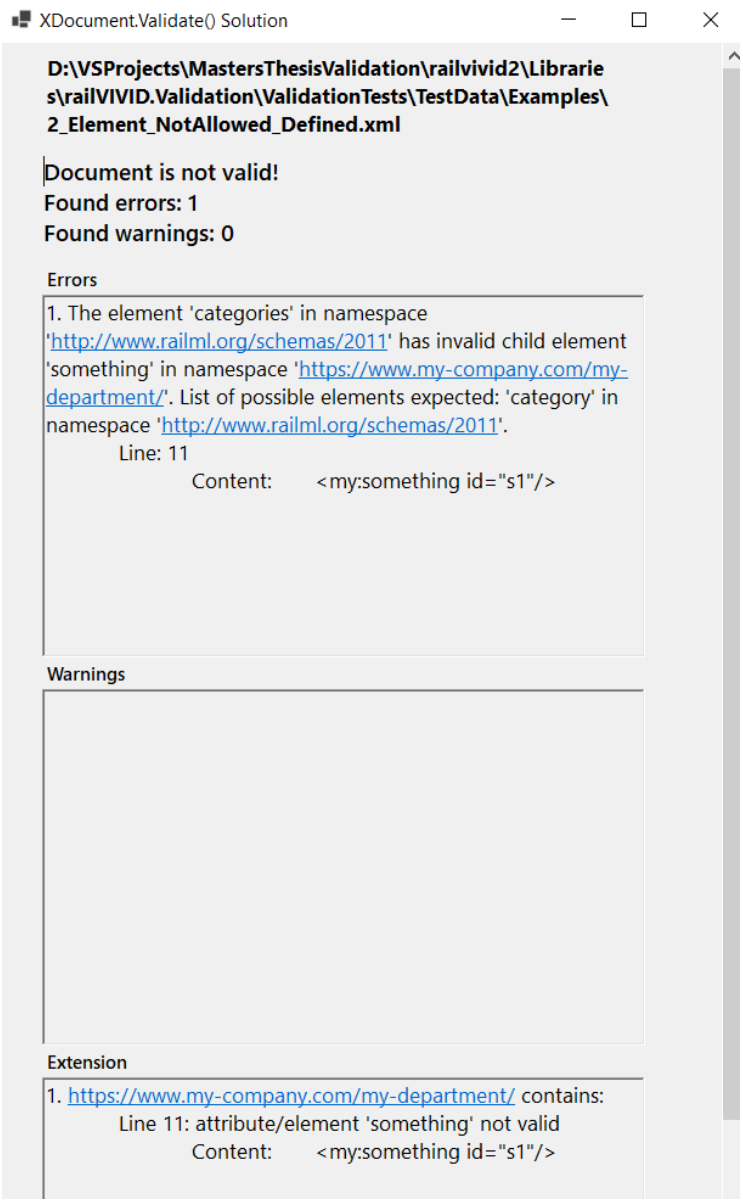


Рисунок 4.15 – Результати тестування файлу для ситуації №2 розробленим рішенням

Ситуація №3.

Валідація стандартними бібліотеками:

Рисунок 4.16 – Результати тестування файлу для ситуації №3 трьома стандартними бібліотеками

Валідація розробленим рішенням на основі XDocument.Validate():

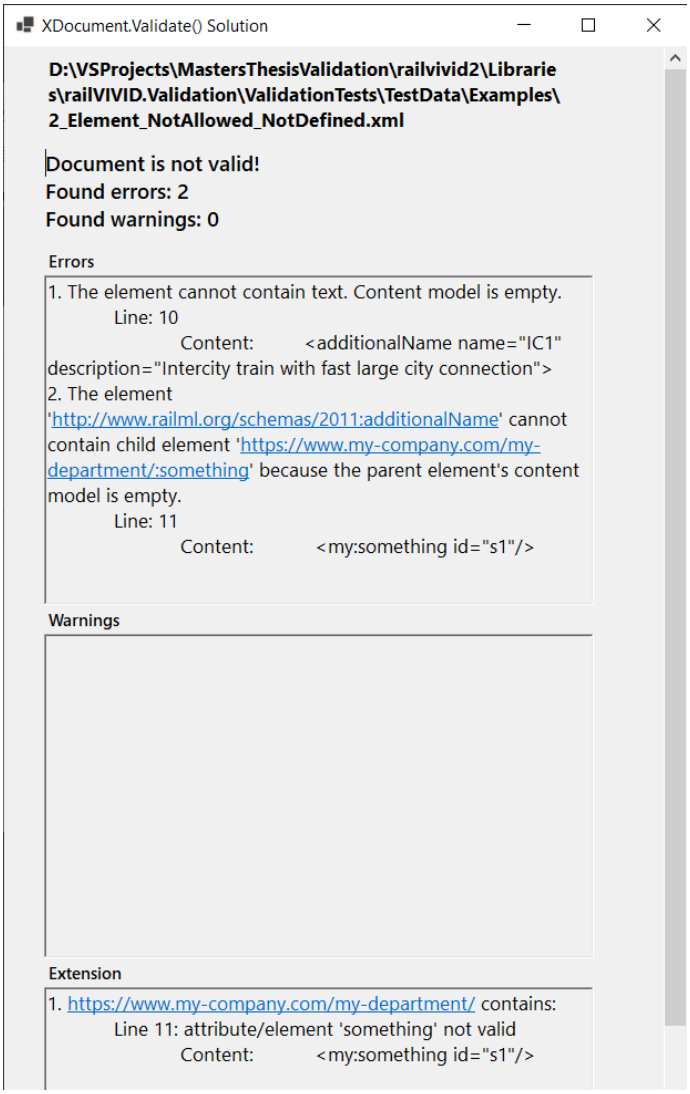


Рисунок 4.17 – Результати тестування файлу для ситуації №3 розробленим рішенням

Для валідаторів XDocument та XMLReader можна помітити, що в результатах присутньо дві помилки, перша з яких зазначає, що «елемент не може мати текст». Шляхом тестуванням було визначено, що ця помилка спричинена інтерпретацією пробілу як тексту. Така реакція валідатора є дивною, її причини та рішення знайдено не було.

Ситуація №4.

Валідація стандартними бібліотеками:

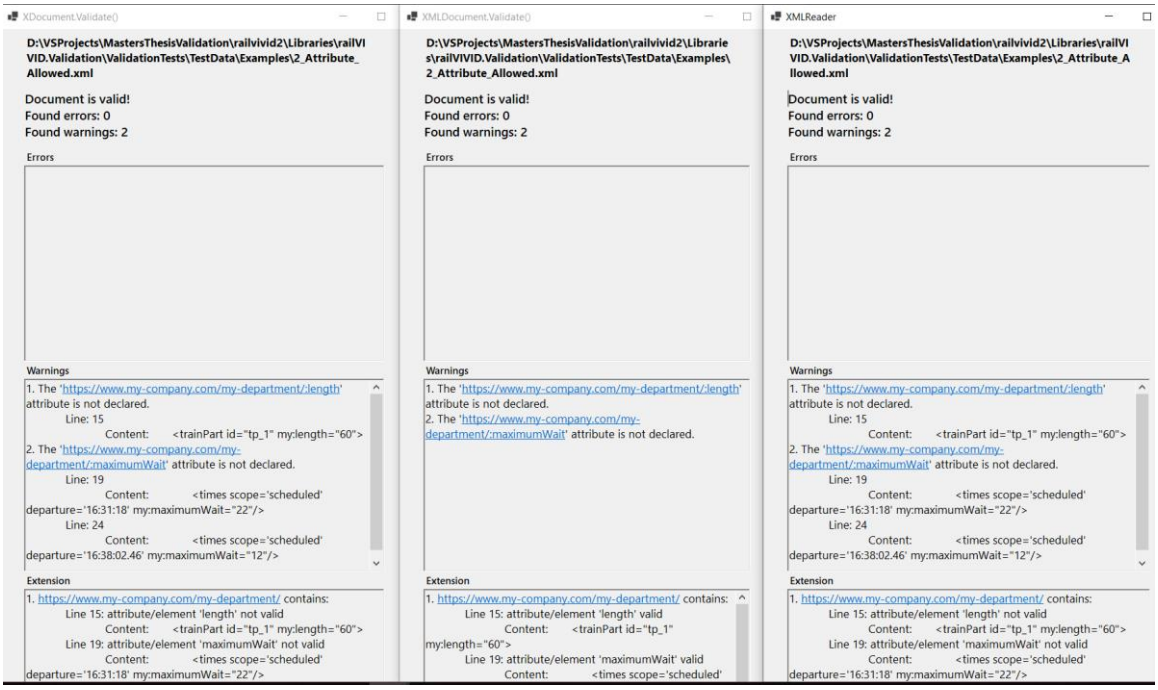


Рисунок 4.18 – Результати тестування файлу для ситуації №4 трьома стандартними бібліотеками (початковий вигляд)

Рисунок 4.19 – Результати тестування файлу для ситуації №4 трьома стандартними бібліотеками (вигляд з деталями розширень)

Валідація розробленим рішенням на основі XDocument.Validate():

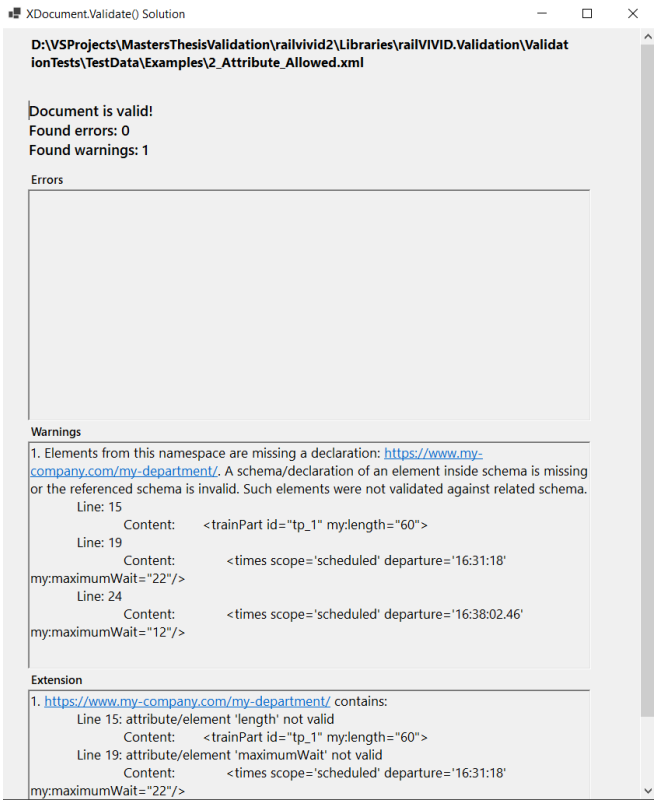


Рисунок 4.20 – Результати тестування файлу для ситуації №4 розробленим рішенням (початковий вигляд)

Рисунок 4.21 – Результати тестування файлу для ситуації №4 розробленим рішенням (вигляд з деталями розширень)

Ситуація №5.

Валідація стандартними бібліотеками:

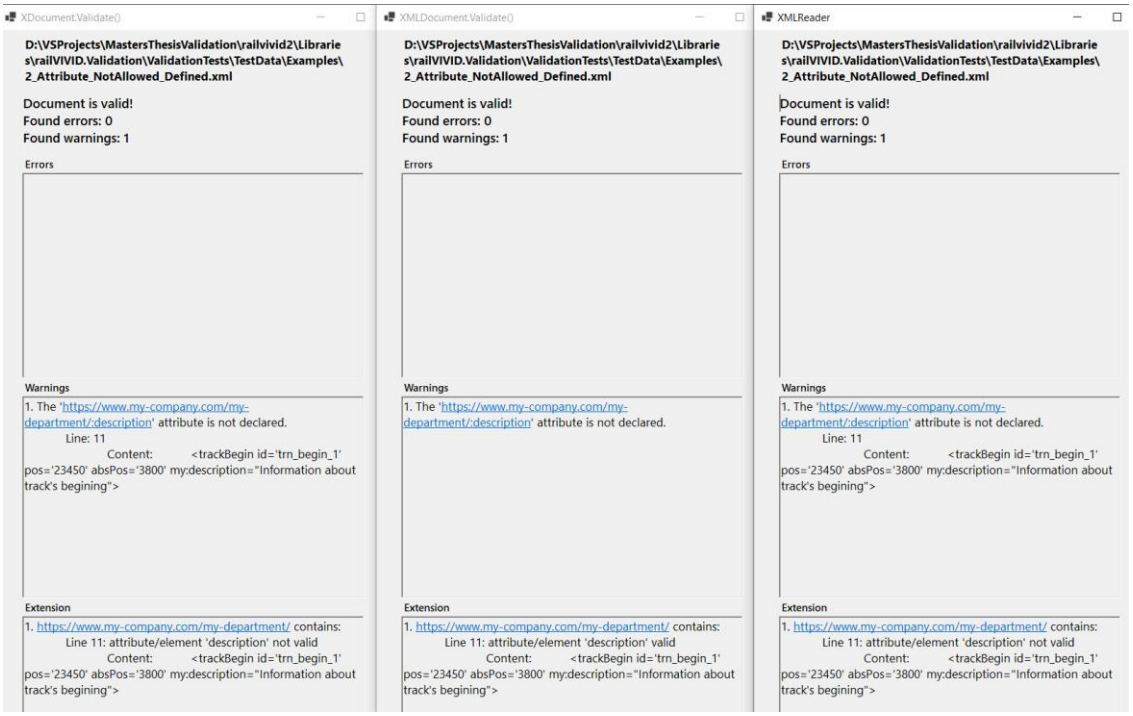


Рисунок 4.22 – Результати тестування файлу для ситуації №5 трьома стандартними бібліотеками

Валідація розробленим рішенням на основі XDocument.Validate():

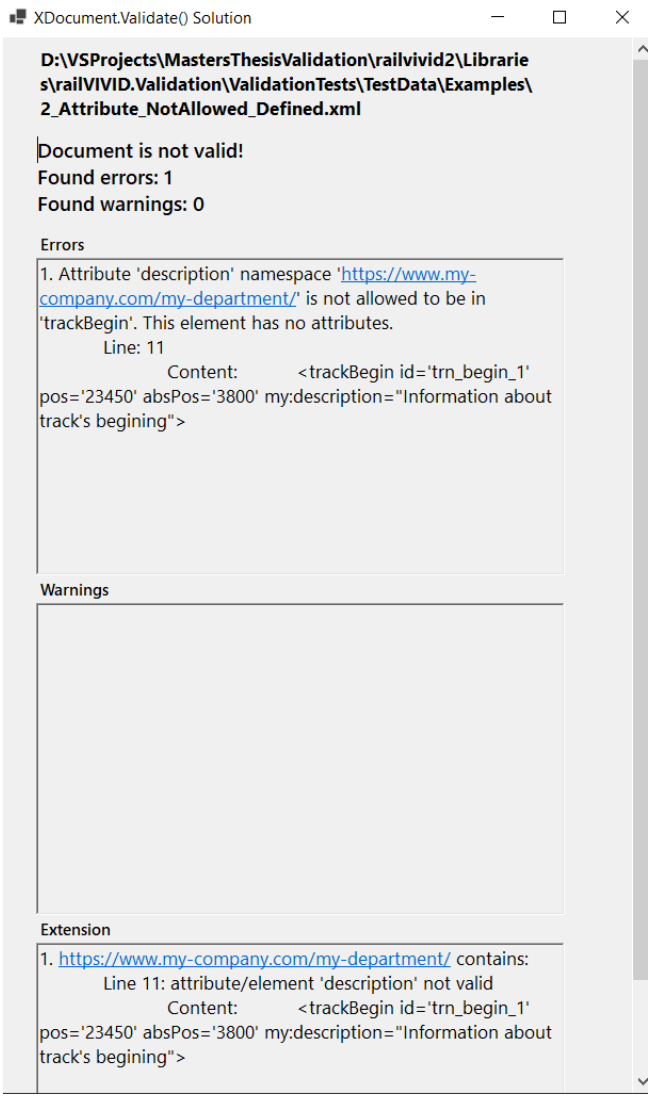


Рисунок 4.23 – Результати тестування файлу для ситуації №5 розробленим рішенням

Ситуація №5 (з схемою розширень).

Валідація стандартними бібліотеками:

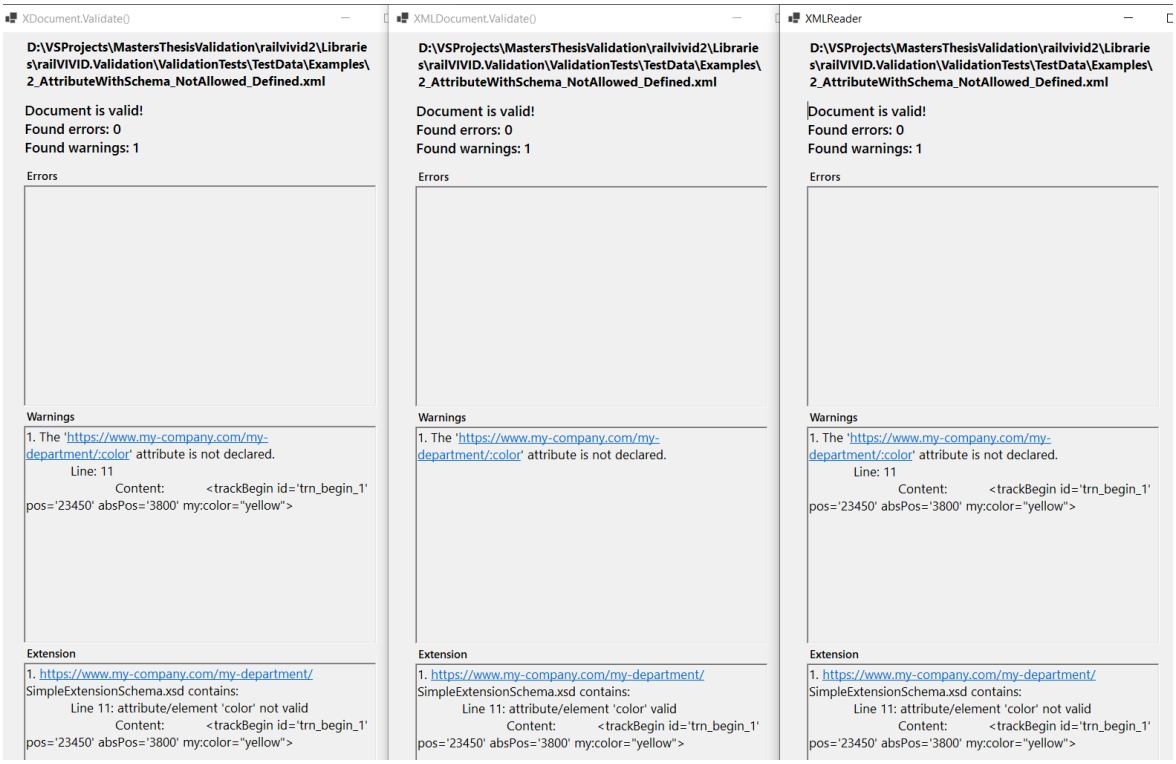


Рисунок 4.24 – Результати тестування файлу для ситуації №5 (зі схемою розширення) трьома стандартними бібліотеками

Валідація розробленим рішенням на основі XDocument.Validate():

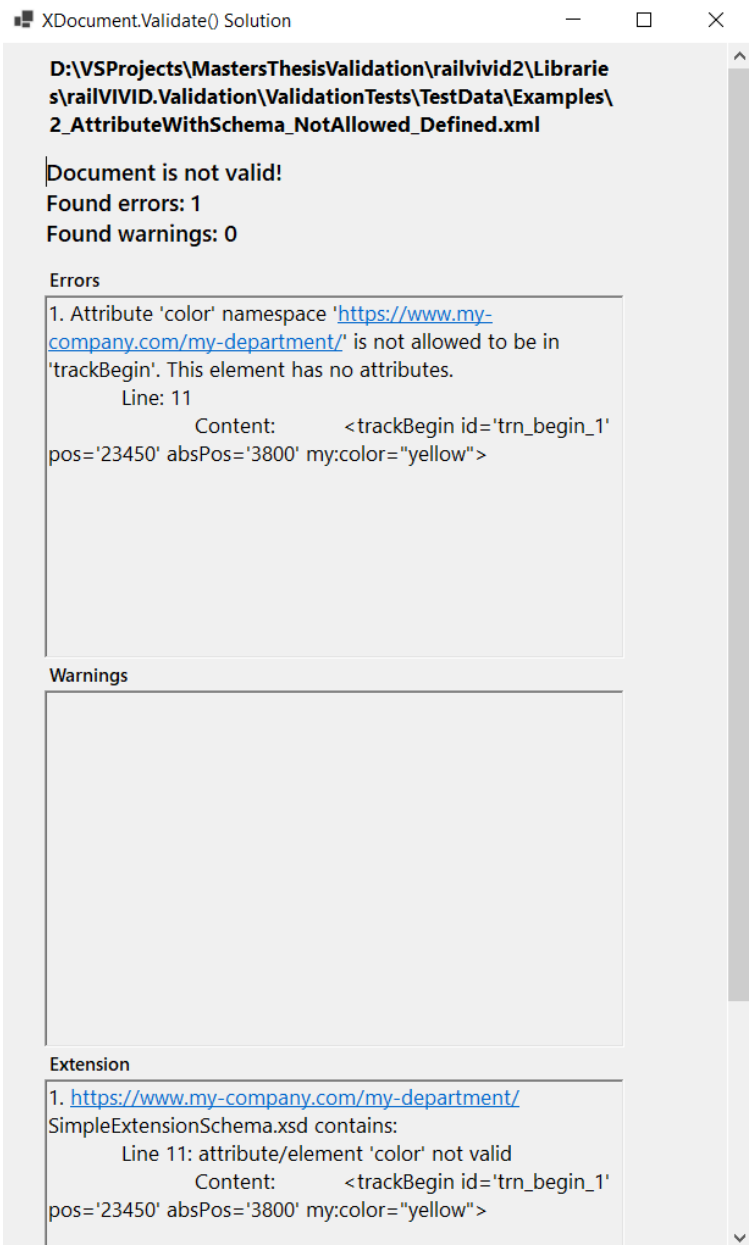


Рисунок 4.25 – Результати тестування файлу для ситуації №5 (зі схемою розширення) розробленим рішенням

Ситуація №6.

Валідація стандартними бібліотеками:

Валідація розробленим рішенням на основі XDocument.Validate():

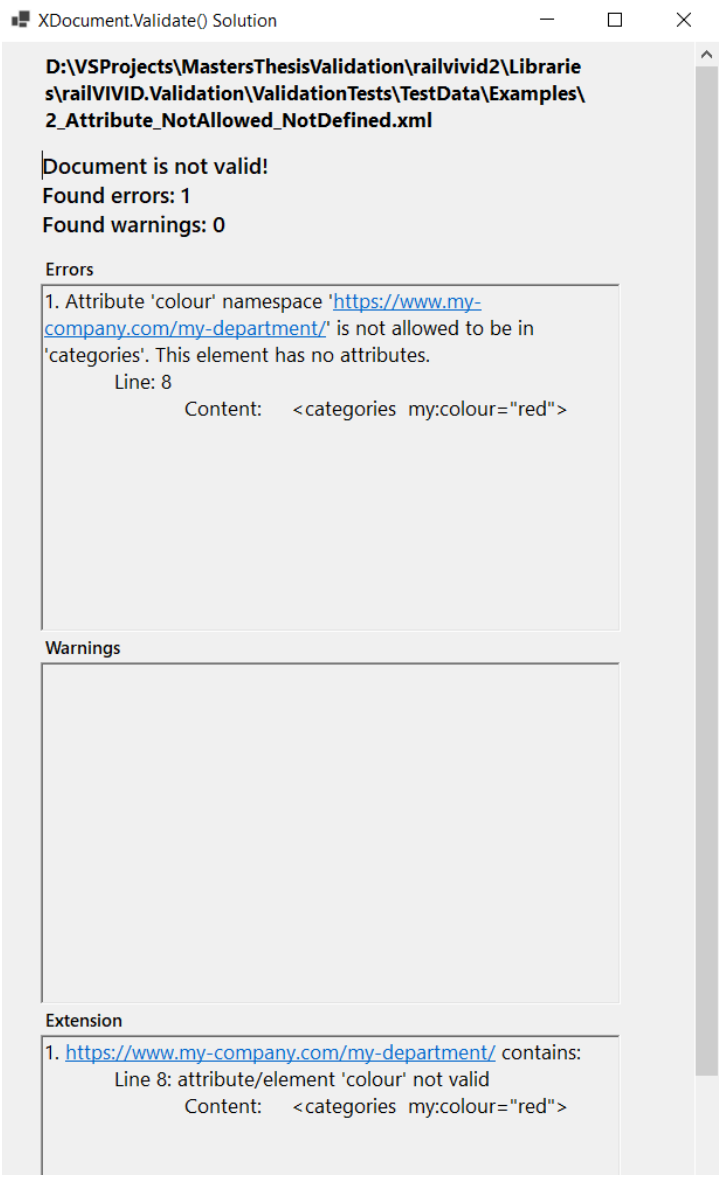


Рисунок 4.27 – Результати тестування файлу для ситуації №6 розробленим рішенням

Ситуація №6 (з схемою розширень).

Валідація стандартними бібліотеками:

Рисунок 4.28 – Результати тестування файлу для ситуації №6 (зі схемою розширення) трьома стандартними бібліотеками

Валідація розробленим рішенням на основі XDocument.Validate():

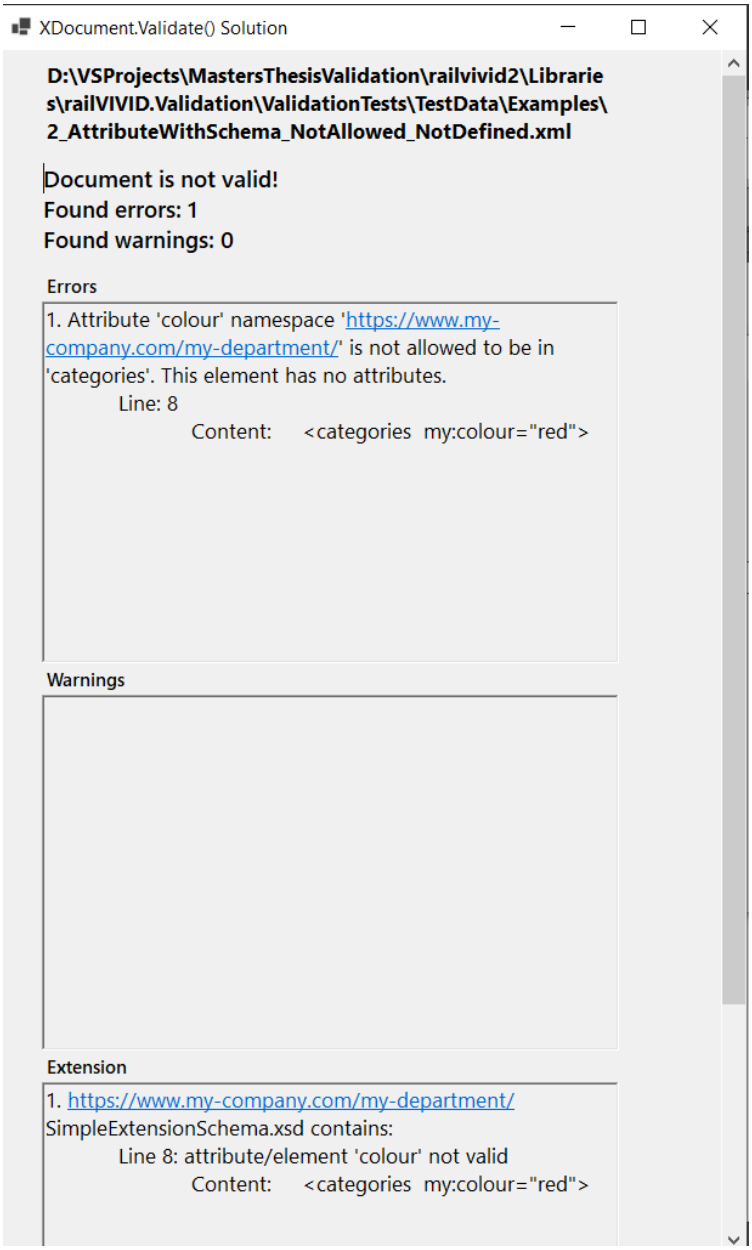


Рисунок 4.29 – Результати тестування файлу для ситуації №6 (зі схемою розширення) розробленим рішенням

Тобто як і при виконанні Unit-тестів результати між ситуацією №5 та №6 без схеми розширень та її присутністю не мають відмінності. Таким чином можна точно визначити, що проблема виникає через порушення правил схеми railML, що розроблене рішення має змогу виявити.

Ситуація №7.

Валідація стандартними бібліотеками:

XDocument:

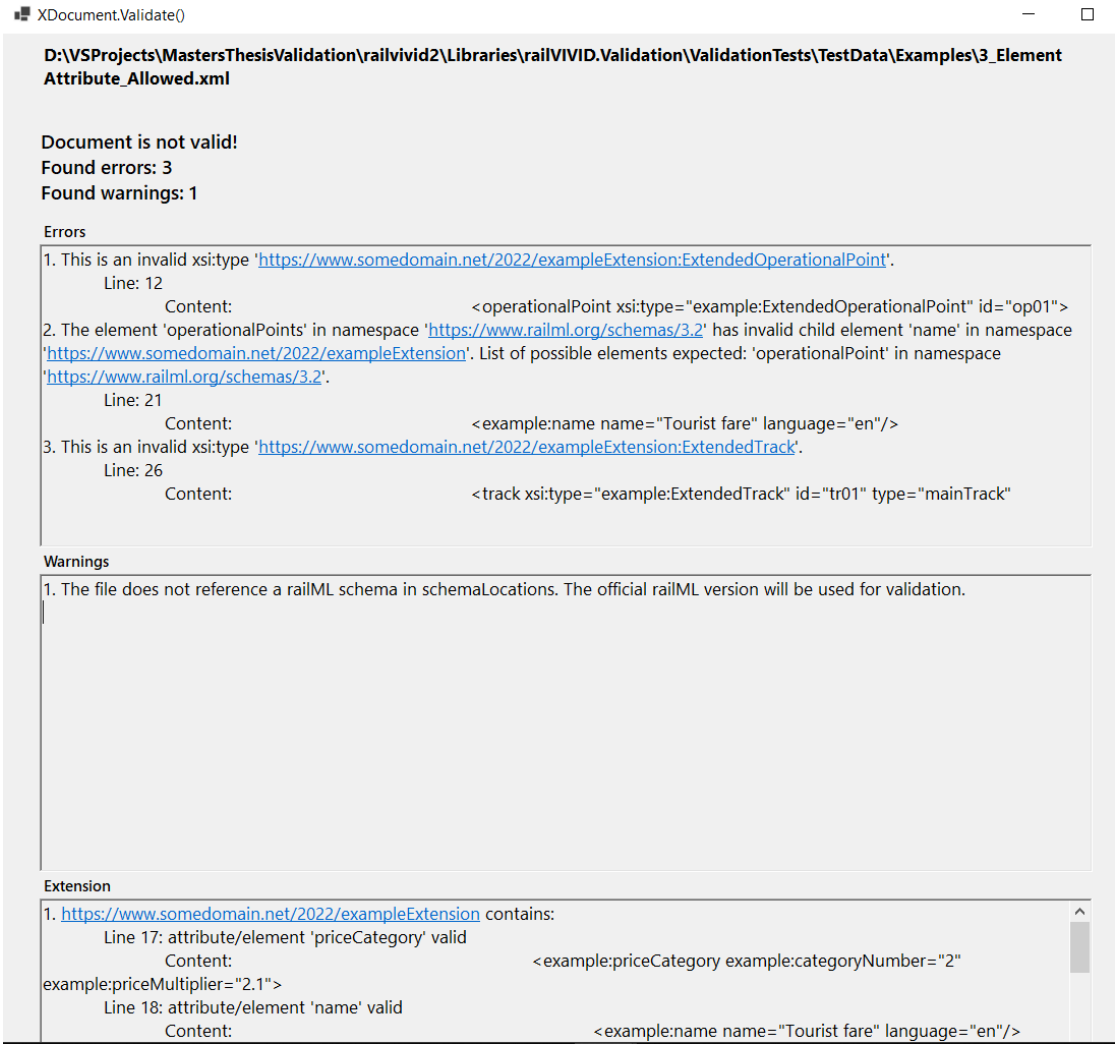


Рисунок 4.30 – Результати тестування файлу для ситуації №7 валідатором XDocument

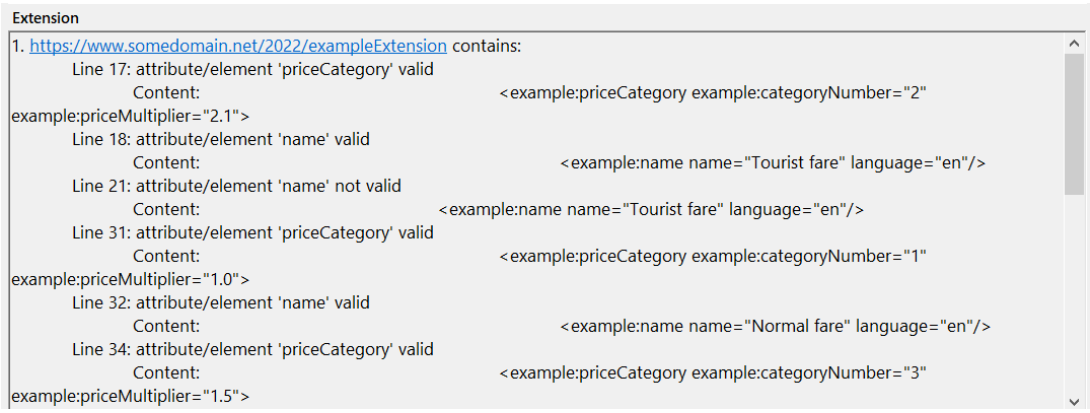


Рисунок 4.31 – Результати тестування файлу для ситуації №7 валідатором XDocument (деталі розширень №1)

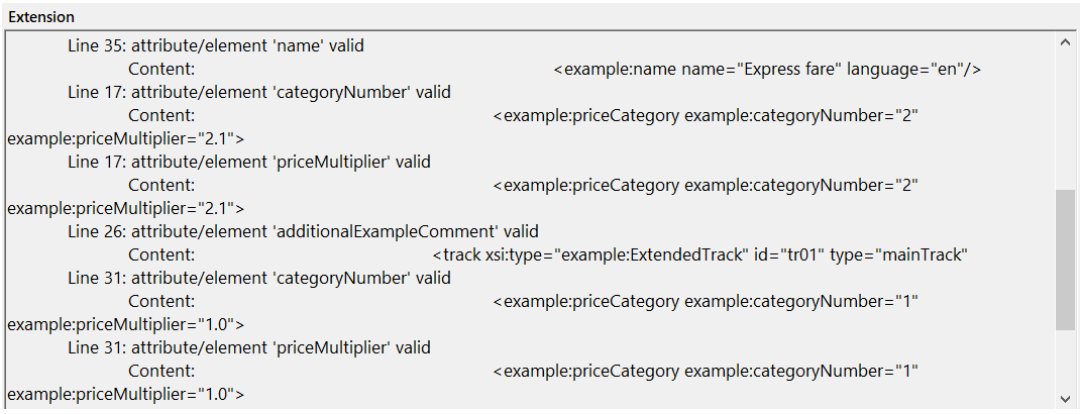


Рисунок 4.32 – Результати тестування файлу для ситуації №7 валідатором XDocument (деталі розширень №2)

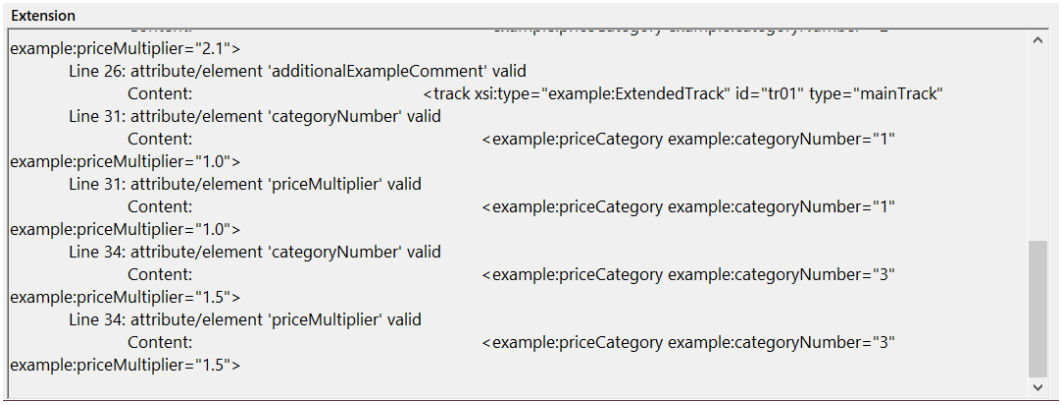


Рисунок 4.33 – Результати тестування файлу для ситуації №7 валідатором XDocument (деталі розширень №3)

XMLDocument:

Рисунок 4.34 – Результати тестування файлу для ситуації №7 валідатором XMLDocument

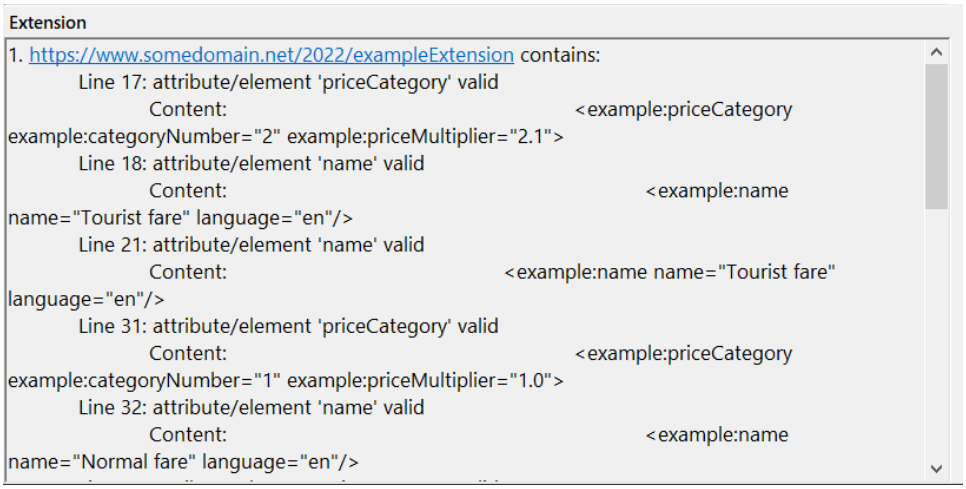


Рисунок 4.35 – Результати тестування файлу для ситуації №7 валідатором XMLDocument (деталі розширень №1)

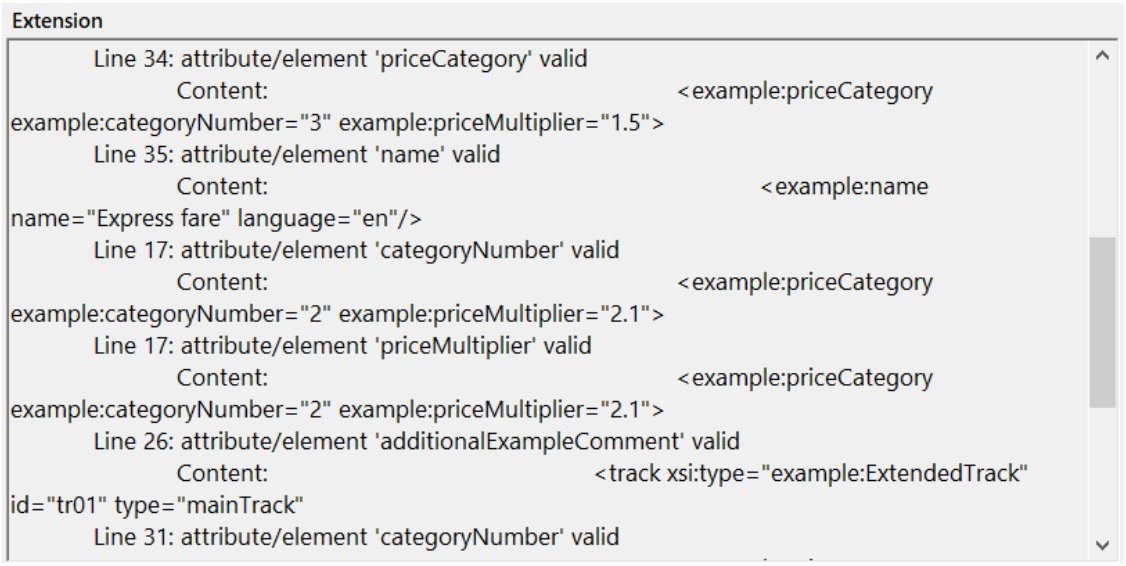


Рисунок 4.36 – Результати тестування файлу для ситуації №7 валідатором XMLDocument (деталі розширень №2)

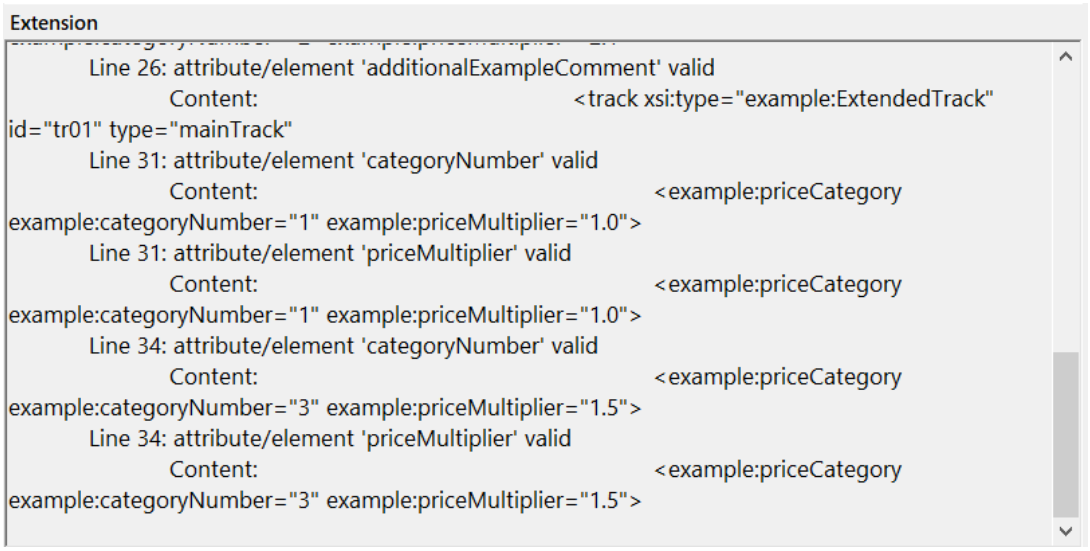


Рисунок 4.37 – Результати тестування файлу для ситуації №7 валідатором XMLDocument (деталі розширень №3)

XMLReader:

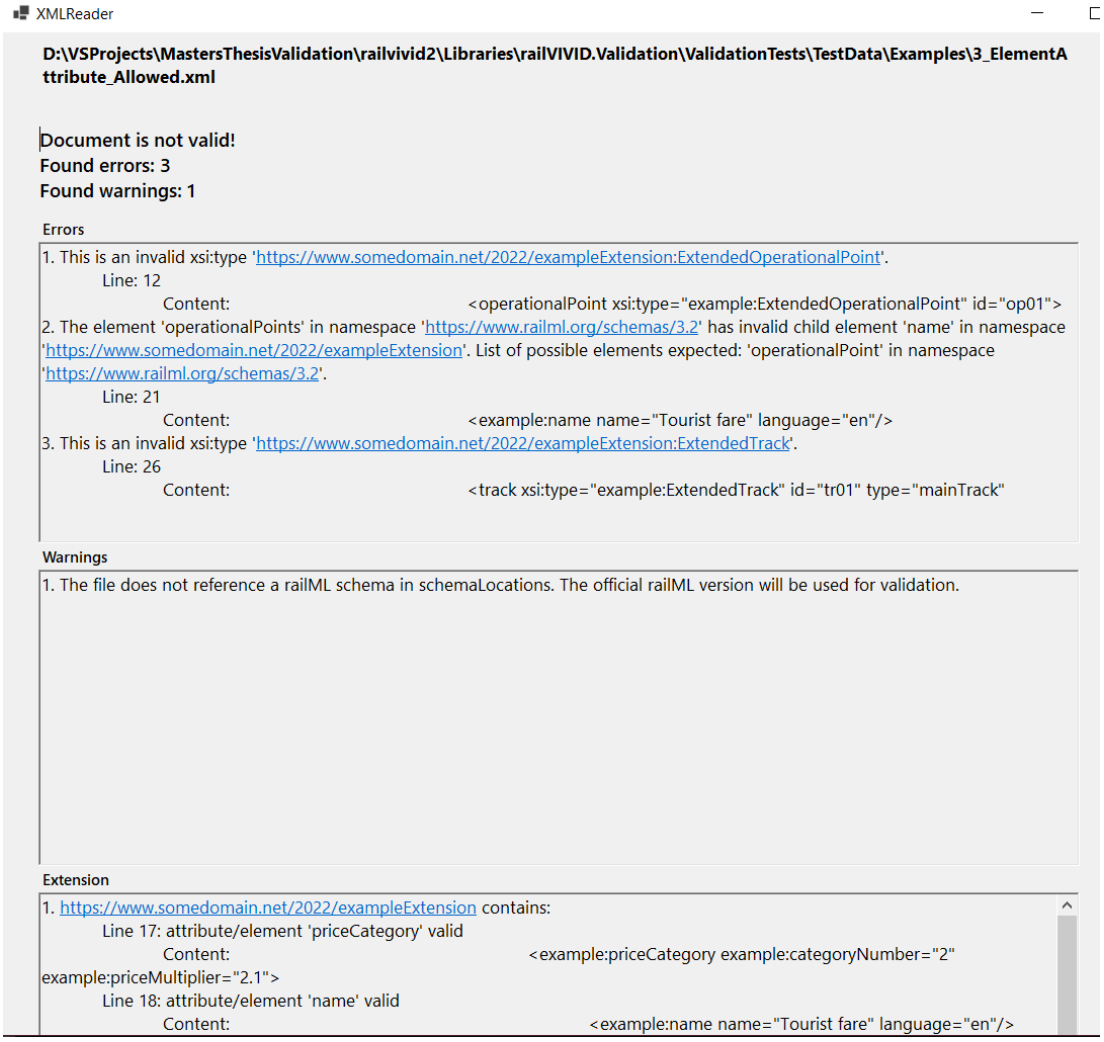


Рисунок 4.38 – Результати тестування файлу для ситуації №7 валідатором XMLReader

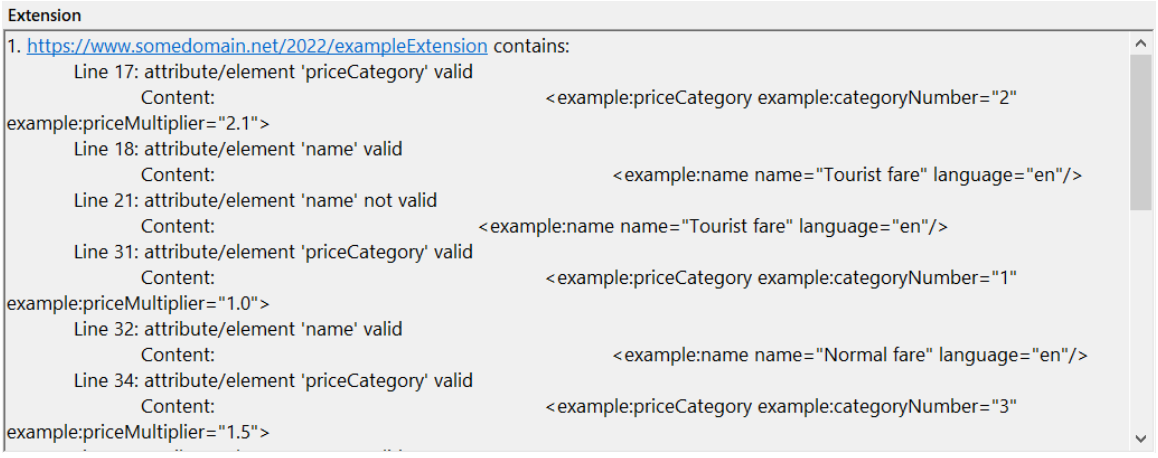


Рисунок 4.39 – Результати тестування файлу для ситуації №7 валідатором XMLReader (деталі розширень №1)

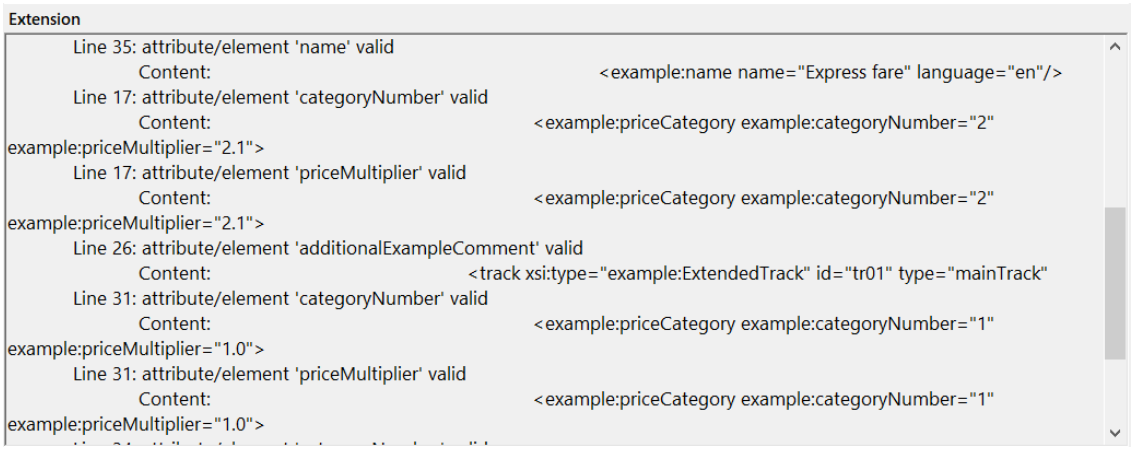


Рисунок 4.40 – Результати тестування файлу для ситуації №7 валідатором XMLReader (деталі розширень №2)

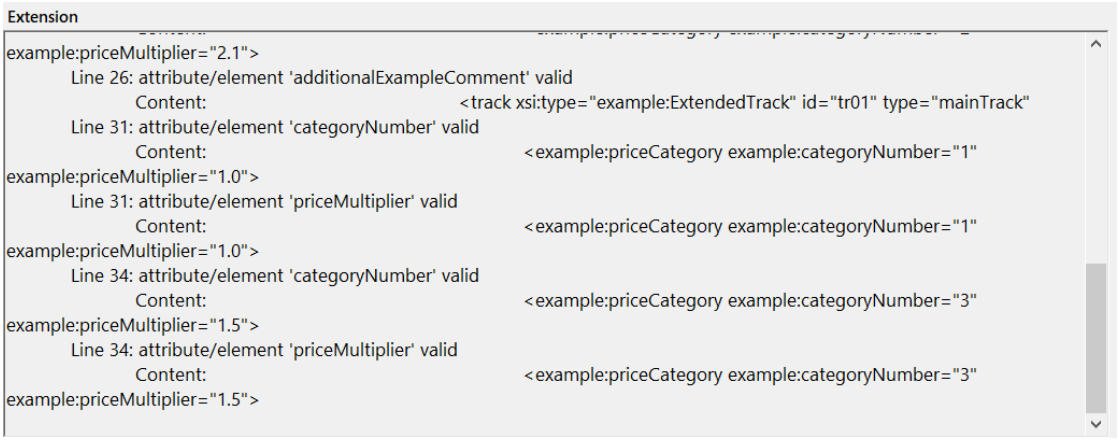


Рисунок 4.41 – Результати тестування файлу для ситуації №7 валідатором XMLReader (деталі розширень №3)

Валідація розробленим рішенням на основі XDocument.Validate():

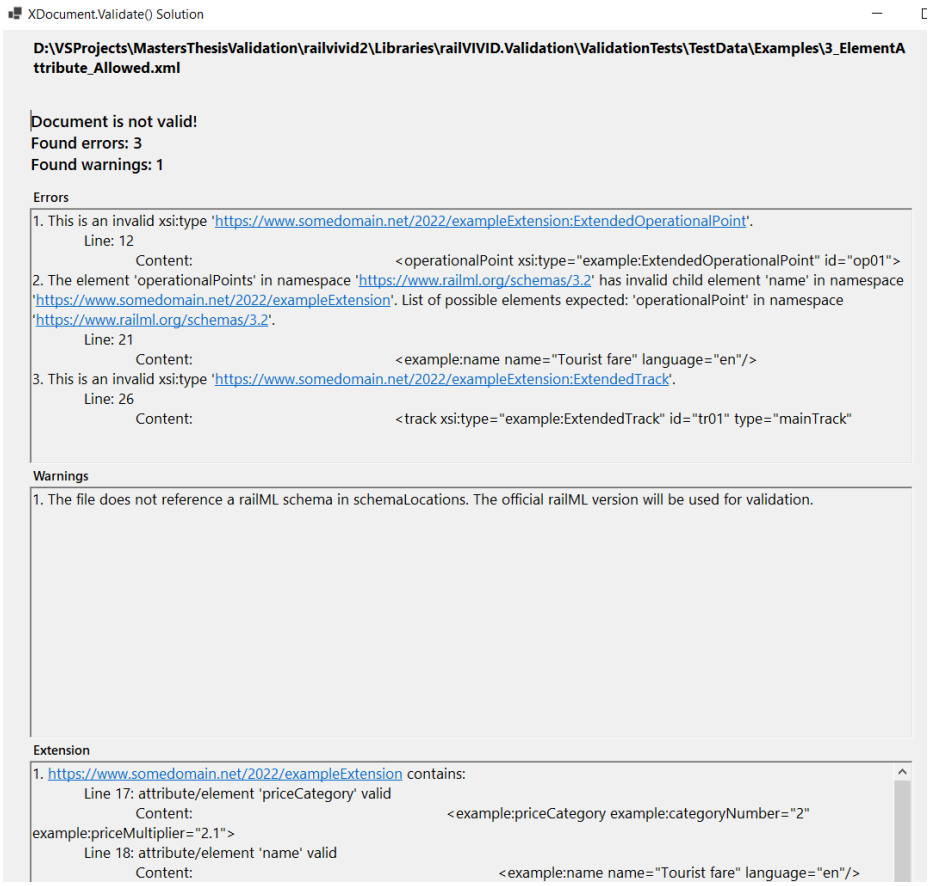


Рисунок 4.42 – Результати тестування файлу для ситуації №7 розробленим рішенням

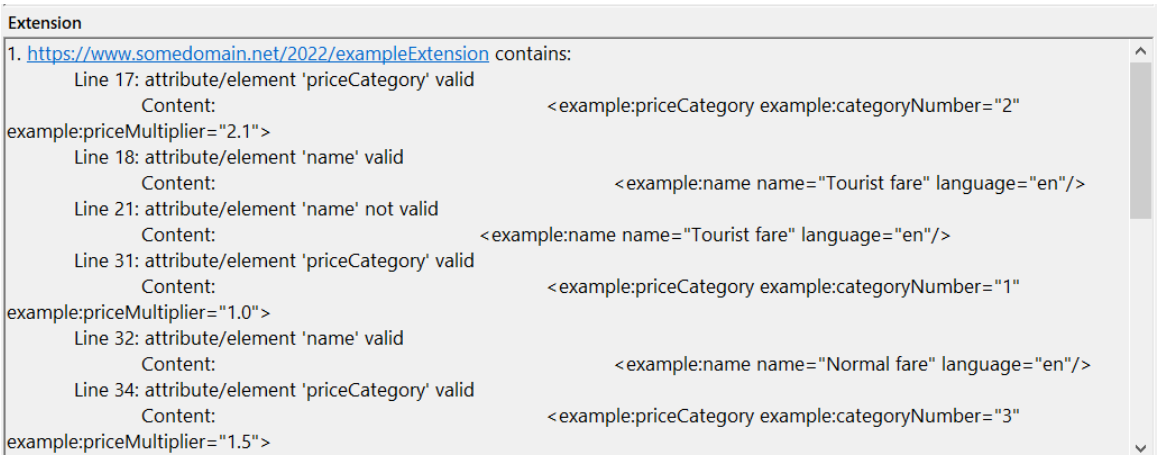
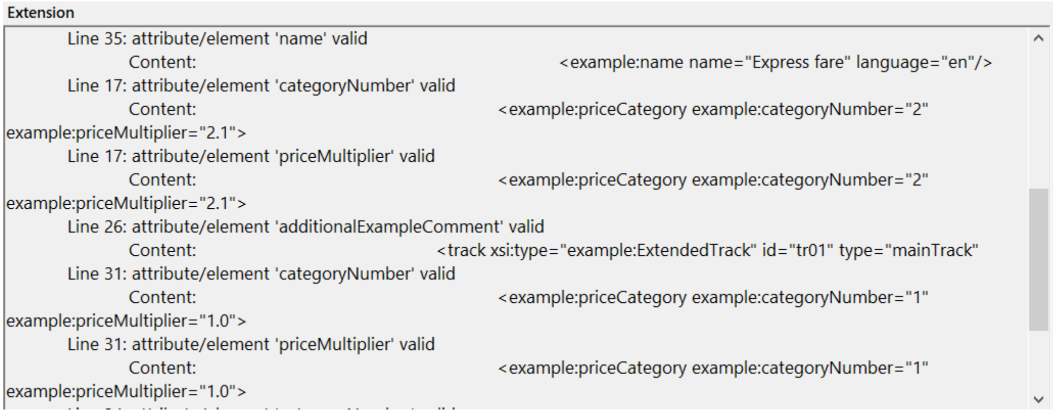
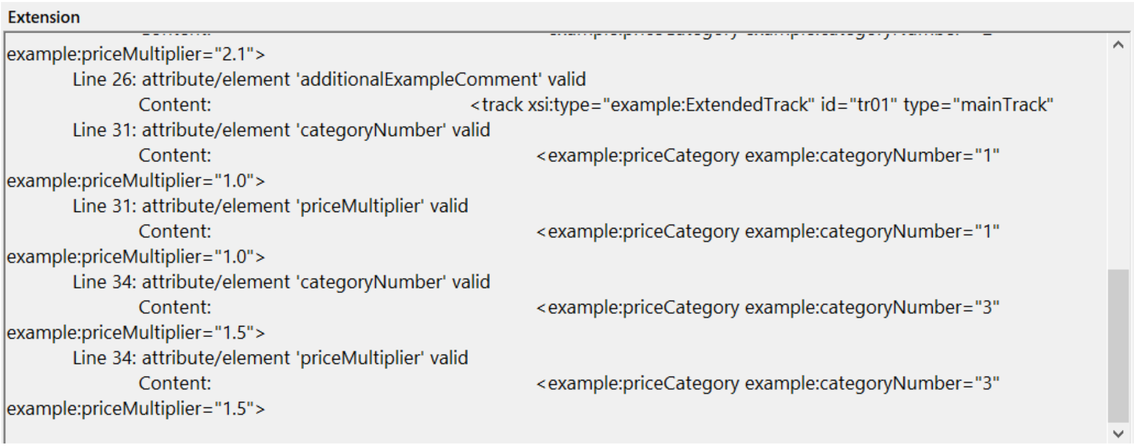


Рисунок 4.43 – Результати тестування файлу для ситуації №7 розробленим рішенням (деталі розширень №1)



```
Extension
        Line 35: attribute/element 'name' valid
                Content:                        <example:name name="Express fare" language="en"/>
        Line 17: attribute/element 'categoryNumber' valid
                Content:                        <example:priceCategory example:categoryNumber="2"
example:priceMultiplier="2.1">
        Line 17: attribute/element 'priceMultiplier' valid
                Content:                        <example:priceCategory example:categoryNumber="2"
example:priceMultiplier="2.1">
        Line 26: attribute/element 'additionalExampleComment' valid
                Content:                        <track xsi:type="example:ExtendedTrack" id="tr01" type="mainTrack"
        Line 31: attribute/element 'categoryNumber' valid
                Content:                        <example:priceCategory example:categoryNumber="1"
example:priceMultiplier="1.0">
        Line 31: attribute/element 'priceMultiplier' valid
                Content:                        <example:priceCategory example:categoryNumber="1"
example:priceMultiplier="1.0">
```

Рисунок 4.44 – Результати тестування файлу для ситуації №7 розробленим рішенням (деталі розширень №2)



```
Extension
example:priceMultiplier="2.1">
        Line 26: attribute/element 'additionalExampleComment' valid
                Content:                        <track xsi:type="example:ExtendedTrack" id="tr01" type="mainTrack"
        Line 31: attribute/element 'categoryNumber' valid
                Content:                        <example:priceCategory example:categoryNumber="1"
example:priceMultiplier="1.0">
        Line 31: attribute/element 'priceMultiplier' valid
                Content:                        <example:priceCategory example:categoryNumber="1"
example:priceMultiplier="1.0">
        Line 34: attribute/element 'categoryNumber' valid
                Content:                        <example:priceCategory example:categoryNumber="3"
example:priceMultiplier="1.5">
        Line 34: attribute/element 'priceMultiplier' valid
                Content:                        <example:priceCategory example:categoryNumber="3"
example:priceMultiplier="1.5">
```

Рисунок 4.45 – Результати тестування файлу для ситуації №7 розробленим рішенням (деталі розширень №3)

Як можна помітити, валідація з XMLDocument не надає інформації про рядок документу, на якому виникає проблема. Це виникає через те, що при завантаженні XMLDocument неможливо налаштувати збереження номерів рядків для документу, на відмінність від XDocument. Через це статус елементів розширень у результатах цього валідатору завжди мають статус «valid», так як для зміни статусу елементу потрібно знати його точне розташування, так як цей елемент може використовуватись в декількох місцях. У всіх інших аспектах стандартні бібліотеки функціонують однаково з однаковими текстами помилок та попереджень, окрім описаної вище ситуації №3.

ЗАГАЛЬНИЙ ВИСНОВОК

Під час виконання дипломної роботи було досліджено методи валідації документів railML в середовищі .NET. Для цього попередьо було визначено основні критичні ситуації специфічні для цього формату даних, правильна обробка яких є необхідною для коректної роботи розроблюваного валідатора.

В процесі роботи було розглянуто та проаналізовано функціонування доступних стандартних бібліотек для вирішення цієї задачі (валідатори класів XDocument, XMLDocument, XMLReader), під час чого було виявлено недолік у функціонуванні обраних стандартних методів.

Було визначено, що всі попередньо протестовані бібліотеки мають однакове нечітке повідомлення помилки «атрибут не є оголошеним» у трьох ситуаціях, які потребують різної обробки для правильної валідації файлів формату railML.

Знайдену проблему було досліджено, на основі чого було висунуто декілька гіпотез про її причину та способи її усунення, в результаті чого було визначено, що єдиним доцільним варіантом є розробка власного рішення. Таким чином можливо буде мати за основу стандартний метод XDocument.Validate(), який і надалі зможе бути оновленим у майбутньому, і використовувати власний алгоритм перевірки визначених проблемних ситуацій.

В результаті розроблене програмне забезпечення надає можливість валідації документів railML з урахуванням особливостей даного формату обміну даних. Таким чином було знайдено рішення виявленої проблеми валідації стандартних методів .NET для формату railML, що надало можливість використання розробленого в процесі цього програмного забезпечення для офіційного застосунку від railML.org – railVIVID 2.

Найкращою рекомендацією для більш оптимального рішення цієї проблемної ситуації для розробників перевірених стандартних методів є можлива зміна алгоритму визначення статусу атрибутів на інший, який надавав би більш чіткий статус за замовченням, що дозволить розрізнити виявлені атрибути без оголошення від недозволених. Таким чином робота валідатора для більш

стандартних форматів даних, як XML, залишилася б незмінною, однак інформація про помилку була б більш точною та детальною, що б сприяло розрізненню ситуацій для правильної валідації railML документів.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Contributors to Wikimedia projects. XML validation - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/XML_validation (дата звернення: 01.01.2025).

2. RailML – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/RailML (дата звернення: 01.01.2025).

3. XML для новачків - Підтримка від Microsoft. Microsoft Support. URL: https://support.microsoft.com/uk-ua/office/xml-для-новачків-a87d234d-4c2e-4409-9cbc-45e4eb857d44 (дата звернення: 01.01.2025).

4. Including or Importing XML Schemas - .NET. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/dotnet/standard/data/xml/including-or-importing-xml-schemas (дата звернення: 01.01.2025).

5. Dev:Extending railML - railML 3 Wiki. railML 3 Wiki. URL: https://wiki3.railml.org/wiki/Dev:Extending_railML#Namespace_handling (дата звернення: 02.01.2025).

6. railVIVID - railML.org (EN). Home - railML.org (EN). URL: https://www.railml.org/en/railvivid дата звернення: 02.01.2025).

7. 45th railML Conference - railML.org (EN). Home - railML.org (EN). URL: https://www.railml.org/en/event-reader/45th-railml-conference-virtual (дата звернення: 02.01.2025).

8. XML Editor. XML Editor, JSON Editor, Code Generator, Data Integration and Web Services Toolkit. URL: https://www.liquid-technologies.com/xml-editor (дата звернення: 02.01.2025).

9. XML Editor: XMLSpy. XML, Data Integration, and Mobile App Development Solutions by Altova | Altova. URL: https://www.altova.com/xmlspy-xml-editor (дата звернення: 02.01.2025).

10. Hyper-fast XML Validator. XML, Data Integration, and Mobile App Development Solutions by Altova | Altova. URL: https://www.altova.com/xmlspy-

xml-editor/xml-validator (дата звернення: 02.01.2025).

11. TT:category - railML 2 Wiki. railML 2 Wiki. URL: https://wiki2.railml.org/wiki/TT:category (дата звернення: 02.01.2025).

12. TT:categories - railML 2 Wiki. railML 2 Wiki. URL: https://wiki2.railml.org/wiki/TT:categories (дата звернення: 02.01.2025).

13. TT:additionalName category - railML 2 Wiki. railML 2 Wiki. URL: https://wiki2.railml.org/wiki/TT:additionalName_category (дата звернення: 02.01.2025).

14. TT:times ocpTT ocpsTT trainPart - railML 2 Wiki. railML 2 Wiki. URL: https://wiki2.railml.org/wiki/TT:times_ocpTT_ocpsTT_trainPart (дата звернення: 02.01.2025).

15. TT:trainPart - railML 2 Wiki. railML 2 Wiki. URL: https://wiki2.railml.org/wiki/TT:trainPart (дата звернення: 02.01.2025).

16. IS:trackBegin - railML 2 Wiki. railML 2 Wiki. URL: https://wiki2.railml.org/wiki/IS:trackBegin (дата звернення: 02.01.2025).

17. IS:operationalPoint - railML 3 Wiki. railML 3 Wiki. URL: https://wiki3.railml.org/wiki/IS:operationalPoint (дата звернення: 02.01.2025).

18. IS:track - railML 3 Wiki. railML 3 Wiki. URL: https://wiki3.railml.org/wiki/IS:track (дата звернення: 02.01.2025).

19. IKVM. IKVM · IKVM. URL: https://ikvm.org/ (дата звернення: 02.01.2025).

20. .NET XML validator doesn't give accurate error message for prohibited attributes. Stack Overflow. URL: https://stackoverflow.com/questions/78953257/net-xml-validator-doesnt-give-accurate-error-message-for-prohibited-attributes (дата звернення: 02.01.2025).

21. Wijendra D. R., Hewagamage K. P. Cognitive Complexity Beyond Generalization: A Subjective Rating for the Human Comprehension. ITM Web of Conferences. 2022. T. 50. C. 01003. URL: https://doi.org/10.1051/itmconf/20225001003 (дата звернення: 03.01.2025).

22. XDocument Class (System.Xml.Linq). Microsoft Learn: Build skills

that open doors in your career. URL: https://learn.microsoft.com/en-us/dotnet/api/system.xml.linq.xdocument?view=net-8.0 (дата звернення: 02.01.2025).

23. XmlDocument Class (System.Xml). Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/dotnet/api/system.xml.xmldocument?view=net-8.0 (дата звернення: 02.01.2025).

24. XmlReader Class (System.Xml). Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/dotnet/api/system.xml.xmlreader?view=net-9.0 (дата звернення: 02.01.2025).

25. Що таке об’єктно-орієнтоване програмування: принципи, переваги та недоліки. GoIT – платформа IT-курсів. URL: https://goit.global/ua/articles/shcho-take-ob-iektno-oriientovane-prohramuvannia-pryntsypy-perevahy-ta-nedoliky/ (дата звернення: 03.01.2025).

26. Dev:fileConventions - railML 2 Wiki. railML 2 Wiki. URL: https://wiki2.railml.org/wiki/Dev:fileConventions (дата звернення: 02.01.2025).

27. 46th railML Conference - railML.org (EN). Home - railML.org (EN). URL: https://www.railml.org/en/event-reader/46th-railml-conference-prague (дата звернення: 03.01.2025).

ДОДАТОК А

Технічне завдання

ЗАТВЕРДЖУЮ
Проректор Українського державного
університету науки і технологій
Анатолій РАДКЕВИЧ

«ДОСЛІДЖЕННЯ МЕТОДІВ ВАЛІДАЦІЇ RAILML ДОКУМЕНТІВ В СЕРЕДОВИЩІ .NET»
Технічне завдання
44165850.1451-01

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Олександр ІВАНОВ
Виконавець
_____Маргарита ВИСКАРКА
Нормоконтролер
_____Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО
44165850.1451-01

«ДОСЛІДЖЕННЯ МЕТОДІВ ВАЛІДАЦІЇ RAILML ДОКУМЕНТІВ В СЕРЕДОВИЩІ .NET»

Технічне завдання

44165850.1451-01

Листів 12

2025

ЗМІСТ

ВСТУП

Сучасні залізничні системи постійно розвиваються та ускладняються, через що зростає потреба у обміні даних між різними частинами інфраструктури. Щоб забезпечити такий обмін було створено railML (Railway Markup Language) – відкритий формат обміну даними на основі XML, що використовується для сумісності даних залізничних застосунків. Однак як і з файлами XML, для перевірки коректності структури railML документів потребується їх валідація, яка має враховувати особливості даного формату обміну даних.

.NET платформа широко використовується для обробки та валідації XML-документів, зокрема таких складних форматів, як railML. Однак під час валідації цих документів виникають труднощі, пов'язані з некоректними або недостатньо інформативними повідомленнями про помилки, що може ускладнювати пошук і виправлення помилок у файлах.

Це дослідження спрямоване на аналіз існуючих методів валідації railML документів у середовищі .NET, а також розробку рішень для покращення якості повідомлень про помилки та підвищення точності валідації.

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ від 26.12.2023 №1187ст ректора Українського державного університету науки і технологій “Про затвердження тем та призначення керівників дипломних проектів” за спеціальністю 121 “Інженерія програмного забезпечення» факультету “Комп’ютерних технологій і систем” по кафедрі “Комп’ютерні інформаційні технології”.

Тема дипломної роботи – “Дослідження методів валідації railML документів в середовищі .NET”.

Керівник дипломного проекту: Іванов О. П.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Ця розробка спрямована на проведення аналізу та порівняння різних методів валідації railML документів у середовищі .NET. Метою цієї роботи має бути перевірка відповідності потестованих бібліотек та рішень до вимог railML.org за допомогою перевірки їх результатів для валідації конкретних критичних ситуацій. Шляхом визначення переваг, недоліків та особливостей обраних бібліотек та розробленого рішення можливо буде визначити найкращий метод валідації railML файлів.

2.1 Функціональне призначення розробки

Основні аспекти функціонального призначення розробки:

- проведення аналізу функціональності – основною функцією розробки є дослідження стандартних методів валідації у середовищі .NET;
- порівняння функцій – розробка дозволить порівняти бібліотеки, визначити відповідність їх результатів до вимог railML.org;
- розробка і перевірка рішення проблеми стандартних функцій – розробка також має за мету створення рішення до проблеми не чітких результатів стандартних бібліотек у .NET та порівняння результатів доповненого рішення з очікуваними.

2.2 Експлуатаційне призначення розробки:

Основні аспекти експлуатаційного призначення розробки:

– визначення найкращого рішення до поставленої проблеми валідації railML документів у середовищі .NET;

– створення ряду бібліотек мовою C#, які можливо буде використати повторно при створенні офіційного програмного забезпечення для railML.org.

Загальне експлуатаційне призначення розробки полягає в практичній розробці рішення до існуючих проблем валідації у .NET, яке можливо буде використати при розробці програмного забезпечення для railML.org.

3 ВИМОГИ ДО ПРОГРАМИ

3.1 Вимоги до функціональних характеристик

Програмний продукт має надавати користувачу можливість порівняти результати валідації railML документів за допомогою різних стандартних методів середовища .NET та розробленого рішення на основі одного з обраних методів.

Вхідні дані:

- шлях до файлу, який необхідно перевірити;
- обраний метод валідації.

Вихідні дані:

- загальний результат валідації з визначенням чи є перевірений документ правильним чи ні;
- інформація про знайдені помилки;
- інформація про знайдені попередження;
- інформація про розширення.

3.2 Вимоги до надійності

Вимоги до надійності: контроль вихідної і вхідної інформації.

3.3 Вимоги експлуатації

Працювати з програмою може людина, що має навички роботи з десктопними пристроями та ознайомлена з керівництвом користувача програмного продукту.

3.4 Вимоги до складу та параметрів технічних засобів

Склад технічних засобів для використання даного продукту: процесор з тактовою частотою 2 ГГЦ або вище; 2 Гб. оперативної пам’яті; клавіатура.

3.5 Вимоги до інформаційної та програмної сумісності

Програма має функціонувати в середовищі ОС Windows 10\11, в якому має бути встановлений .NET 6.0 або вище.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу документації мають входити:

– текст програми;

– керівництво користувача для користування настільним застосунком.

Вся документація програмних додатків повинна задовольняти вимоги до програмної документації.

5 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Таблиця 5.1 – Стадії та етапи розробки

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	04.09.24 –15.09.24
Робочий проект	Програмування та відлагодження програми.	16.09.24 –15.10.24
	Тестування програми	16.10.24 – 31.10.24
	Розробка, узгодження і затвердження програмної документації.	01.11.24 – 30.11.24

6 ПОРЯДОК ПРИЙНЯТТЯ

Контроль за виконанням роботи здійснює керівник розробки доц. Іванов О. П.

7 ТЕХНІЧНО-ЕКОНОМІЧНІ ПОКАЗНИКИ

Показники та їх розрахунок не були описані у документах, так як розробка програмного продукту несе в собі навчальний характер, а не комерційний.

ДОДАТОК Б

Технічне завдання

ЗАТВЕРДЖУЮ
Проректор Українського державного
університету науки і технологій
Анатолій РАДКЕВИЧ

«ДОСЛІДЖЕННЯ МЕТОДІВ ВАЛІДАЦІЇ RAILML ДОКУМЕНТІВ В СЕРЕДОВИЩІ .NET»
Текст програми
44165850.01451–01 12–01

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Олександр ІВАНОВ
Виконавець
_____ Маргарита ВИСКАРКА
Нормоконтролер
_____Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО

44165850.1451–01 12–01

ДОСЛІДЖЕННЯ МЕТОДІВ ВАЛІДАЦІЇ RAILML ДОКУМЕНТІВ В СЕРЕДОВИЩІ .NET

Текст програми
44165850.1451–01 12–01
Листів 52

2025

АНОТАЦІЯ

Документ 44165850.1451–01 12 01 «Дослідження методів валідації railML документів в середовищі .NET» входить до складу програмної документації на програму, що дозволяє порівняння результатів валідації документів railML.

У даному документі представлений текст додатку. Додаток написаний на мові C# з використанням стандартних бібліотек валідації у .NET у програмному середовищі Visual Studio 2022.

ЗМІСТ

1 ТЕКСТ ПРОГРАМИ

1.1 Текст файлу IValidator.cs модуля railVIVID.Validation

```
using railVIVID.Validation.Extensions;
using railVIVID.Validation.GeneralInformation;
using System;

namespace railVIVID.Validation
{
   public interface IValidator
   {
      public event Action<int> ValidationProgressChanged;
      public Tuple<ValidationResult, railMLExtensions>
Validate(railMLDocument document, railMLVersion
railMLVersion);
   }
}
```

1.2 Текст файлу Validation.cs модуля railVIVID.Validation

```
using railVIVID.Validation.Extensions;
using System.Text.RegularExpressions;
using System.Xml;
using System.Xml.Schema;

namespace railVIVID.Validation
{
   public class Validation
   {
      public string ExtractNamespaceFromErrorMessage(string
message)
      {
         string pattern1 = @"(http://[^\s:]+):";
         string pattern2 = @"(https://[^\s:]+):";

         Match match = Regex.Match(message, pattern1);
         if (match.Success)
         {
            return match.Groups[1].Value;
         }
         match = Regex.Match(message, pattern2);
         if (match.Success)
         {
            return match.Groups[1].Value;
         }
         return null;
      }

      public string ExtractObjectNameFromErrorMessage(string
message)
      {
         string pattern1 = @":([^:]+)'";
         string pattern2 = @"'(.+?)'";
         string pattern3 = @":([^:]+)(?=\s|$)";

         Match match = Regex.Match(message, pattern1);
         if (match.Success)
         {
            return match.Groups[1].Value;
         }
         match = Regex.Match(message, pattern2);
         if (match.Success)
         {
            return match.Groups[1].Value;
         }
         match = Regex.Match(message, pattern3);
         if (match.Success)
         {
            return match.Groups[1].Value;
         }
         return null;
      }

      public string ExtractNamespaceFromType1(string message)
      {
         string pattern = @"has invalid child element '.*?' in
namespace '(https?://[^\s']+)'";
         Match match = Regex.Match(message, pattern);

         return match.Success ? match.Groups[1].Value : null;
      }

      public string ExtractElementNameFromType1(string message)
      {
         string pattern = @"has invalid child element '(.*?)'";
         Match match = Regex.Match(message, pattern);

         return match.Success ? match.Groups[1].Value : null;
      }

      public string ExtractNamespaceElementPairFromType2(string
message)
      {
         string pattern = @"cannot contain child element
'(https?://[^\s']+)'";
         Match match = Regex.Match(message, pattern);

         return match.Success ? match.Groups[1].Value : null;
      }

      //Used by standard methods
      //Processes issues raised by validator (requires special handling
of non-railml related issues)
      public void HandleIssues(ValidationEventArgs e,
ValidationResult result, railMLExtensions extensions)
      {
         string namespaceUri = string.Empty;
         string objectName = string.Empty;

         if (e.Severity == XmlSeverityType.Warning)
         {
            result.AddWarning(e.Exception.LineNumber,
e.Exception.Message);
         }
         else
         {
            //All railML elements are declared by default use of
official schema,
            //so issue regarding declaration are considered warnings
because they can only be caused by extensions
            //But in reality if the issue is undeclared element, it can
appear in three cases:
            //1. Element/attribute is missing a schema
            //2. Element/attribute is missing a declaration
```

```
            //3. Unknown attribute is used within element without
<xsi:any> or <xsi:anyAttribute> (Microsoft .NET issue)
            if (e.Exception.Message.Contains("cannot find the
declaration") ||
            e.Exception.Message.Contains("attribute is not declared") ||
            e.Exception.Message.Contains("element is not declared"))
            {
               namespaceUri =
ExtractNamespaceFromErrorMessage(e.Message);
               objectName =
ExtractObjectNameFromErrorMessage(e.Message);
               result.AddWarning(e.Exception.LineNumber,
e.Exception.Message)
            }
            else
            {
               result.AddError(e.Exception.LineNumber, e.Message);
            }
         }

         if (namespaceUri == string.Empty || objectName ==
string.Empty)
         {
            namespaceUri =
ExtractNamespaceFromType1(e.Message);
            objectName =
ExtractElementNameFromType1(e.Message);

            if(!extensions.ChangeElementValidStatus(namespaceUri,
objectName, e.Exception.LineNumber))
            {
               string namespaceElement =
ExtractNamespaceElementPairFromType2(e.Message);
               if(namespaceElement != null) {
                  namespaceUri =
ExtractNamespaceFromErrorMessage(namespaceElement);
                  objectName =
ExtractObjectNameFromErrorMessage(namespaceElement);
                  extensions.ChangeElementValidStatus(namespaceUri,
objectName, e.Exception.LineNumber);
               }

            }
         }
         else
         {
            extensions.ChangeElementValidStatus(namespaceUri,
objectName, e.Exception.LineNumber);
         }

      }

      public void AddRailML(XmlSchemaSet schemaSet,
SchemaControl schemaControl, railMLVersion railMLVersion)
      {
         //If there is no access to remote railML schemas
         if (SchemaControl.DefaultRailMLLocation == "local")
         {
            //Needs direct adding of railML schema as embedded
resource because non-imported local schemas need to be added
directly
            //(XmlReader.Create will have issues otherwise). Imported
schemas are resolved as embedded resources in custom resolver.
            string version = railMLVersion.VersionFromNamespace;
            if (string.Equals(version, "3.1") || string.Equals(version,
"3.2"))
            {
               schemaControl.LoadRailMLFromResource(schemaSet,
"railml3.xsd", version);
            }
            else
            {
               schemaControl.LoadRailMLFromResource(schemaSet,
"railML.xsd", version);
            }
         }
         else
         {
            using (var reader =
XmlReader.Create(SchemaControl.DefaultRailMLLocation))
            {
               schemaSet.Add(null, reader);
            }
         }
      }

   }
}
```

1.3 Текст файлу CustromXmlUrlResolver.cs модуля railVIVID.Validation

```
using System;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Xml;
using System.Xml.Linq;

namespace railVIVID.Validation
{
   /// <summary>
   /// Custom resolver to be able to handle internal exceptions and
embedded resources
   /// </summary>
   public class CustomXmlUrlResolver : XmlUrlResolver
   {
      private ValidationResult _validationResult;
      private railMLVersion _railMLVersion;
      public bool _railMLAdded { private get; set; }
      private string lastAbsoluteURI;

      public CustomXmlUrlResolver(ValidationResult
validationResult, railMLVersion railMLVersion)
      {
         _validationResult = validationResult;
         _railMLAdded = false;
         _railMLVersion = railMLVersion;
      }

      public override object GetEntity(Uri absoluteUri, string role,
Type ofObjectToReturn)
      {
         try
         {
            //Always resolve mathML as embedded schemas due to
possible denial of access (hard to track)
            if (absoluteUri.ToString().Contains("mathml"))
            {
               Stream schemaStream =
GetMathMLSchemaStream(absoluteUri);
               if (schemaStream != null)
               {
                  return schemaStream;
               }
            }
```

```
            try
            {
               if (_railMLAdded)
               {
                  //Using LINQ and XDocument to get target
namespace of schema
                  XDocument schemaDoc =
XDocument.Load(absoluteUri.ToString());
                  var targetNamespace =
schemaDoc.Root?.Attribute("targetNamespace")?.Value;

                  //If it's railML namespace, don't import it, because
official version was already added
                  if
(targetNamespace.Contains("http://www.railml.org/schemas/") ||
targetNamespace.Contains("https://www.railml.org/schemas/"))
                  {
                     _validationResult.AddWarning(-1, $"Schema
{lastAbsoluteURI} imports railML, the official version will be
used");
                     return null;

                  }
               }
            }
            catch (Exception e)
            {
               _validationResult.AddWarning(-1, $"Problem with
loading of {absoluteUri} schema. It will not be used during
validation. " +
                  $"Details: {e.Message}");
               return null;
            }
            lastAbsoluteURI = absoluteUri.ToString();
            return HandleSchema(absoluteUri, role,
ofObjectToReturn);

         }
         catch (Exception ex)
         {

            if (ex.Message.Contains("Response status code does not
indicate success: 429 (Too Many Requests)"))
            {
               _validationResult.AddWarning(-1, $"HTTP Error:
{ex.Message} Related with schema: {absoluteUri}. " +
                  $"This issue typically occurs due to overload of
www.w3.org with requests. Please try at another time.");
            }
            else
            {
               _validationResult.AddError(-1, $"Error retrieving
schema: {ex.Message} Related with schema: {absoluteUri}.");
            }


            return null;
         }
      }

      private object HandleSchema(Uri absoluteUri, string role, Type
ofObjectToReturn)
      {
         //If there is no Internet connection - resolve main embedded
schemas
         if (SchemaControl.DefaultRailMLLocation == "local")
         {
            Stream schemaStream =
GetEmbeddedSchemaStream(absoluteUri);
            if (schemaStream != null)
            {
               return schemaStream;
            }
         }
         //Otherwise resolve as regular
         return base.GetEntity(absoluteUri, role, ofObjectToReturn);
      }

      private Stream GetMathMLSchemaStream(Uri absoluteUri)
      {
         Assembly assembly =
Assembly.Load("railVIVID.Validation");

         string relativePath = absoluteUri.LocalPath;
         string[] pathParts = relativePath.Split(new[] { '/', '\\' },
StringSplitOptions.RemoveEmptyEntries);

         string schemaFileName = pathParts.Last();

         string folderName = pathParts[pathParts.Length - 2];

         string resourceBaseName =
$"railVIVID.Validation.Resources.mathML";

         string resourcePath;
         if (folderName == "mathml2")
         {
            resourcePath =
$"{resourceBaseName}.{schemaFileName}";
         }
         else
         {
            resourcePath =
$"{resourceBaseName}.{folderName}.{schemaFileName}";
         }

         Stream stream =
assembly.GetManifestResourceStream(resourcePath);
         if (stream != null)
         {
            return stream;
         }

         return null;
      }


      private Stream GetEmbeddedSchemaStream(Uri absoluteUri)
      {
         Assembly assembly =
Assembly.Load("railVIVID.Validation");
         string[] folders = { "railML", "XML" };

         foreach (var folder in folders)
         {
            string embeddedSchemaName =
GetEmbeddedSchemaName(absoluteUri, folder);
            Stream stream =
assembly.GetManifestResourceStream(embeddedSchemaName);

            if (stream != null)
            {
               return stream;
            }
         }

         return null;
      }

      //Finds embedded resource based on the name and folder
      private string GetEmbeddedSchemaName(Uri schemaUri, string
folderName)
      {
         string schemaFileName =
Path.GetFileName(schemaUri.LocalPath);
         string formattedVersion =
SchemaControl.FormatVersionForResource(_railMLVersion.Version
FromNamespace);
```

```
        if (String.Equals(folderName, "railML"))
        {
            return
$"railVIVID.Validation.Resources.{folderName}._{formattedVersio
n}.{schemaFileName}";
        }
        else
        {
            return
$"railVIVID.Validation.Resources.{folderName}.{schemaFileName
}";
            }

        }

    }


}
```

1.4 Текст файлу Prevalidation.cs модуля railVIVID.Validation

```
using System;
using System.IO;
using System.Xml;
using System.Linq;
using System.Xml.Linq;
using railVIVID.Validation.GeneralInformation;
using System.IO.Compression;


namespace railVIVID.Validation
{
    /// <summary>
    /// Class for checking if document is well-formed and basic railML
requirements
    /// </summary>
    public class Prevalidation
    {
        private XDocument LoadRailML(string filePath)
        {
            string extension = Path.GetExtension(filePath);
            if (String.Equals(extension, ".railmlx",
StringComparison.OrdinalIgnoreCase))
            {
                return LoadRailMLXFile(filePath);
            }
            else if(String.Equals(extension, ".railml",
StringComparison.OrdinalIgnoreCase) || String.Equals(extension,
".xml", StringComparison.OrdinalIgnoreCase))
            {
                return LoadFile(XmlReader.Create(filePath));
            }
            else
            {
                throw new InvalidDataException("Unsupported file
format.");
            }

        }

        private XDocument LoadRailMLXFile(string filePath)
        {
            using (var archive = ZipFile.OpenRead(filePath))
            {
                //Look for files with .xml or .railml extension
                string zipName =
Path.GetFileNameWithoutExtension(filePath);

                var railMLFile = archive.Entries
                                .FirstOrDefault(e =>
e.FullName.EndsWith(".xml",
StringComparison.OrdinalIgnoreCase) ||

e.FullName.EndsWith(".railml",
StringComparison.OrdinalIgnoreCase)
                                                &&
e.FullName.Remove(zipName.Length) == zipName);

                if (railMLFile == null)
                {
                    throw new FileNotFoundException("No valid railML
file found in the archive.");
                }

                return LoadFile(XmlReader.Create(railMLFile.Open()));
            }
        }

        private XDocument LoadFile(XmlReader reader)
        {
            using (reader)
            {
                return XDocument.Load(reader,
LoadOptions.SetLineInfo);
            }
        }

        private string[] LoadLines(string filePath)
        {
            return File.ReadAllLines(filePath);
        }

        private void SetDirectory(string path)
        {

Directory.SetCurrentDirectory(Path.GetDirectoryName(path));
        }

        /// <summary>
        /// Prevalidates file (checks presence of railML namespace,
compares versions, creates XDocument)
        /// </summary>
        /// <param name="filePath"></param>
        /// <param name="railMLDocument"></param>
        /// <param name="railMLVersion"></param>
        /// <returns></returns>
        public ValidationResult Prevalidate(string filePath, out
railMLDocument railMLDocument, out railMLVersion
railMLVersion)
        {
            railMLDocument = new railMLDocument();
            railMLVersion = new railMLVersion();
            ValidationResult result = new
ValidationResult(railMLDocument);
            try
            {
                railMLDocument = new
railMLDocument(LoadRailML(filePath), filePath,
LoadLines(filePath));
            }
            catch (XmlException ex)
            {
                result.AddError(ex.LineNumber, ex.Message);
```

```
            return result;
        }
        catch (Exception ex)
        {
            result.AddError(ex.Message);
            return result;
        }
        SetDirectory(filePath);

        XElement rootNode = railMLDocument.document.Root;

        //Checking presence of railML version in the file
        var versionValue =
rootNode.Attributes("version").SingleOrDefault();

        try
        {
            railMLNamespaces railMLNamespaces = new
railMLNamespaces();
            string railMLNamespace =
railMLNamespaces.IdentifyRailMLNamespace(railMLDocument.do
cument);

railMLVersion.IdentifyRailMlVersion((string)versionValue,
railMLNamespace);

            if (string.IsNullOrEmpty(railMLNamespace))
            {
                result.AddError(-1, $"railML namespace is missing in
the file '{filePath}'.");
            }
            else
            {
                if (!string.IsNullOrEmpty(railMLVersion.Version) &&
!string.IsNullOrEmpty(railMLVersion.VersionFromNamespace))
                {
                    if (!string.Equals(railMLVersion.Version,
railMLVersion.VersionFromNamespace))
                    {
                        result.AddWarning(-1, $"Version and namespace
do not match in the file '{filePath}'. " +
                            $"Version = '{railMLVersion.Version}', version
from namespace = '{railMLVersion.VersionFromNamespace}'. " +
                            $"Version from namespace will be used during
validation.");
                    }
                }
            }

        }
        catch (InvalidDataException ex)
        {
            var lineInfo = (IXmlLineInfo)rootNode;
            result.AddError(lineInfo.LineNumber, ex.Message);
            return result;
        }

        return result;
    }
  }


}
```

1.5 Текст файлу railMLNamespaces.cs модуля railVIVID.Validation

```
using System;
using System.Collections.Generic;
using System.Xml.Linq;

namespace railVIVID.Validation
{
  public class railMLNamespaces
  {
    public static List<string> StandardRailMLNamespaces = new
List<string>
        {
            "http://www.railml.org/schemas/2009",
            "http://www.railml.org/schemas/2011",
            "http://www.railml.org/schemas/2013",
            "http://www.railml.org/schemas/2016",
            "https://www.railml.org/schemas/2018",
            "https://www.railml.org/schemas/2021",
            "https://www.railml.org/schemas/3.1",
            "https://www.railml.org/schemas/3.2"
        };

    public string IdentifyRailMLNamespace(XDocument
railMLDoc)
                        {
        XElement root = railMLDoc.Root;

        HashSet<string> rootNamespaces =
GetRootNamespaces(root);
        foreach (var ns in rootNamespaces)
        {
            if (StandardRailMLNamespaces.Contains(ns))
            {
                return ns;
            }
        }

        return null;
    }

    private static HashSet<string> GetRootNamespaces(XElement
root)
    {
        HashSet<string> namespaces = new HashSet<string>();

        //Add the namespace of the root element
        if (!string.IsNullOrEmpty(root.Name.NamespaceName))
        {
            namespaces.Add(root.Name.NamespaceName);
        }

        //Add namespaces declared as attributes (e.g., xmlns:prefix)
        foreach (XAttribute attr in root.Attributes())
        {
            if (attr.IsNamespaceDeclaration)
            {
                namespaces.Add(attr.Value);
            }
        }

        return namespaces;
    }

  }
}
```

1.6 Текст файлу railMLVersion.cs модуля railVIVID.Validation

```
namespace railVIVID.Validation
{
   public class railMLVersion
   {
      private string version = "";

      //Actual version attribute in the file
      public string Version
      {
         get { return version; }
         set { version = value; }
      }

      private string versionNamespace = "";

      //Prioritized version deducted from namespace
      public string VersionFromNamespace
      {
         get { return versionNamespace; }
         set { versionNamespace = value; }
      }

      /// <summary>
      /// Sets railML versions based on actual version attribute and
railML namespace
      /// </summary>
      /// <param name="version"></param>
      /// <param name="railMLNamespace"></param>
      public void IdentifyRailMlVersion(string version, string
railMLNamespace)
      {
         Version = "";
         VersionFromNamespace = "";
         if (!string.IsNullOrEmpty(railMLNamespace))
         {
            VersionFromNamespace =
GetVersionFromNamespace(railMLNamespace);
         }
         if (!string.IsNullOrEmpty(version))
         {
            switch (version)
            {
               case "2.0": Version = "2.0"; break;
               case "2.1": Version = "2.1"; break;
               case "2.2": Version = "2.2"; break;
               case "2.3": Version = "2.3"; break;
               case "2.4": Version = "2.4"; break;
               case "2.5": Version = "2.5"; break;
               case "3.1": Version = "3.1"; break;
               case "3.2": Version = "3.2"; break;
            }
         }

      }

      private string GetVersionFromNamespace(string
railMLNamespace)
      {
         switch (railMLNamespace)
         {
            case "http://www.railml.org/schemas/2009": return "2.0";
            case "http://www.railml.org/schemas/2011": return "2.1";
            case "http://www.railml.org/schemas/2013": return "2.2";
            case "http://www.railml.org/schemas/2016": return "2.3";
            case "https://www.railml.org/schemas/2018": return "2.4";
            case "https://www.railml.org/schemas/2021": return "2.5";
            case "https://www.railml.org/schemas/3.1": return "3.1";
            case "https://www.railml.org/schemas/3.2": return "3.2";
            default: return "";
         }
      }
   }
}
```

1.7 Текст файлу SchemaControl.cs модуля railVIVID.Validation

```
using railVIVID.Validation.GeneralInformation;
using System;
using System.Collections.Generic;
using System.IO;
using System.Net;
using System.Reflection;
using System.Text.RegularExpressions;
using System.Xml;
using System.Xml.Linq;
using System.Xml.Schema;

namespace railVIVID.Validation
{
   /// <summary>
   /// Class for everything related to handling of schemas
   /// </summary>
   public class SchemaControl
   {
      private static string defaultRailMLLocation = "";

      /// <summary>
      /// Expected railML location depending on the railML version
and the situation (with/without connection)
      /// </summary>
      public static string DefaultRailMLLocation
      {
         get { return defaultRailMLLocation; }
         set { defaultRailMLLocation = value; }
      }

      //Patterns for possible official railML locations
      private Dictionary<string, string> locationPatterns = new
Dictionary<string, string>
      {
         { "2.0",
@"https?://(www\.|schemas\.)?railml\.org/(schemas/)?2009/railML-
2\.0/(railML\.xsd|schema/railML\.xsd)" },
         { "2.1",
@"https?://(www\.|schemas\.)?railml\.org/(schemas/)?2011/railML-
2\.1/(railML\.xsd|schema/railML\.xsd)" },
         { "2.2",
@"https?://(www\.|schemas\.)?railml\.org/(schemas/)?2013/railML-
2\.2/(railML\.xsd|schema/railML\.xsd)" },
         { "2.3",
@"https?://(www\.|schemas\.)?railml\.org/(schemas/)?2016/railML-
2\.3/(railML\.xsd|schema/railML\.xsd)" },
         { "2.4",
@"https?://(www\.|schemas\.)?railml\.org/(schemas/)?2018/railML-
2\.4/(railML\.xsd|schema/railML\.xsd)" },
         { "2.5",
@"https?://(www\.|schemas\.)?railml\.org/(schemas/)?2021/railML-
2\.5/(railML\.xsd|schema/railML\.xsd)" },
         { "3.1",
@"https?://(www\.|schemas\.)?railml\.org/(schemas/)?3\.1/railml3\.xs
d" },
         { "3.2",
@"https?://(www\.|schemas\.)?railml\.org/(schemas/)?3\.2/railml3\.xs
d" }
      };
```

```
        private Dictionary< string, string > schemaLocations = new
Dictionary<string, string>();

        /// <summary>
        /// Found referenced valid schemas in the file
        /// </summary>
        public Dictionary<string, string> SchemaLocations
        {
            get { return schemaLocations; }
            set { schemaLocations = value; }
        }

        public SchemaControl()
        {
            defaultRailMLLocation = "";
        }

        public void SetDefaultLocationFromVersion(string version)
        {
            if (!string.IsNullOrEmpty(version))
            {
                //If there is Internet connection or access to remote railML
schemas
                if
(IsUrlAccessible("https://schemas.railml.org/2009/railML-
2.0/railML.xsd"))
                {
                    switch (version)
                    {
                        case "2.0":
                            defaultRailMLLocation =
"https://schemas.railml.org/2009/railML-2.0/railML.xsd"; break;
                        case "2.1":
                            defaultRailMLLocation =
"https://schemas.railml.org/2011/railML-2.1/railML.xsd"; break;
                        case "2.2":
                            defaultRailMLLocation =
"https://schemas.railml.org/2013/railML-2.2/railML.xsd"; break;
                        case "2.3":
                            defaultRailMLLocation =
"https://schemas.railml.org/2016/railML-2.3/schema/railML.xsd";
break;
                        case "2.4":
                            defaultRailMLLocation =
"https://schemas.railml.org/2018/railML-2.4/schema/railML.xsd";
break;
                        case "2.5":
                            defaultRailMLLocation =
"https://schemas.railml.org/2021/railML-2.5/schema/railML.xsd";
break;
                        case "3.1":
                            defaultRailMLLocation =
"https://schemas.railml.org/3.1/railml3.xsd"; break;
                        case "3.2":
                            defaultRailMLLocation =
"https://schemas.railml.org/3.2/railml3.xsd"; break;
                    }
                }
                else
                {
                    //Otherwise using embedded resources (local solution)
                    defaultRailMLLocation = "local";
                }


            }

        }

        /// <summary>
        /// Adds found non-railML schemas
        /// </summary>
        /// <param name="schemaSet"></param>
        public void AddOtherShemas(XmlSchemaSet schemaSet)
        {
            if(defaultRailMLLocation != "local")
            {
                AddW3(schemaSet);
            }

            foreach (var pair in schemaLocations)
            {
                schemaSet.Add(pair.Key, pair.Value);
            }
        }

        //Necessary for proper compiling of schemas
        private void AddW3(XmlSchemaSet schemaSet)
        {
            // Add the schema for the XML namespace
            using (StringReader stringReader = new StringReader(
                @"<xs:schema
xmlns:xs='http://www.w3.org/2001/XMLSchema'

xmlns:xml='http://www.w3.org/XML/1998/namespace'>
                    <xs:import
namespace='http://www.w3.org/XML/1998/namespace'

schemaLocation='https://www.w3.org/2001/xml.xsd'/>
                </xs:schema>"))
            {
                schemaSet.Add(XmlSchema.Read(stringReader, null));
            }
        }

        /// <summary>
        /// For finding information about extensions, adds all referenced
non-railML schemas to the list without their checking
        /// </summary>
        /// <param name="doc"></param>
        public void SetAllSchemas(XDocument doc)
        {
            XNamespace xsi = "http://www.w3.org/2001/XMLSchema-
instance";

            var root = doc.Root;
            if (root == null)
                return;

            var schemaLocationAttribute = root.Attribute(xsi +
"schemaLocation");
            if (schemaLocationAttribute != null)
            {
                var schemaLocationPairs =
schemaLocationAttribute.Value.Split(new[] { ' ' },
StringSplitOptions.RemoveEmptyEntries);
                for (int i = 0; i < schemaLocationPairs.Length; i += 2)
                {
                    if (i + 1 < schemaLocationPairs.Length)
                    {
                        var namespaceUri = schemaLocationPairs[i];
                        var schemaLocationUri = schemaLocationPairs[i + 1];

                        //If schema has railML namespace, not include it into
extensions
                        if ((namespaceUri !=
doc.Root.Name.NamespaceName))
                        {
                            schemaLocations.Add(namespaceUri,
schemaLocationUri);
                        }
                    }
                }
            }

        }

        /// <summary>
        /// For validation, additionally checks the validity of referenced
schemas before adding the non-railML ones to the list
```

```
    /// </summary>
    /// <param name="doc"></param>
    /// <param name="result"></param>
    public void SetAndCheckAllSchemas(XDocument doc,
ValidationResult result, string railMLFilePath)
    {
      XNamespace xsi = "http://www.w3.org/2001/XMLSchema-
instance";

      var root = doc.Root;
      if (root == null)
        return;

      bool railMLSchemaPresent = false;

      var schemaLocationAttribute = root.Attribute(xsi +
"schemaLocation");
      if (schemaLocationAttribute != null)
      {
        var schemaLocationPairs =
schemaLocationAttribute.Value.Split(new[] { ' ' },
StringSplitOptions.RemoveEmptyEntries);
        for (int i = 0; i < schemaLocationPairs.Length; i += 2)
        {
          if (i + 1 < schemaLocationPairs.Length)
          {
            var namespaceUri = schemaLocationPairs[i];
            var schemaLocationUri = schemaLocationPairs[i + 1];

            //If the schema has the railML namespace, do not
include it in schema locations
            //because the official version is used
            if (namespaceUri !=
doc.Root.Name.NamespaceName)
            {
              if (IsAccessibleAndValid(namespaceUri,
schemaLocationUri, result, railMLFilePath))
              {
                schemaLocations.Add(namespaceUri,
schemaLocationUri);
              }
            }
            else
            {
              //Set that file references a railML schema in
schemaLocations
              railMLSchemaPresent = true;
              bool locationOfficial = false;
              //Checking if location matches pattern of official
              foreach (var location in locationPatterns)
              {
                if (Regex.IsMatch(schemaLocationUri,
location.Value))
                {
                  locationOfficial = true;
                }
              }

              if (!locationOfficial)
              {
                result.AddWarning(-1, $"The file references
unknown local/unofficial version of railML schema " +
                  $": '{schemaLocationUri}'. The official
version will been used for validation.");

                if (!IsAccessibleAndValid(namespaceUri,
schemaLocationUri, result, railMLFilePath))
                {
                  result.AddWarning(-1, "Referenced railML
schema isn't valid/accesible!");
                }

              }
            }
          }
        }

        if (!railMLSchemaPresent)
        {
          result.AddWarning(-1, "The file does not reference a
railML schema in schemaLocations." +
            " The official railML version will be used for
validation.");
        }
      }



    private bool IsAccessibleAndValid(string namespaceUri, string
schemaLocationUri, ValidationResult result, string railMLFilePath)
    {
      //Check if the schema location is a remote URL
      if (schemaLocationUri.StartsWith("http://") ||
schemaLocationUri.StartsWith("https://"))
      {
        if (IsUrlAccessible(schemaLocationUri))
        {
          if (IsXSDValid(namespaceUri, schemaLocationUri,
result))
          {
            return true;
          }

        }
      }
      else
      {
        string resolvedSchemaUri =
GetLocalPath(schemaLocationUri);

        if (System.IO.File.Exists(resolvedSchemaUri))
        {
          if (IsXSDValid(namespaceUri, schemaLocationUri,
result))
          {
            return true;
          }
        }
      }
      return false;
    }

    private bool IsXSDValid(string namespaceUri, string
schemaLocationUri, ValidationResult result)
    {
      XmlSchemaSet schemaSet = new XmlSchemaSet();
      schemaSet.XmlResolver = new XmlUrlResolver();

      try
      {
        AddW3(schemaSet);
        schemaSet.Add(namespaceUri, schemaLocationUri);
        schemaSet.Compile();
      }catch(Exception e)
      {
        result.AddWarning(-1, $"Issue with schema
'{schemaLocationUri}' or related schemas: '{e.Message}'");
        return false;
      }
      return true;
    }

    private string GetLocalPath(string schemaLocationUri)
    {
      if (schemaLocationUri.StartsWith("file:"))
      {
        schemaLocationUri = new
Uri(schemaLocationUri).LocalPath;
      }
```

```
        //Check if the schemaLocationUri is an absolute path
        if (Path.IsPathRooted(schemaLocationUri))
        {
            return schemaLocationUri;
        }
        else
        {
            //string location =
Path.GetDirectoryName(railMLFilePath);
            return Path.GetFullPath(schemaLocationUri);
        }
    }

    private bool IsUrlAccessible(string url)
    {
        try
        {
            var request = (HttpWebRequest)WebRequest.Create(url);
            request.Method = "HEAD"; //To check if the URL is
accessible
            request.Timeout = 5000;

            using (var response =
(HttpWebResponse)request.GetResponse())
            {
                if (response.StatusCode == HttpStatusCode.OK)
                {
                    response.Close();
                    return true;
                }
                else return false;
            }

        }
        catch (WebException)
        {
            return false;
        }
    }

    public static string FormatVersionForResource(string version)
    {
        //Split the version by '.', then join with '._' to create the format
for embedded resource
        return string.Join("._", version.Split('.'));
    }

    /// <summary>
    /// Adds railML embedded resource file to the XMLSchemaSet.
    /// Needed because non-imported local schemas need to be
added directly.
    /// </summary>
    /// <param name="schemaSet"></param>
    /// <param name="resourceName"></param>
    /// <param name="version"></param>
    public void LoadRailMLFromResource(XmlSchemaSet
schemaSet, string resourceName, string version)
    {
        Assembly assembly = Assembly.GetExecutingAssembly();
        string versionResource =
FormatVersionForResource(version);
        string fullResourceName =
$"{assembly.GetName().Name}.Resources.railML._{versionResourc
e}.{resourceName}";

        AddStreamToSchemas(fullResourceName, assembly,
schemaSet);
    }

    private void AddStreamToSchemas(string fullResourceName,
Assembly assembly, XmlSchemaSet schemaSet)
    {
        using (Stream stream =
assembly.GetManifestResourceStream(fullResourceName))
        {
            if (stream == null)
            {
                throw new FileNotFoundException($"Resource
'{fullResourceName}' not found.");
            }

            using (XmlReader reader = XmlReader.Create(stream))
            {
                schemaSet.Add(null, reader);
            }
        }
    }

  }
}
```

1.8 Текст файлу ValidationIssueBasic.cs модуля railVIVID.Validation

```
namespace railVIVID.Validation
{
  public class ValidationIssueBasic
  {
    public string description { get; protected set; }

    public ValidationIssueBasic(string description)
    {
      this.description = description;
    }

    public override bool Equals(object obj)
    {
      if (obj is ValidationIssueBasic otherIssue)
      {
        return description == otherIssue.description;
      }
      return false;
    }

    public override int GetHashCode()
    {
      return description.GetHashCode();
    }
  }
}
```

1.9 Текст файлу ValidationIssueDetailed.cs модуля railVIVID.Validation

```
using railVIVID.Validation.GeneralInformation;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace railVIVID.Validation
{
   public class ValidationIssueDetailed : ValidationIssueBasic
   {
      public ValidationIssueDetailed(string description) :
base(description)
      {
      }

      public ValidationIssueDetailed(int line, string desc,
railMLDocument document) : this(desc)
      {
         lines = [new Tuple<int, string>(line, document.lines[line-1])];
      }
      public List<Tuple<int, string>> lines { get; private set; }

      public void AddLine(int line, railMLDocument document)
      {
         if (!lines.Any(t => t.Item1 == line))
         {
            lines.Add(new Tuple<int, string>(line,
document.lines[line-1]));
         }
      }

   }
}
```

1.10 Текст файлу ValidationResult.cs модуля railVIVID.Validation

```
using railVIVID.Validation.GeneralInformation;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace railVIVID.Validation
{
   public class ValidationResult
   {
      public List<ValidationIssueBasic> errors { get; private set; }
      public List<ValidationIssueBasic> warnings { get; private set; }
      public List<ValidationIssueBasic> semanticViolations { get;
private set; }
      private railMLDocument document;

      public ValidationResult(railMLDocument railMLDocument)
      {
         document = railMLDocument;
         errors = new List<ValidationIssueBasic>();
         warnings = new List<ValidationIssueBasic>();
         semanticViolations = new List<ValidationIssueBasic>();
      }

      public void AddError(string description)
      {
         AddIssue(errors, description);
      }

      public void AddWarning(string description)
      {
         AddIssue(warnings, description);
      }

      public void AddError(int line, string description)
      {

         if (line != null && line > 0)
         {
            AddIssue(errors, line, description);
         }
         else AddIssue(errors, description);
      }

      public void AddWarning(int line, string description)
      {
         if (line != null && line > 0)
         {
            AddIssue(warnings, line, description);
         }
         else AddIssue(warnings, description);
      }

      public void AddSemanticWarning(int line, string description)
      {
         if (line != null && line > 0)
         {
            AddIssue(semanticViolations, line, description);
         }
         else AddIssue(semanticViolations, description);
      }

      private void AddIssue(List<ValidationIssueBasic> issues, string
description)
      {
         var existingIssue = issues.FirstOrDefault(issue =>
issue.description == description);
         if (existingIssue == null)
         {
            ValidationIssueBasic newIssue = new
ValidationIssueBasic(description);
            issues.Add(newIssue);
         }
      }

      private void AddIssue(List<ValidationIssueBasic> issues, int
line, string description)
      {
         ValidationIssueDetailed existingIssue =
issues.FirstOrDefault(issue => issue.description == description) as
ValidationIssueDetailed;

         if (existingIssue != null)
         {
            existingIssue.AddLine(line, document);
         }
         else
         {
            ValidationIssueDetailed newIssue = new
ValidationIssueDetailed(line, description, document);
            issues.Add(newIssue);
         }
      }
   }
}
```

1.11 Текст файлу SemanticValidator.cs модуля railVIVID.Validation.SemanticValidation

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Xml;
using System.Xml.Linq;

namespace railVIVID.Validation.SemanticValidation
{
   public class SemanticValidator
   {
      private XDocument xDocument;
      private ValidationResult validationResult;
      private XNamespace ns;

      public SemanticValidator() { }

      public ValidationResult Validate(XDocument xDocument, ref
ValidationResult result)
      {
         railMLNamespaces railMLNamespaces = new
railMLNamespaces();
         ns =
railMLNamespaces.IdentifyRailMLNamespace(xDocument);
         this.xDocument = xDocument;
         this.validationResult = result;
         checkStartEndOverlap();
         CheckTT002();
         CheckTT003();
         CheckTT004();
         CheckTT005();
         CheckTT006();
         CheckTT009();
         CheckTT012();
         CheckTT014();
         CheckTT015();
         CheckTT016();
         CheckTT017();
         CheckIS015();
         return result;
      }

      int GetLineNumber(XElement element)
      {
         if (element is IXmlLineInfo lineInfo &&
lineInfo.HasLineInfo())
         {
            return lineInfo.LineNumber;
         }
         return -1;
      }

      private void checkStartEndOverlap()
      {
         CheckElementStartEndOverlap("TT:001", "timetablePeriod",
"timetable");
         CheckElementStartEndOverlap("TT:018", "specialService",
"timetable");
         CheckElementStartEndOverlap("C0:002",
"operatingPeriodRef", "timetable");
         CheckElementStartEndOverlap("CO:002", "operatingPeriod",
"timetable");
         CheckElementStartEndOverlap("TT:019", "operatingDay",
"timetable");
         CheckElementStartEndOverlap("CO:002", "circulation",
"timetable");
         CheckElementStartEndOverlap("CO:002", "blockPart",
"timetable"); //not yet done
         CheckElementStartEndOverlap("RS:001", "designator",
"rollingstock");
         CheckElementStartEndOverlap("RS:002", "operator",
"rollingstock");
         CheckElementStartEndOverlap("RS:003", "owner",
"rollingstock");
         CheckElementStartEndOverlap("RS:004", "state",
"rollingstock");
         CheckElementStartEndOverlap("IS:020/IS:021", "state",
"infrastructure");
         CheckElementStartEndOverlap("CO:001", "designator",
"infrastructure");
         CheckElementStartEndOverlap("CO:001", "designator",
"metadata");
         CheckElementStartEndOverlap("CO:002", "phase",
"metadata");
      }

      private void CheckElementStartEndOverlap(string
semConName, string element, string type)
      {
         var elementList = xDocument.Descendants(ns + type)
                  .Descendants(ns + element);

         checkStartEnd(semConName, elementList);

         if (!new[] { "circulation", "blockPart", "operatingPeriod",
"operatingPeriodRef" }.Contains(element))
         {
            checkOverlaps(semConName, elementList, element, type);
         }
      }

      private void checkStartEnd(string semConName,
IEnumerable<XElement> elementList)
      {
         foreach (var element in elementList)
         {
            if (!DateTimeISCorrect(element.Attribute("startDate"),
                  element.Attribute("endDate"),
                  element.Attribute("startTime"),
                  element.Attribute("endTime")))
            {
               XElement closestNodeWithId = ExtractNode(element);
               int lineNumber = GetLineNumber(closestNodeWithId);

               validationResult.AddSemanticWarning(lineNumber,
$"Violation of constraint {semConName} in
{element.Name.LocalName}. " +
                        $"Associated with
{closestNodeWithId.Name.LocalName} id =
{ExtractId(element)}.");
            }
         }
      }

      private string ExtractId(XElement element)
      {
         var id = element.Attribute("id")?.Value;
         if (id != null)
         {
            return id;
         }
         //If 'id' is not present, check the parent element recursively
         return element.Parent != null ? ExtractId(element.Parent) :
"unknown";
      }

      private XElement ExtractNode(XElement element)
      {
         if (element.Attribute("id") != null)
```

```
        {
            return element;
        }
        //If 'id' is not present, check the parent element recursively
        return element.Parent != null ? ExtractNode(element.Parent) :
element;
    }

    private void checkOverlaps(string semConName,
IEnumerable<XElement> elementList, string element, string type)
    {
        if (elementList.Count() > 0)
        {
            var parentNames = new
HashSet<string>(elementList.Select(e =>
e.Parent?.Name.LocalName));

            ConflictType conflictType = element == "designator" ?
ConflictType.ConditionDateOverlap : ConflictType.DateOverlap;

            foreach (var parentName in parentNames)
            {

                var overlappingNodes = CheckConflict(parentName,
element, type, conflictType);

                foreach (var entry in overlappingNodes)
                {
                    var pairList = entry.ToList();
                    var association = ExtractId(pairList[0]) ==
ExtractId(pairList[1])
                        ? $"{ExtractNode(pairList[0]).Name.LocalName}
id = {ExtractId(pairList[0])}"
                        : $"{ExtractNode(pairList[0]).Name.LocalName}
id = {ExtractId(pairList[0])} as well as in
{ExtractNode(pairList[1]).Name.LocalName} id =
{ExtractId(pairList[1])}";

                    int lineNumber = GetLineNumber(pairList[0]);

                    validationResult.AddSemanticWarning(lineNumber,
$"Violation of constraint {semConName} in
{pairList[0].Name.LocalName}. Associated with {association}. ");
                }
            }
        }

    }

    enum ConflictType
    {
        DateOverlap,
        ConditionDateOverlap,
        GradientPos
    }

    private HashSet<HashSet<XElement>> CheckConflict(string
parentName, string element,
                                        string type, ConflictType
conflictType)
    {
        HashSet<HashSet<XElement>> overlaps = new
HashSet<HashSet<XElement>> ();

        var parentNodes = xDocument.Descendants(ns +
type).Descendants(ns + parentName).ToList();

        foreach (var searchContext in parentNodes)
        {
            var elements = searchContext.Descendants(ns +
element).ToList();

            for (int j = 0; j < elements.Count; j++)
            {
                var nodeJ = elements[j];

                for (int k = 0; k < elements.Count; k++)
                {
                    if (j != k)
                    {
                        var nodeK = elements[k];
                        bool result = false;

                        switch (conflictType)
                        {
                            case ConflictType.DateOverlap:
                                result = DateTimeOverlap(nodeJ, nodeK);
                                break;
                            case ConflictType.GradientPos:
                                result = PosOverlap(nodeJ, nodeK);
                                break;
                            case ConflictType.ConditionDateOverlap:
                                result = ConditionDateTimeOverlap(nodeJ,
nodeK);
                                break;
                        }

                        if (result)
                        {
                            var pair = new HashSet<XElement> { nodeJ,
nodeK };
                            overlaps.Add(pair);
                        }
                    }
                }
            }
        }

        return overlaps;

    }

    private bool DateTimeOverlap(XElement firstElement,
XElement secondElement)
    {
        string fStartDateStr =
firstElement.Attribute("startDate")?.Value;
        string fEndDateStr =
firstElement.Attribute("endDate")?.Value;
        string fStartTimeStr =
firstElement.Attribute("startTime")?.Value;
        string fEndTimeStr =
firstElement.Attribute("endTime")?.Value;

        string sStartDateStr =
secondElement.Attribute("startDate")?.Value;
        string sEndDateStr =
secondElement.Attribute("endDate")?.Value;
        string sStartTimeStr =
secondElement.Attribute("startTime")?.Value;
        string sEndTimeStr =
secondElement.Attribute("endTime")?.Value;

        if (fStartDateStr != null && fEndDateStr != null &&
fStartTimeStr != null && fEndTimeStr != null &&
    sStartDateStr != null && sEndDateStr != null &&
sStartTimeStr != null && sEndTimeStr != null)
        {
            DateTime combinedFStart =
DateTime.Parse($"{fStartDateStr} {fStartTimeStr}");
            DateTime combinedFEnd =
DateTime.Parse($"{fEndDateStr} {fEndTimeStr}");
            DateTime combinedSStart =
DateTime.Parse($"{sStartDateStr} {sStartTimeStr}");
            DateTime combinedSEnd =
DateTime.Parse($"{sEndDateStr} {sEndTimeStr}");

            return combinedFStart < combinedSEnd &&
combinedFEnd > combinedSStart;
        }
```

```
        else if (fStartDateStr != null && fEndDateStr != null &&
sStartDateStr != null && sEndDateStr != null)
        {
            DateTime fStartDate = DateTime.Parse(fStartDateStr);
            DateTime fEndDate = DateTime.Parse(fEndDateStr);
            DateTime sStartDate = DateTime.Parse(sStartDateStr);
            DateTime sEndDate = DateTime.Parse(sEndDateStr);

            return fStartDate <= sEndDate && fEndDate >=
sStartDate;
        }
        else return false;
    }

    private bool PosOverlap(XElement firstElement, XElement
secondElement)
    {
        int firstPos = int.Parse(firstElement.Attribute("pos")?.Value);
        int secondPos =
int.Parse(secondElement.Attribute("pos")?.Value);
        return firstPos == secondPos;
    }

    private bool ConditionDateTimeOverlap(XElement
firstElement, XElement secondElement)
    {
        string firstRegister =
firstElement.Attribute("register")?.Value;
        string secondRegister =
secondElement.Attribute("register")?.Value;
        if (string.Equals(firstRegister, secondRegister))
        {
            return DateTimeOverlap(firstElement, secondElement);
        }
        else return false;
    }

    private bool DateTimeISCorrect(XAttribute startDateNode,
XAttribute endDateNode,
                                   XAttribute startTimeNode, XAttribute
endTimeNode)
    {
        DateTime startDate;
        DateTime startTime;
        DateTime endDate;
        DateTime endTime;
        if (startDateNode != null && endDateNode != null &&
startTimeNode != null && endTimeNode != null)
        {
            startDate = DateTime.Parse(startDateNode.Value);
            startTime = DateTime.Parse(startTimeNode.Value);
            DateTime combinedStart =
startDate.Date.Add(startTime.TimeOfDay);

            endDate = DateTime.Parse(endDateNode.Value);
            endTime = DateTime.Parse(endTimeNode.Value);
            DateTime combinedEnd =
endDate.Date.Add(endTime.TimeOfDay);

            return combinedEnd >= combinedStart;
        }
        else if (startDateNode != null && endDateNode != null)
        {
            startDate = DateTime.Parse(startDateNode.Value);
            endDate = DateTime.Parse(endDateNode.Value);
            return endDate >= startDate;
        }
        else if (startTimeNode != null && endTimeNode != null)
        {
            int line;
            if (startDateNode is IXmlLineInfo lineInfo)
            {
                line = lineInfo.LineNumber;
            }
            else
            {
                line = -1;
            }
            validationResult.AddWarning(line, "Start and end times
provided without dates!");
            return false;
        }
        else return true;

    }

    //Checks if the attribute sequence of ocpTT is increasing
according to the train path./
    private void CheckTT002()
    {
        int firstSequenceValue;
        string secondSequenceValue;

        var trainParts = xDocument.Descendants(ns + "timetable")
                .Descendants(ns + "trainPart");
        foreach (var trainPart in trainParts)
        {
            var ocpTTs = trainPart.Descendants()
                .Where(e => (e.Name.LocalName == "ocpTT" &&
                        (e.Parent?.Name.LocalName == "ocpsTT" ||
e.Parent == trainPart))).ToList();
            for (int j = 0; j < ocpTTs.Count - 1; j++)
            {
                //Get the sequence attributes of the current and next
ocpTT elements
                XAttribute firstSequence =
ocpTTs[j].Attribute("sequence");
                XAttribute secondSequence = ocpTTs[j +
1].Attribute("sequence");

                if (firstSequence != null && secondSequence != null)
                {
                    //Parse the sequence values
                    firstSequenceValue = int.Parse(firstSequence.Value);
                    secondSequenceValue = secondSequence.Value;

                    //Check if the first sequence is greater than or equal to
the second
                    if (firstSequenceValue >=
int.Parse(secondSequenceValue))
                    {

                        int lineNo1 = GetLineNumber(ocpTTs[j]);
                        int lineNo2 = GetLineNumber(ocpTTs[j+1]);
                        string message = $"Violation of constraint TT:002
in trainPart id = {trainPart.Attribute("id")?.Value}. " +
                            $"First ocpTT sequence =
{firstSequence.Value}, " +
                            $"ocpRef =
{ocpTTs[j].Attribute("ocpRef")?.Value}; " +
                            $"second ocpTT sequence =
{secondSequence.Value}, ocpRef = {ocpTTs[j +
1].Attribute("ocpRef")?.Value}.";

                        validationResult.AddSemanticWarning(lineNo1,
message);
                        validationResult.AddSemanticWarning(lineNo2,
message);
                    }
                }
            }

        }
    }

    //Checks that there are no two <gradientChange> belonging to
the same track with the same @pos attribute
    //private void checkIS019()
    //{
```

```
    //    var overlapingNodes = CheckConflict("gradientChanges",
"gradientChange",
    //          "infrastructure", ConflictType.GradientPos);
    //    foreach (var entry in overlapingNodes)
    //    {
    //       List<XElement> pairList = new List<XElement>(entry);
    //
validationResult.AddSemanticWarning(GetLineNumber(pairList[0]),
    //          $"Violation of constraint IS:019 in
{pairList[0].Name.LocalName}. " +
    //          $"Associated with
{ExtractNode(pairList[0]).Name.LocalName} id =
{ExtractId(pairList[0])}" +
    //          $" as well as in
{ExtractNode(pairList[1]).Name.LocalName}" +
    //          $"id = {ExtractId(pairList[1])}. First occurrences : ");

    //    }
    //}

    private void CheckTT014()
    {
       var ocpTTs = xDocument.Descendants(ns + "timetable")
                    .Descendants(ns + "ocpTT")
                    .Where(e => (string)e.Attribute("ocpType") ==
"pass")
                    .ToList();
       //we need to check that if the ocpType is pass only the
departure time is specified in times
       foreach (var ocpTT in ocpTTs)
       {
          var times = ocpTT.Elements(ns + "times");

          if (!times.Any())
          {
             continue;
          }

          foreach (var time in times)
          {
             if (time.Attribute("arrival") != null)
             {
                int lineNo = GetLineNumber(ocpTT);
                validationResult.AddSemanticWarning(lineNo,
$"Violation of constraint TT:014 in ocpTT, associated with " +
                   $"{ExtractNode(ocpTT).Name.LocalName} id =
{ExtractId(ocpTT)}.");
                break;
             }
          }
       }
    }

    private void CheckTT012()
    {
       var trainParts = xDocument
          .Descendants(ns + "timetable")
          .Descendants(ns + "trainPart")
          .Where(e => e.Descendants(ns + "ocpsTT")
          .Descendants(ns + "ocpTT")
          .Descendants(ns + "times")
          .Any(t => (string)t.Attribute("scope") == "actual"))
          .ToList();

       var operatingPeriods = xDocument
          .Descendants(ns + "timetable")
          .Descendants(ns + "operatingPeriods")
          .ToList();

       if (!operatingPeriods.Any())
       {
          return;
       }

       var operatingPeriodsNode = operatingPeriods.First();

       //make sure that for each trainPart the reference
operatingPeriodRef refers to an operatingPeriod, that has only a
single day selected
       //so either the bitmask only has a single zero, or startdate and
enddate are specified like this or a single date is specified for
specialService
       foreach (var trainPart in trainParts)
       {
          var operatingPeriodRef = trainPart
             .Descendants(ns + "operatingPeriodRef")
             .FirstOrDefault();
          if (operatingPeriodRef == null)
          {
             continue;
          }
          var reference =
operatingPeriodRef.Attribute("ref")?.Value;
          var operatingPeriod = operatingPeriodsNode
             .Descendants(ns + "operatingPeriod")
             .FirstOrDefault(op => (string)op.Attribute("id") ==
reference);

          var bitmask =
DetermineBitmaskOfOperatingPeriod(operatingPeriod);

          if (bitmask == null)
          {
             continue;
          }
          if (bitmask.Count(ch => ch == '1') != 1)
          {
             int lineNo = GetLineNumber(trainPart);
             validationResult.AddSemanticWarning(lineNo,
$"Violation of constraint TT:012 in trainPart id =
{ExtractId(trainPart)}");
          }
       }

    }

    private string DetermineBitmaskOfOperatingPeriod(XElement
operatingPeriod)
    {
       string opBitmask;
       if (operatingPeriod.Attribute("bitMask") != null)
       {
          opBitmask = operatingPeriod.Attribute("bitMask").Value;
       }
       else
       {
          opBitmask = GenerateBitmask(operatingPeriod);
       }
       return opBitmask;
    }

    private string GenerateBitmask(XElement operatingPeriod)
    {
       string opBitmask;
       //no bitmask available. we need to assemble it using timetable
period and special service
       var timetablePeriods = xDocument
          .Descendants(ns + "timetable")
          .Descendants(ns + "timetablePeriod")
          .ToList();

       var startDateNode =
timetablePeriods.First().Attribute("startDate");
       var endDateNode =
timetablePeriods.First().Attribute("endDate");

       if (startDateNode == null || endDateNode == null)
       {
          //we cannot work with this.
          return null;
```

```
        }

        var startDate = DateTime.Parse(startDateNode.Value);
        var endDate = DateTime.Parse(endDateNode.Value);
        //find out how many days are in between
        var days = (endDate - startDate).Days;
        //create a string with that many 0s
        opBitmask = new string('0', days);
        //now we need to check startDate and endDate of the
operatingPeriod
        var opStartDateNode =
operatingPeriod.Attribute("startDate");
        var opEndDateNode = operatingPeriod.Attribute("endDate");

        if (opStartDateNode != null && opEndDateNode != null)
        {
            //mark the days of the bitmask that are indicated by
opStartDate and opEndDate as 1s
            var opStartDate =
DateTime.Parse(opStartDateNode.Value);
            var opEndDate = DateTime.Parse(opEndDateNode.Value);
            var opStart = (opStartDate - startDate).Days;
            var opEnd = (opEndDate - startDate).Days;
            opBitmask = opBitmask.Substring(0, opStart) + new
string('1', opEnd - opStart) + opBitmask.Substring(opEnd);
        }
        //now we need to check the special services
        var specialServices = operatingPeriod
            .Descendants(ns + "specialService")
            .Where(s => (string)s.Attribute("type") == "include")
            .ToList();

        //for each special service we need to mark the days as 1s
        foreach (var specialService in specialServices)
        {
            var specialStartDateNode =
specialService.Attribute("startDate");
            var specialEndDateNode =
specialService.Attribute("endDate");

            if (specialStartDateNode != null && specialEndDateNode
!= null)
            {
                var specialStartDate =
DateTime.Parse(specialStartDateNode.Value);
                var specialEndDate =
DateTime.Parse(specialEndDateNode.Value);
                var specialStart = (specialStartDate - startDate).Days;
                var specialEnd = (specialEndDate - startDate).Days;
                opBitmask = opBitmask.Substring(0, specialStart) + new
string('1', specialEnd - specialStart) +
opBitmask.Substring(specialEnd);
            }
            else
            {
                //if no startdate and enddate are specified there still may
be a singleDate attribute be specified
                var singleDateNode =
specialService.Attribute("singleDate");
                if (singleDateNode != null)
                {
                    var singleDate =
DateTime.Parse(singleDateNode.Value);
                    var singleDay = (singleDate - startDate).Days;
                    opBitmask = opBitmask.Substring(0, singleDay) +
"1" + opBitmask.Substring(singleDay + 1);
                }
            }
        }

        return opBitmask;
    }

    //Not tested, as no examples were found and constraint
description is unclear
    private void CheckTT006()
    {
        var ocpTTs = xDocument
            .Descendants(ns + "timetable")
            .Descendants(ns + "ocpTT")
            .ToList();

        foreach (var ocpTT in ocpTTs)
        {
            var ocpTypeAttr = ocpTT.Attribute("ocpType");
            if (ocpTypeAttr == null)
            {
                continue; //if there is no ocpType, no need to check
            }

            if (ocpTypeAttr.Value == "pass")
            {
                if (!CheckTT006Pass(ocpTT))
                {
                    int lineNo = GetLineNumber(ocpTT);
                    validationResult.AddSemanticWarning(lineNo,
$"Violation of constraint TT:006 in trainPart id =
{ExtractId(ocpTT)}.");
                }
            }
            else
            {
                if (!CheckTT006Stop(ocpTT))
                {
                    int lineNo = GetLineNumber(ocpTT);
                    validationResult.AddSemanticWarning(lineNo,
$"Violation of constraint TT:006 in trainPart id =
{ExtractId(ocpTT)}.");
                }
            }
        }
    }


    private bool CheckTT006Stop(XElement ocpTT)
    {
        var stopDescription = ocpTT
            .Elements(ns + "stopDescription")
            .FirstOrDefault();

        if (stopDescription == null)
        {
            return true;
        }

        if (stopDescription.Attribute("guaranteedPass") != null)
        {
            return false;
        }

        bool commercial = true;
        var commercialAttr =
stopDescription.Attribute("commercial");
        if (commercialAttr != null)
        {
            commercial = bool.Parse(commercialAttr.Value);
        }

        if (commercial)
        {
            if (stopDescription.Attribute("operationalStopOrdered") !=
null)
            {
                return false;
            }
        }

        return true;
    }
```

```
    private bool CheckTT006Pass(XElement ocpTT)
    {
        var stopDescription = ocpTT
            .Elements(ns + "stopDescription")
            .FirstOrDefault();

        if (stopDescription == null)
        {
            return true;
        }

        if (stopDescription.Attribute("commercial") != null)
        {
            return false;
        }
        if (stopDescription.Attribute("onOff") != null)
        {
            return false;
        }
        if (stopDescription.Attribute("stopOnRequest") != null)
        {
            return false;
        }
        if (stopDescription.Attribute("operationalStopOrdered") !=
null)
        {
            return false;
        }

        return true;
    }

    private void CheckTT003()
    {
        var blockParts = xDocument
            .Descendants(ns + "blockPart")
            .ToList();

        foreach (var blockPart in blockParts)
        {
            var mission = blockPart.Attribute("mission");
            if (mission == null) continue; //No mission, no need to
check

            var missionValue = mission.Value;
            string cause = string.Empty;
            bool violationDetected = missionValue switch
            {
                "timetable" => !CheckTT003Timetable(blockPart, ref
cause),
                "fullRun" or "emptyRun" =>
!CheckTT003FullEmptyRun(blockPart, ref cause),
                _ => !CheckTT003Other(blockPart, ref cause)
            };

            if (violationDetected)
            {
                int lineNo = GetLineNumber(blockPart);
                validationResult.AddSemanticWarning(lineNo,
$"Violation of constraint TT:003 in blockPart id =
{blockPart.Attribute("id").Value}. " +
                    $"Reason: {cause} when 'mission' is
{missionValue}");
            }
        }
    }

    private bool CheckTT003Other(XElement blockPart, ref string
cause)
    {
        //trainPartRef must not be set
        var trainPartRef = blockPart.Attribute("trainPartRef");
        if (trainPartRef != null)
        {
            cause = "'trainPartRef' should not be set";
            return false;
        }

        //startOcpRef and endOcpRef must be set and must be the
same
        var startOcpRef = blockPart.Attribute("startOcpRef");
        var endOcpRef = blockPart.Attribute("endOcpRef");
        if (startOcpRef == null || endOcpRef == null ||
startOcpRef.Value != endOcpRef.Value)
        {
            cause = "'startOcpRef' and 'endOcpRef' must be set and
must be the same";
            return false;
        }

        //begin and end must be set
        var begin = blockPart.Attribute("begin");
        var end = blockPart.Attribute("end");
        if (begin == null || end == null)
        {
            cause = "'begin' and 'end' must be set";
            return false;
        }

        return true;
    }

    private bool CheckTT003FullEmptyRun(XElement blockPart,
ref string cause)
    {
        var trainPartRef = blockPart.Attribute("trainPartRef");
        if (trainPartRef != null)
        {
            cause = "'trainPartRef' should not be set";
            return false;
        }

        //for this, startOcpRef and endOcpRef must be set
        var startOcpRef = blockPart.Attribute("startOcpRef");
        var endOcpRef = blockPart.Attribute("endOcpRef");
        if (startOcpRef == null || endOcpRef == null)
        {
            cause = "'startOcpRef' and 'endOcpRef' must be set";
            return false;
        }

        //begin and end need to be set as well
        var begin = blockPart.Attribute("begin");
        var end = blockPart.Attribute("end");
        if (begin == null || end == null)
        {
            cause = "'begin' and 'end' must be set";
            return false;
        }

        return true;
    }

    private bool CheckTT003Timetable(XElement blockPart, ref
string cause)
    {
        var trainPartRef = blockPart.Attribute("trainPartRef");
        if (trainPartRef == null)
        {
            cause = "'trainPartRef' should be set";
            return false;
        }

        //now we need to check if startOcpRef and endOcpRef are not
set or if they are, if they are without contradiction to the referenced
trainPart
        var startOcpRef = blockPart.Attribute("startOcpRef");
        var endOcpRef = blockPart.Attribute("endOcpRef");
        //same goes for beginDay and endDay, begin and end time
        var beginDay = blockPart.Attribute("beginDay");
```

```
        var endDay = blockPart.Attribute("endDay");
        var beginTime = blockPart.Attribute("begin");
        var endTime = blockPart.Attribute("end");
        //last but not least runLength needs to be checked
        var runLength = blockPart.Attribute("runLength");

        if (startOcpRef == null && endOcpRef == null &&
beginDay == null && endDay == null && beginTime == null &&
endTime == null)
        {
            return true;
        }

        //if they are specified, they must not contradict the referenced
trainPart
        var trainParts = xDocument
            .Descendants(ns + "timetable")
            .Descendants(ns + "trainPart")
            .Where(tp => (string)tp.Attribute("id") ==
trainPartRef.Value)
            .ToList();

        if (!trainParts.Any())
        {
            throw new InvalidOperationException($"trainPart
{trainPartRef.Value} referenced by blockPart
{blockPart.Attribute("id").Value} not found");
        }

        var trainPart = trainParts.First();
        var trainPartOcpTTs = trainPart
            .Descendants(ns + "ocpsTT")
            .Descendants(ns + "ocpTT")
            .ToList();

        var trainPartStart = trainPartOcpTTs.First();
        var trainPartEnd = trainPartOcpTTs.Last();
        var isEndPass = trainPartEnd.Attribute("ocpType")?.Value
== "pass";

        if (startOcpRef != null &&
trainPartStart.Attribute("ocpRef")?.Value != startOcpRef.Value)
        {
            cause = "'startOcpRef' must be empty or without
contradiction to referenced trainPart";
            return false;
        }

        if (endOcpRef != null &&
trainPartEnd.Attribute("ocpRef")?.Value != endOcpRef.Value)
        {
            cause = "'endOcpRef' must be empty or without
contradiction to referenced trainPart";
            return false;
        }

        var trainPartBeginTime = trainPartStart
            .Descendants(ns + "times")
            .FirstOrDefault(t => (string)t.Attribute("scope") ==
"scheduled");

        var trainPartEndTime = trainPartEnd
            .Descendants(ns + "times")
            .FirstOrDefault(t => (string)t.Attribute("scope") ==
"scheduled");

        if (trainPartBeginTime != null && beginTime != null &&
trainPartBeginTime.Attribute("departure")?.Value !=
beginTime.Value)
        {
            cause = "'begin' must be empty or without contradiction to
referenced trainPart";
            return false;
        }

        string compareValue;
        if (isEndPass)
        {
            compareValue =
trainPartEndTime?.Attribute("departure")?.Value;
        }
        else
        {
            compareValue =
trainPartEndTime?.Attribute("arrival")?.Value;
        }

        if (trainPartEndTime != null && endTime != null &&
compareValue != endTime.Value)
        {
            cause = "'end' must be empty or without contradiction to
referenced trainPart";
            return false;
        }

        //check of operating days is not included as it is far too
complicated.
        //runLength is not checked as it is not clear what it should be
checked against

        return true;
    }

    private void CheckTT005()
    {
        CheckFormationRefAndVehicleRef("TT:005", "rostering",
"timetable");
    }

    private void CheckTT004()
    {
        CheckFormationRefAndVehicleRef("TT:004", "blockPart",
"timetable");
    }

    private void CheckFormationRefAndVehicleRef(string
constraintName, string element, string type)
    {
        var blockParts = xDocument
            .Descendants(ns + type)
            .Descendants(ns + element)
            .ToList();

        foreach (var blockPart in blockParts)
        {
            var vehicleRef = blockPart.Attribute("vehicleRef");
            var formationRef = blockPart.Attribute("formationRef");

            if (vehicleRef != null && formationRef != null)
            {
                var blockPartId = blockPart.Attribute("id")?.Value;
                int lineNo = GetLineNumber(blockPart);
                validationResult.AddSemanticWarning(lineNo,
$"Violation of constraint {constraintName} in {element}, id =
{blockPartId}.");
            }
        }
    }

    private void CheckTT009()
    {
        var ocpTTs = xDocument
            .Descendants(ns + "timetable")
            .Descendants(ns + "ocpTT")
            .Where(ocpTT => ocpTT
            .Descendants(ns + "stopDescription")
            .Elements(ns + "trackInfo")
            .Any(trackInfo =>
trackInfo.Attribute("operatingPeriodRef") != null))
            .ToList();
```

```
        var operatingPeriods = xDocument
            .Descendants(ns + "timetable")
            .Elements(ns + "operatingPeriods")
            .ToList();

        if (!operatingPeriods.Any()) return; //nothing to check

        var operatingPeriodsNode = operatingPeriods.First();

        foreach (var ocpTT in ocpTTs)
        {
            var trainPart = ocpTT.Parent?.Parent;
            var operatingPeriodRefNode = trainPart?.Elements(ns +
"operatingPeriodRef").FirstOrDefault();

            string trainPartBitMask = "";
            if (operatingPeriodRefNode != null)
            {
                var reference =
operatingPeriodRefNode.Attribute("ref")?.Value;
                var operatingPeriod = operatingPeriodsNode
                    .Descendants(ns + "operatingPeriod")
                    .FirstOrDefault(op => op.Attribute("id")?.Value ==
reference);

                trainPartBitMask =
DetermineBitmaskOfOperatingPeriod(operatingPeriod);
                if (trainPartBitMask == null) continue; //no need to
check
            }

            var trackInfos = ocpTT
                .Descendants(ns + "stopDescription")
                .Elements(ns + "trackInfo")
                .Where(trackInfo =>
trackInfo.Attribute("operatingPeriodRef") != null)
                .ToList();

            var listOfTrackInfoBitmask = new List<string>();
            foreach (var trackInfo in trackInfos)
            {
                var operatingPeriodRef =
trackInfo.Attribute("operatingPeriodRef");
                var operatingPeriod = operatingPeriodsNode
                    .Descendants(ns + "operatingPeriod")
                    .FirstOrDefault(op => op.Attribute("id")?.Value ==
operatingPeriodRef?.Value);

                if (operatingPeriod == null) continue; //no need to check

                var trackInfoBitmask =
DetermineBitmaskOfOperatingPeriod(operatingPeriod);
                if (trackInfoBitmask == null) continue; //no need to
check

                if (trackInfoBitmask.Length !=
trainPartBitMask.Length)
                {
                    int lineNo = GetLineNumber(trackInfo);
                    validationResult.AddSemanticWarning(lineNo,
$"Violation of constraint TT:009 in trainPart id =
{ExtractId(trainPart)}.");
                    continue;
                }

                for (int k = 0; k < trackInfoBitmask.Length; k++)
                {
                    if (trackInfoBitmask[k] == '1' &&
trainPartBitMask[k] == '0')
                    {
                        int lineNo = GetLineNumber(trackInfo);
                        validationResult.AddSemanticWarning(lineNo,
$"Violation of constraint TT:009 in trainPart id =
{ExtractId(trainPart)}.");
                        break;
                    }
                }

                listOfTrackInfoBitmask.Add(trackInfoBitmask);
            }

            //Check for overlapping of the trackinfo bitmasks
            for (int j = 0; j < listOfTrackInfoBitmask.Count; j++)
            {
                for (int k = j + 1; k < listOfTrackInfoBitmask.Count;
k++)
                {
                    var bitmask1 = listOfTrackInfoBitmask[j];
                    var bitmask2 = listOfTrackInfoBitmask[k];

                    for (int l = 0; l < bitmask1.Length; l++)
                    {
                        if (bitmask1[l] == '1' && bitmask2[l] == '1')
                        {
                            int lineNo = GetLineNumber(trainPart);
                            validationResult.AddSemanticWarning(lineNo,
$"Violation of constraint TT:009 in trainPart id =
{ExtractId(trainPart)}.");
                            break;
                        }
                    }
                }
            }
        }

    }

    private void CheckTT015()
    {
        //create lookuptable of trainParts
        Dictionary<string, XElement> trainPartLookup = new
Dictionary<string, XElement>();
        {
            var trainParts = xDocument.Descendants(ns +
"timetable").Descendants(ns + "trainPart").ToList();
            foreach (var trainPart in trainParts)
            {
                trainPartLookup.Add(trainPart.Attribute("id").Value,
trainPart);
            }
        }
        var trains = xDocument.Descendants(ns +
"timetable").Descendants(ns + "train").ToList();
        foreach (var train in trains)
        {
            var trainPartSequence = train.Descendants(ns +
"trainPartSequence").ToList();
            if (trainPartSequence.Count() < 2)
                continue; //nothing to do here
            var previousTrainPartRefs =
trainPartSequence.First().Descendants(ns + "trainPartRef").Select(tp
=> tp.Attribute("ref").Value).ToList();
            var previousTrainParts = previousTrainPartRefs.Select(tp
=> trainPartLookup[tp]).ToList();
            foreach (var tps in trainPartSequence.Skip(1))
            {
                var currentTrainPartRefs = tps.Descendants(ns +
"trainPartRef").Select(tp => tp.Attribute("ref").Value).ToList();
                var currentTrainParts = currentTrainPartRefs.Select(tp
=> trainPartLookup[tp]).ToList();
                foreach (var currentTrainPart in currentTrainParts)
                {
                    var firstOcpTT = currentTrainPart.Descendants(ns +
"ocpTT").FirstOrDefault();
                    foreach (var firstTimes in firstOcpTT.Descendants(ns
+ "times"))
                    {
                        if (firstTimes.Attribute("arrival") == null)
                        {
```

```
                        continue; // it is optional, so if its not given then
we dont need to check anything
                    }
                    var firstScope =
firstTimes.Attribute("scope")?.Value;
                    var previousTrainPartsRelevantTimes =
currentTrainParts.SelectMany(tp =>
                        tp.Descendants(ns +
"ocpTT").Last().Descendants(ns + "times")
                            .Where(t => t.Attribute("scope")?.Value ==
firstScope)).ToList();

                    if (!previousTrainPartsRelevantTimes.Any() &&
firstScope == "published")
                    {
                        previousTrainPartsRelevantTimes =
currentTrainParts.SelectMany(tp =>
                            tp.Descendants(ns +
"ocpTT").Last().Descendants(ns + "times")
                                .Where(t => t.Attribute("scope")?.Value ==
"scheduled")).ToList();
                    }
                    if (!previousTrainPartsRelevantTimes.Any())
                    {
                        continue; //no relevant times in next trainPart,
nothing to compare
                    }
                    var currentArrival =
firstTimes.Attribute("arrival").Value;
                    foreach (var previousTimes in
previousTrainPartsRelevantTimes)
                    {
                        var previousArrival =
previousTimes.Attribute("arrival").Value;
                        if (currentArrival != previousArrival)
                        {
                            int lineNo1 = GetLineNumber(firstTimes);
                            int lineNo2 =
GetLineNumber(previousTimes);
                            string message = $"Violation of constraint
TT:015 between trainPart id = {ExtractId(currentTrainPart)} " +
                                $"and trainPart id =
{ExtractId(previousTimes.Parent.Parent)}.";

validationResult.AddSemanticWarning(lineNo1, message);

validationResult.AddSemanticWarning(lineNo2, message);
                        }
                    }
                }
            }
            previousTrainParts = currentTrainParts;
        }
    }

    private void CheckTT016()
    {
        //create lookuptable of trainParts
        Dictionary<string, XElement> trainPartLookup = new
Dictionary<string, XElement>();
        {
            var trainParts = xDocument.Descendants(ns +
"timetable").Descendants(ns + "trainPart").ToList();
            foreach (var trainPart in trainParts)
            {
                trainPartLookup.Add(trainPart.Attribute("id").Value,
trainPart);
            }
        }
        var trains = xDocument.Descendants(ns +
"timetable").Descendants(ns + "train").ToList();
        foreach (var train in trains)
        {
            var trainPartSequence = train.Descendants(ns +
"trainPartSequence").ToList();
            if (trainPartSequence.Count() < 2)
                continue; //nothing to do here
            var currentTrainPartRefs =
trainPartSequence.First().Descendants(ns + "trainPartRef").Select(tp
=> tp.Attribute("ref").Value).ToList();
            var currentTrainParts = currentTrainPartRefs.Select(tp =>
trainPartLookup[tp]).ToList();
            foreach (var tps in trainPartSequence.Skip(1))
            {
                var nextTrainPartRefs = tps.Descendants(ns +
"trainPartRef").Select(tp => tp.Attribute("ref").Value).ToList();
                var nextTrainParts = nextTrainPartRefs.Select(tp =>
trainPartLookup[tp]).ToList();
                foreach (var currentTrainPart in currentTrainParts)
                {
                    var lastOcpTT = currentTrainPart.Descendants(ns +
"ocpTT").LastOrDefault();
                    foreach (var lastTimes in lastOcpTT.Descendants(ns +
"times"))
                    {
                        if(lastTimes.Attribute("departure") == null)
                        {
                            continue; // it is optional, so if its not given then
we dont need to check anything
                        }
                        var lastScope =
lastTimes.Attribute("scope")?.Value;
                        var nextTrainPartsRelevantTimes =
nextTrainParts.SelectMany(tp =>
                            tp.Descendants(ns +
"ocpTT").First().Descendants(ns + "times")
                                .Where(t => t.Attribute("scope")?.Value ==
lastScope)).ToList();
                        if(!nextTrainPartsRelevantTimes.Any() &&
lastScope == "published")
                        {
                            nextTrainPartsRelevantTimes =
nextTrainParts.SelectMany(tp =>
                                tp.Descendants(ns +
"ocpTT").First().Descendants(ns + "times")
                                    .Where(t => t.Attribute("scope")?.Value ==
"scheduled")).ToList();
                        }
                        if (!nextTrainPartsRelevantTimes.Any())
                        {
                            continue; //no relevant times in next trainPart,
nothing to compare
                        }
                        var currentDeparture =
lastTimes.Attribute("departure").Value;
                        foreach (var nextTimes in
nextTrainPartsRelevantTimes)
                        {
                            var nextDeparture =
nextTimes.Attribute("departure").Value;
                            if (currentDeparture != nextDeparture)
                            {
                                int lineNo1 = GetLineNumber(lastTimes);
                                int lineNo2 = GetLineNumber(nextTimes);
                                string message = $"Violation of constraint
TT:016 between trainPart id = {ExtractId(currentTrainPart)} " +
                                    $"and trainPart id =
{ExtractId(nextTimes.Parent.Parent)}.";

validationResult.AddSemanticWarning(lineNo1, message);

validationResult.AddSemanticWarning(lineNo2, message);
                            }
                        }
                    }
                }
                currentTrainParts = nextTrainParts;
            }
```

```
        }
    }

    private void CheckTT017()
    {
        //Getting all ocpTTs
        var ocpTTs = xDocument.Descendants(ns +
"timetable").Descendants(ns + "ocpTT").ToList();

        //Loop through each ocpTT, getting all connections and
checking if referenced trainPartRef and trainRef reference each other
        foreach (var ocpTT in ocpTTs)
        {
            var connections = ocpTT.Descendants(ns +
"connection").ToList();

            foreach (var connection in connections)
            {
                //Getting the id of the referenced trainPart
                var trainPartIDTarget =
connection.Attribute("trainPartRef")?.Value;

                if(trainPartIDTarget != null)
                {
                    //Getting the id of the referenced train
                    var trainID = connection.Attribute("trainRef")?.Value;

                    //If there is no train referenced, it's a violation
                    if (trainID == null)
                    {
                        int lineNo = GetLineNumber(connection);
                        XElement trainPart = ocpTT.Parent?.Parent;
                        validationResult.AddSemanticWarning(lineNo,
$"Violation of constraint TT:017 in " +
                            $"trainPart id = {ExtractId(trainPart)}.
<connection> has @trainPartRef, but no @trainRef.");
                        continue;
                    }

                    //Getting the actual train that was referenced
                    XElement train = xDocument.Descendants(ns +
"train")
                      .FirstOrDefault(e => (string)e.Attribute("id") ==
trainID);

                    //Getting all the referenced train parts in the train
                    var trainPartRefs = train.Descendants(ns +
"trainPartRef").ToList();
                    bool trainPartFound = false;

                    foreach(var trainPartRefElement in trainPartRefs)
                    {
                        var trainPartID =
trainPartRefElement.Attribute("ref")?.Value;
                        if(trainPartID == trainPartIDTarget)
                        {
                            trainPartFound = true;
                        }
                    }
                    if (!trainPartFound)
                    {
                        int lineNo_ = GetLineNumber(connection);
                        XElement trainPart_ = ocpTT.Parent?.Parent;
                        validationResult.AddSemanticWarning(lineNo_,
$"Violation of constraint TT:017 in " +
                            $"trainPart id = {ExtractId(trainPart_)}.
<trainPart> referenced by @trainPartRef in <connection> " +
                            $"doesn't reference same <train> as referenced in
@trainRef.");
                    }
                }
            }

        }

    }

    private void CheckIS015()
    {
        //Getting all ocps parts
        var ocpsWithParent = xDocument.Descendants(ns +
"infrastructure").Descendants(ns + "ocp")
            .Where(e => e.Attribute("parentOcpRef") != null).ToList();
        HashSet<string> foundCircles = new HashSet<string>();

        foreach(XElement ocp in ocpsWithParent)
        {
            bool checkedAllParents = false;
            string ocpID = ocp.Attribute("id")?.Value;
            var ocpcurrent = ocp;
            do
            {
                //Getting the id of the parent ocp
                var parentOcpID =
ocpcurrent.Attribute("parentOcpRef")?.Value;
                if (parentOcpID != null)
                {
                    //Getting the actual parent ocp that was referenced
                    XElement parentOcp =
ocpsWithParent.FirstOrDefault(e => (string)e.Attribute("id") ==
parentOcpID);
                    if (parentOcp != null)
                    {
                        //If parent ocp also references some parent ocp,
make sure it is not descendant ocp that already references it to avoid
circle
                        var parent_parentOcpID =
parentOcp.Attribute("parentOcpRef")?.Value;
                        if (parent_parentOcpID != null)
                        {
                            if (ocpID == parent_parentOcpID)
                            {
                                //Checking if these ocps were not already
found in reversed order
                                string pairKey = $"{parentOcpID}-{ocpID}";
                                if (!foundCircles.Contains(pairKey) &&
!foundCircles.Contains(pairKey))
                                {
                                    pairKey = $"{ocpID}-{parentOcpID}";
                                    foundCircles.Add(pairKey);
                                    int lineNo1 = GetLineNumber(ocp);
                                    int lineNo2 = GetLineNumber(parentOcp);
                                    string message = $"Violation of constraint
IS:015 between <ocp> id = {ocpID} and <ocp> id =
{parentOcpID}.";

validationResult.AddSemanticWarning(lineNo1, message);

validationResult.AddSemanticWarning(lineNo2, message);
                                }

                                checkedAllParents = true;
                            }
                            else
                            {
                                ocpcurrent = parentOcp;
                            }
                        }

                    }else checkedAllParents = true;

                }

            } while (!checkedAllParents);

        }
    }
  }
}
```

1.12 Текст файлу ExtensionElement.cs модуля railVIVID.Validation.Extensions

```
using System;

namespace railVIVID.Validation.Extensions
{
   public class ExtensionElement
   {
      public string Name { get; set; }
      public int Position { get; set; }
      public bool Validated { get; set; }

      public ExtensionElement(string name, int position, bool
validated)
      {
         Name = name;
         Position = position;
         Validated = validated;
      }

      public override bool Equals(object obj)
      {
         if (obj is ExtensionElement other)
         {
            return this.Name == other.Name &&
                   this.Position == other.Position;
         }
         return false;
      }

      public override int GetHashCode()
      {
         return HashCode.Combine(Name, Position);
      }
   }
}
```

1.13 Текст файлу railMLExtensions.cs модуля railVIVID.Validation.Extensions

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml;
using System.Xml.Linq;

namespace railVIVID.Validation.Extensions
{
   /// <summary>
   /// Class for finding all extension schemas and their
elements/attributes
   /// </summary>
   public class railMLExtensions
   {
      public event EventHandler<int> ProgressUpdated;
      public Dictionary<string, Tuple<string,
HashSet<ExtensionElement>>> Extensions { get; private set; }


      private List<string> StandardNamespaces = new List<string>()
                {
                   "http://purl.org/dc/elements/1.1/",
                   "http://www.w3.org/XML/1998/namespace",
                   "http://www.w3.org/1998/Math/MathML",
                   "http://www.w3.org/2001/XMLSchema-instance"
                };

      /// <summary>
      /// Finds and adds to Extensions list all the attributes/elements
that extend the railML file
      /// </summary>
      /// <param name="xDocument"></param>
      /// <param name="schemaLocations"></param>
      public void FindAndSetExtensions(XDocument xDocument,
Dictionary<string, string> schemaLocations)
      {
         SchemaControl schemaControl = new SchemaControl();

         if (schemaLocations == null)
         {
            schemaControl.SetAllSchemas(xDocument);
            schemaLocations = schemaControl.SchemaLocations;
         }

         Extensions = new Dictionary<string, Tuple<string,
HashSet<ExtensionElement>>>();
         Prevalidation prevalidation = new Prevalidation();

         var standardNamespaces = new
HashSet<string>(railMLNamespaces.StandardRailMLNamespaces.C
oncat(this.StandardNamespaces));

         //Get all namespaces in the document (elements and
attributes)
         var allNamespaces = xDocument.Root.DescendantsAndSelf()
            .Select(e => e.Name.NamespaceName)

.Concat(xDocument.Root.DescendantsAndSelf().Attributes()
               .Where(a => a.IsNamespaceDeclaration == false)
               .Select(a => a.Name.NamespaceName))
            .Distinct().Where(ns => !string.IsNullOrEmpty(ns));

         //Filter out the standard namespaces to identify extensions
         var extensionNamespaces = allNamespaces.Where(ns =>
!standardNamespaces.Contains(ns)).ToList();

         foreach (var ns in extensionNamespaces)
         {
            var schema = schemaLocations.ContainsKey(ns) ?
schemaLocations[ns] : null;

            var elements = xDocument.Root.DescendantsAndSelf()
            .Where(e => e.Name.NamespaceName == ns)
            .Select(e => new
            {
               Element = e,
               Name = e.Name.LocalName,
               LineNumber = (e as IXmlLineInfo).HasLineInfo() ? (e
as IXmlLineInfo).LineNumber : -1
            });

            var attributes = xDocument.Root.DescendantsAndSelf()
               .SelectMany(e => e.Attributes()
                  .Where(a => !a.IsNamespaceDeclaration &&
a.Name.NamespaceName == ns)
                  .Select(a => new
                  {
```

```
                    Attribute = a,
                    Name = a.Name.LocalName,
                    LineNumber = (e as IXmlLineInfo).HasLineInfo() ?
(e as IXmlLineInfo).LineNumber : -1
                }));

            int progressPercentage;
            int i = 1;
            int amount = elements.Count() + attributes.Count();
            foreach (var element in elements)
            {
                progressPercentage = (i * 100) / amount;
                ProgressUpdated?.Invoke(this, progressPercentage);
                i++;

                AddElementToList(ns, schema, element.Name,
element.LineNumber, true);
            }
            foreach (var attribute in attributes)
            {
                progressPercentage = (i * 100) / amount;
                ProgressUpdated?.Invoke(this, progressPercentage);
                i++;
                AddElementToList(ns, schema, attribute.Name,
attribute.LineNumber, true);
            }
        }
    }

    private void AddElementToList(string ns, string schema, string
elementName, int position, bool validated)
    {

        if (!Extensions.ContainsKey(ns))
        {
            Extensions[ns] = new Tuple<string,
HashSet<ExtensionElement>>(schema, new
HashSet<ExtensionElement>());
        }

        Extensions[ns].Item2.Add(new
ExtensionElement(elementName, position, validated));
    }

    private ExtensionElement GetElement(string ns, string
elementName, int position)
    {
        var matchingEntry = Extensions.FirstOrDefault(kvp =>
kvp.Key == ns);

        if (matchingEntry.Key != null)
        {
            foreach (var element in matchingEntry.Value.Item2)
            {
                if (element.Name == elementName &&
element.Position == position)
                {
                    return element;
                }
            }
        }

        return null;
    }

    public bool ChangeElementValidStatus(string ns, string name,
int position)
    {
        var element = GetElement(ns, name, position);

        if (element != null)
        {
            element.Validated = false;
            return true;
        }

        return false;
    }
  }
}
```

1.14 Текст файлу FileContents.cs модуля railVIVID.Validation.GeneralInformation

```
using System.Collections.Generic;
using System.Linq;
using System.Xml.Linq;

namespace railVIVID.Validation.GeneralInformation
{
  /// <summary>
  /// Class to store information about contents of the file (metadata,
infrastructure etc)
  /// </summary>
  public class FileContents
  {
    public List<string> Contents { get; private set; } = new
List<string>();

    /// <summary>
    /// Sets list of contents for the file
    /// </summary>
    /// <param name="railMLDocument"></param>
    public void LoadContent(XDocument railMLDocument)
    {
      XElement root = railMLDocument.Root;

      if (root != null)
      {
        var secondLevelElementNames = root.Elements()
                    .Select(e => e.Name.LocalName)
                    .ToList();

        foreach (var name in secondLevelElementNames)
        {
          Contents.Add(name);
        }
      }

    }
  }
}
```

1.15 Текст файлу Metadata.cs модуля railVIVID.Validation.GeneralInformation

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;

namespace railVIVID.Validation.GeneralInformation
{
   /// <summary>
   /// Class to store information about metadata of the file
   /// </summary>
   public class Metadata
   {
      public Dictionary<string, string> dataPairs { get; private set; } =
new Dictionary<string, string>();

      public void LoadMetadata(XDocument railMLDocument) {
         XNamespace dc = "http://purl.org/dc/elements/1.1/";

         AddValue("Format", railMLDocument.Descendants(dc +
"format").FirstOrDefault()?.Value);
         AddValue("DateTime", railMLDocument.Descendants(dc +
"date").FirstOrDefault()?.Value);
         AddValue("Identifier", railMLDocument.Descendants(dc +
"identifier").FirstOrDefault()?.Value);
         AddValue("Language", railMLDocument.Descendants(dc +
"language").FirstOrDefault()?.Value);
         AddValue("Source", railMLDocument.Descendants(dc +
"source").FirstOrDefault()?.Value);

         AddValue("Creator", railMLDocument.Descendants(dc +
"creator").FirstOrDefault()?.Value);

      }

      private void AddValue(string key, string value)
      {
         if(value != null)
         {
            dataPairs.Add(key, value);
         }
      }

   }
}
```

1.16 Текст файлу railMLDocument.cs модуля railVIVID.Validation.GeneralInformation

```
using System.Xml.Linq;

namespace railVIVID.Validation.GeneralInformation
{
   /// <summary>
   /// Class that stores the file as XDocument and it's path
   /// </summary>
   public class railMLDocument
   {
      public readonly XDocument document;
      public readonly string path;
      public readonly string[] lines;

      public railMLDocument() { }

      public railMLDocument(XDocument railMLDoc, string path,
string[] lines)
      {
         this.document = railMLDoc;
         this.path = path;
         this.lines = lines;
      }
   }
}
```

1.17 Текст файлу ValidationXDocument.cs модуля railVIVID.Validation.XDocument

```
using railVIVID.Validation.Extensions;
using railVIVID.Validation.GeneralInformation;
using System.Xml.Schema;

namespace railVIVID.Validation.XDocument
{
   /// <summary>
   /// This class is used for master's thesis of Vyskarka M.. Do not
delete.
   /// </summary>
   public class ValidationXDocument : IValidator
   {
      private ValidationResult result;
      private SchemaControl schemaControl = new SchemaControl();
      private railMLExtensions extensions = new railMLExtensions();
      private railMLVersion railMLVersion = new railMLVersion();

      public event Action<int> ValidationProgressChanged;
      public event Action<int> ExtensionsProgressRelayed;

      /// <summary>
      /// Partial validation. Uses XDocument.Validate.
      /// </summary>
      /// <returns></returns>
      public Tuple<ValidationResult, railMLExtensions>
Validate(railMLDocument railMLDocument, railMLVersion
railMLVersion)
      {
         if (railMLDocument.document == null)
         {
            result.AddError("Document is empty or not properly
loaded!");
            return new Tuple<ValidationResult,
railMLExtensions>(result, null);
         }

         result = new ValidationResult(railMLDocument);
         Validation validation = new Validation();
         this.railMLVersion = railMLVersion;

schemaControl.SetDefaultLocationFromVersion(railMLVersion.Ver
sionFromNamespace);
```

```
        //Using custom resolver
        CustomXmlUrlResolver resolver = new
CustomXmlUrlResolver(result, railMLVersion);
        XmlSchemaSet schemaSet = new XmlSchemaSet();
        schemaSet.XmlResolver = resolver;

        try
        {
            validation.AddRailML(schemaSet, schemaControl,
railMLVersion);

            resolver._railMLAdded = true;

            //Setting valid non-railML schemas from schemaLocation

schemaControl.SetAndCheckAllSchemas(railMLDocument.docume
nt, result, railMLDocument.path);
            //Adding those schemas to the schema set
            schemaControl.AddOtherShemas(schemaSet);

            //Finding extensions to be able to validate them
            extensions.ProgressUpdated +=
OnExtensionsProgressUpdated;

extensions.FindAndSetExtensions(railMLDocument.document,
schemaControl.SchemaLocations);

            try
            {
                schemaSet.Compile();
            }
            catch (Exception ex)
            {
                result.AddError(ex.Message);
            }

            List<ValidationEventArgs> issues = new
List<ValidationEventArgs>();

            railMLDocument.document.Validate(schemaSet, (sender,
e) =>
            {
                issues.Add(e);

            });

            try
            {
                int i = 1;
                foreach (var issue in issues)
                {
                    OnProgressChanged(i * 100 / issues.Count);
                    i++;
                    validation.HandleIssues(issue, result, extensions);
                }
            }
            catch (Exception ex)
            {
                result.AddError(ex.Message);
            }

        }
        catch (Exception ex)
        {
            result.AddError(ex.Message);
        }

        return new Tuple<ValidationResult,
railMLExtensions>(result, extensions);
    }

    protected virtual void OnProgressChanged(int percent)
    {
        ValidationProgressChanged?.Invoke(percent);
    }

    private void OnExtensionsProgressUpdated(object sender, int
progressPercentage)
    {
        // Relay percentage progress update
        ExtensionsProgressRelayed?.Invoke(progressPercentage);
    }
  }
}
```

1.18 Текст файлу ValidationXDocumentSolution.cs модуля railVIVID.Validation.XDocumentSolution

```
using railVIVID.Validation.Extensions;
using railVIVID.Validation.GeneralInformation;
using railVIVID.Validation.SemanticValidation;
using System.Xml.Linq;
using System.Xml.Schema;

namespace railVIVID.Validation.XDocumentSolution
{
  public class ValidationXDocumentSolution : IValidator
  {
    private ValidationResult result;
    private SchemaControl schemaControl = new SchemaControl();
    private railMLExtensions extensions = new railMLExtensions();
    private railMLVersion railMLVersion = new railMLVersion();
    private AttributeHandling attributeHandling = new
AttributeHandling(); //Initialized here to initialize found elements
without attribute once
    private Validation validation = new Validation();

    public event Action<int> ValidationProgressChanged;
    public event Action<int> ExtensionsProgressRelayed;

    //Processes issues raised by validator (requires special handling
of non-railml related issues and unclear .NET messages)
    private void HandleIssues(ValidationEventArgs e,
XmlSchemaSet schemaSet, XDocument railMLDocument)
    {
        string namespaceUri = string.Empty;
        string objectName = string.Empty;
        if (e.Severity == XmlSeverityType.Warning)
        {
            result.AddWarning(e.Exception.LineNumber, e.Message);
        }
        else
        {
            //If the issue is undeclared element, it can appear in three
cases:
            //1. Element/attribute is missing a schema
            //2. Element/attribute is missing a declaration
            //3. Unknown attribute is used within element without
<xsi:any> or <xsi:anyAttribute> (Microsoft .NET issue)
            if (e.Exception.Message.Contains("cannot find the
declaration") ||
            e.Exception.Message.Contains("attribute is not declared") ||
            e.Exception.Message.Contains("element is not declared"))
```

```
            {
                namespaceUri =
validation.ExtractNamespaceFromErrorMessage(e.Message);
                objectName =
validation.ExtractObjectNameFromErrorMessage(e.Message);
                string elementName;

                if (e.Exception.Message.Contains("attribute is not
declared"))
                {
                    bool elementFound;
                    bool isAttributePossible =
attributeHandling.IsAttributePossible(railMLDocument, schemaSet,
objectName,
                        namespaceUri, e.Exception.LineNumber, out
elementName, out elementFound);

                    if (elementFound == false)
                    {
                        result.AddWarning(e.Exception.LineNumber,
                            $"Not possible to find element '{elementName}
related to attribute '{objectName}' in the schema. " +
                            $"Attribute is undeclared, reason is unknown.");
                    }
                    else
                    {
                        //Checking if the issue is due to element not having
any attributes defined
                        if (!isAttributePossible)
                        {
                            result.AddError(e.Exception.LineNumber,
$"Attribute '{objectName}' namespace '{namespaceUri}' is not
allowed to be " +
                                $"in '{elementName}'. This element has no
attributes.");
                        }
                        else
                        {
                            DeclarationIssue(e, objectName, namespaceUri);
                        }
                    }

                }
                else DeclarationIssue(e, objectName, namespaceUri);
            }
            else
            {
                result.AddError(e.Exception.LineNumber, e.Message);
            }
        }

        if (namespaceUri == string.Empty || objectName ==
string.Empty)
        {
            namespaceUri =
validation.ExtractNamespaceFromType1(e.Message);
            objectName =
validation.ExtractElementNameFromType1(e.Message);

            if (!extensions.ChangeElementValidStatus(namespaceUri,
objectName, e.Exception.LineNumber))
            {
                string namespaceElement =
validation.ExtractNamespaceElementPairFromType2(e.Message);
                if (namespaceElement != null)
                {
                    namespaceUri =
validation.ExtractNamespaceFromErrorMessage(namespaceElement
);
                    objectName =
validation.ExtractObjectNameFromErrorMessage(namespaceElemen
t);
                    extensions.ChangeElementValidStatus(namespaceUri,
objectName, e.Exception.LineNumber);
                }
            }
            else
            {
                extensions.ChangeElementValidStatus(namespaceUri,
objectName, e.Exception.LineNumber);
            }
        }

        private void DeclarationIssue(ValidationEventArgs e, string
attributeName, string namespaceUri)
        {
            if (!string.IsNullOrEmpty(namespaceUri))
            {
                string description = "Elements from this namespace are
missing a declaration: " + namespaceUri
                    + ". A schema/declaration of an element inside schema
is missing or the referenced schema is invalid. Such elements were
not validated against related schema.";

                result.AddWarning(e.Exception.LineNumber, description);

                extensions.ChangeElementValidStatus(namespaceUri,
attributeName, e.Exception.LineNumber);

            }
            else
            {
                result.AddWarning(e.Exception.LineNumber, e.Message);
            }
        }

        /// <summary>
        /// Full validation. Uses XDocument.Validate and supplemented
manual check of attributes.
        /// </summary>
        /// <param name="railMLDocument"></param>
        /// <param name="railMLVersion"></param>
        /// <returns></returns>
        public Tuple<ValidationResult, railMLExtensions>
Validate(railMLDocument railMLDocument, railMLVersion
railMLVersion)
        {
            if (railMLDocument.document == null)
            {
                result.AddError("Document is empty or not properly
loaded!");
                return new Tuple<ValidationResult,
railMLExtensions>(result, null);
            }

            result = new ValidationResult(railMLDocument);
            this.railMLVersion = railMLVersion;

schemaControl.SetDefaultLocationFromVersion(railMLVersion.Ver
sionFromNamespace);

            //Using custom resolver
            CustomXmlUrlResolver resolver = new
CustomXmlUrlResolver(result, railMLVersion);
            XmlSchemaSet schemaSet = new XmlSchemaSet();
            schemaSet.XmlResolver = resolver;

            try
            {
                validation.AddRailML(schemaSet, schemaControl,
railMLVersion);

                resolver._railMLAdded = true;

                //Setting valid non-railML schemas from schemaLocation

schemaControl.SetAndCheckAllSchemas(railMLDocument.docume
nt, result, railMLDocument.path);
```

```
            //Adding those schemas to the schema set
            schemaControl.AddOtherShemas(schemaSet);

            //Finding extensions to be able to validate them
            extensions.ProgressUpdated +=
OnExtensionsProgressUpdated;

extensions.FindAndSetExtensions(railMLDocument.document,
schemaControl.SchemaLocations);

            try
            {
               schemaSet.Compile();
            }
            catch (Exception ex)
            {
               result.AddError(ex.Message);
            }

            List<ValidationEventArgs> issues = new
List<ValidationEventArgs>();

            railMLDocument.document.Validate(schemaSet, (sender,
e) =>
            {
               issues.Add(e);

            });

            try
            {
               int i = 1;
               foreach (var issue in issues)
               {
                  OnProgressChanged(i * 100 / issues.Count);
                  i++;
                  HandleIssues(issue, schemaSet,
railMLDocument.document);
               }
            }
            catch (Exception ex)
            {
               result.AddError(ex.Message);
            }

            SemanticValidator semanticValidator = new
SemanticValidator();
            semanticValidator.Validate(railMLDocument.document,
ref result);
         }
         catch (Exception ex)
         {
            result.AddError(ex.Message);
         }

         return new Tuple<ValidationResult,
railMLExtensions>(result, extensions);
      }

      protected virtual void OnProgressChanged(int percent)
      {
         ValidationProgressChanged?.Invoke(percent);
      }

      private void OnExtensionsProgressUpdated(object sender, int
progressPercentage)
      {
         // Relay percentage progress update
         ExtensionsProgressRelayed?.Invoke(progressPercentage);
      }
   }
}
```

1.19 Текст файлу AttributeHandling.cs модуля railVIVID.Validation.XDocumentSolution

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;
using System.Xml.Schema;
using System.Xml;
using System.Text.RegularExpressions;


namespace railVIVID.Validation.XDocumentSolution
{
   /// <summary>
   /// Solution to check prohibited attributes (from message about
'undeclared attributes')
   /// </summary>
   internal class AttributeHandling
   {
      private HashSet<XmlSchemaElement> foundElements = new
HashSet<XmlSchemaElement>();


      public bool IsAttributePossible(XDocument xDocument,
XmlSchemaSet schemaSet, string attributeName,
         string attributeNamespace, int lineNumber, out string
elementName, out bool found)
      {
         XElement attribute = xDocument.Descendants()
            .Where(e => e.Attributes()
            .Any(a => a.Name.LocalName == attributeName &&
            (a.Name.NamespaceName == attributeNamespace ||
string.IsNullOrEmpty(a.Name.NamespaceName)))
            && ((IXmlLineInfo)e).HasLineInfo() &&
((IXmlLineInfo)e).LineNumber == lineNumber)
            .FirstOrDefault();


         elementName = attribute.Name.LocalName;
         string elementNamespace = attribute.Name.NamespaceName;
```

```
        found = false;
        XmlSchemaElement element = null;

        //Check if such element was already found for other attributes
        foreach (XmlSchemaElement _element in foundElements)
        {
            if (_element.Name == elementName)
            {
                element = _element;
                break;
            }
        }

        if (element != null)
        {
            found = true;
            return HasAny(element, schemaSet);
        }
        else
        {
            element = FindElementInSchemas(schemaSet, elementName, elementNamespace);

            if (element == null)
            {
                return false;
            }
            else
            {
                foundElements.Add(element);
                found = true;
                return HasAny(element, schemaSet);
            }
        }

    }

    private bool HasAny(XmlSchemaElement element, XmlSchemaSet schemaSet)
    {
        if (element.ElementType is XmlSchemaComplexType complexType)
        {
            return HasAny(complexType, schemaSet);
        }

        return false;
    }

    private bool HasAny(XmlSchemaComplexType complexType, XmlSchemaSet schemaSet)
    {
        bool hasAny = false;

        if (complexType.AnyAttribute != null)
        {
            hasAny = true;
        }

        //Check for <xsi:any>
        if (complexType.ContentTypeParticle is XmlSchemaGroupBase groupBase
            && ContainsAnyElement(groupBase))
        {
            hasAny = true;
        }

        //In-depth search for <xsi:anyAttribute>
        if (!hasAny)
        {
            foreach (XmlSchemaObject attributeObj in complexType.Attributes)
            {
                if (attributeObj is XmlSchemaAttributeGroupRef groupRef)
                {
                    var attributeGroup = FindAttributeGroup(groupRef.RefName, schemaSet);
                    if (attributeGroup != null && HasAnyAttribute(attributeGroup))
                    {
                        hasAny = true;
                        break;
                    }
                }
            }
        }

        //If the complex type extends another type, check the base type
        if (!hasAny &&
            complexType.ContentModel is XmlSchemaComplexContent complexContent &&
            complexContent.Content is XmlSchemaComplexContentExtension extension)
        {
```

```
            //Find the base type (which could be in another schema)
            XmlSchemaType baseType = FindSchemaType(extension.BaseTypeName, schemaSet);

            if (baseType is XmlSchemaComplexType baseComplexType)
            {
               //Recursively check the base type for attributes
               return HasAny(baseComplexType, schemaSet);
            }
         }

         return hasAny;
      }

      private bool HasAnyAttribute(XmlSchemaAttributeGroup attributeGroup)
      {
         //Check if the attribute group contains <xs:anyAttribute>
         if (attributeGroup.AnyAttribute != null)
         {
            return true;
         }

         return false;
      }

      private XmlSchemaAttributeGroup FindAttributeGroup(XmlQualifiedName groupName, XmlSchemaSet schemaSet)
      {
         foreach (XmlSchema schema in schemaSet.Schemas())
         {
            if (schema.AttributeGroups.Contains(groupName))
            {
               return schema.AttributeGroups[groupName] as XmlSchemaAttributeGroup;
            }
         }
         return null;
      }

      private bool ContainsAnyElement(XmlSchemaGroupBase groupBase)
      {
         //Check for <xs:any> in group base
         foreach (var item in groupBase.Items)
         {
            if (item is XmlSchemaAny)
            {
               return true;
            }

            //If it's another group base, recursively check its items
            if (item is XmlSchemaGroupBase nestedGroupBase)
            {
               if (ContainsAnyElement(nestedGroupBase))
               {
                  return true;
               }
            }
         }

         return false;
      }

      private static XmlSchemaType FindSchemaType(XmlQualifiedName baseTypeName, XmlSchemaSet schemaSet)
      {
         foreach (XmlSchema schema in schemaSet.Schemas())
         {
            if (schema.SchemaTypes.Contains(baseTypeName))
            {
               return schema.SchemaTypes[baseTypeName] as XmlSchemaType;
            }
         }
         return null;
      }

      private XmlSchemaElement FindElementInSchemas(XmlSchemaSet schemaSet, string elementName, string elementNamespace)
      {
         foreach (XmlSchema schema in schemaSet.Schemas())
         {
            string schemaTargetNamespace = schema.TargetNamespace;
            if (schemaTargetNamespace == elementNamespace)
            {
               //Check directly
               var simpleElement = schema.Items.OfType<XmlSchemaElement>()
                  .FirstOrDefault(e => e.Name == elementName);

               if (simpleElement != null)
               {
```

```
                return simpleElement;
            }

            foreach (XmlSchemaElement element in schema.Items.OfType<XmlSchemaElement>())
            {
                //Check if the element name matches
                if (element.Name == elementName)
                {
                    return element;
                }

                //Check its type if it's a complex type
                if (element.ElementType is XmlSchemaComplexType complexType)
                {
                    HashSet<XmlSchemaComplexType> visitedTypes = new HashSet<XmlSchemaComplexType>();
                    var foundElement = SearchComplexType(complexType, elementName, schemaSet, visitedTypes);

                    if (foundElement != null)
                    {
                        return foundElement; //Return the found nested element
                    }
                }
            }

        }
    }
    return null;
}

private static XmlSchemaElement SearchComplexType(XmlSchemaComplexType complexType, string elementName, XmlSchemaSet schemaSet, HashSet<XmlSchemaComplexType> visitedTypes)
{

    if (complexType == null)
    {
        return null;
    }

    if (visitedTypes.Contains(complexType))
    {
        return null; //To prevent infinite recursion
    }

    visitedTypes.Add(complexType);

    //Get the content model of the complex type
    var contentModel = complexType.ContentTypeParticle;

    //If it's a sequence, iterate through the items
    if (contentModel is XmlSchemaGroupBase groupBase)
    {
        foreach (var item in groupBase.Items)
        {
            //If the item is an element
            if (item is XmlSchemaElement childElement)
            {
                var childElem = childElement;

                //Handle elements with a "ref" attribute
                if (!string.IsNullOrEmpty(childElement.RefName.Name))
                {
                    var referencedElement = schemaSet.GlobalElements[
                        new XmlQualifiedName(childElement.RefName.Name, childElement.RefName.Namespace)] as XmlSchemaElement;

                    if (referencedElement != null)
                    {
                        childElem = referencedElement;
                    }
                }

                if (childElem.Name == elementName)
                {
                    return childElem;
                }

                //Recursively search in child elements if they are complex types
                if (childElem.ElementSchemaType is XmlSchemaComplexType childComplexType)
                {
                    var foundElement = SearchComplexType(childComplexType, elementName, schemaSet, visitedTypes);
                    if (foundElement != null)
                    {
                        return foundElement;
```

```
                }
            }
        }
    }
}
```

```
        return null; // Return null if not found
    }

}
}
```

1.20 Текст файлу ValidationXMLDocument.cs модуля railVIVID.Validation.XMLDocument

```
using railVIVID.Validation.Extensions;
using railVIVID.Validation.GeneralInformation;
using System.Xml;
using System.Xml.Schema;

namespace railVIVID.Validation.XMLDocument
{
   /// <summary>
   /// This class is used for master's thesis of Vyskarka M.. Do not
delete.
   /// </summary>
   public class ValidationXMLDocument : IValidator
   {
      private ValidationResult result;
      private SchemaControl schemaControl = new SchemaControl();
      private railMLExtensions extensions = new railMLExtensions();
      private railMLVersion railMLVersion = new railMLVersion();

      public event Action<int> ValidationProgressChanged;
      public event Action<int> ExtensionsProgressRelayed;

      /// <summary>
      /// Partial validation. Uses XMLDocument.Validate.
      /// </summary>
      /// <returns></returns>
      public Tuple<ValidationResult, railMLExtensions>
Validate(railMLDocument railMLDocument, railMLVersion
railMLVersion)
      {
         if (railMLDocument.document == null)
         {
            result.AddError("Document is empty or not properly
loaded!");
            return new Tuple<ValidationResult,
railMLExtensions>(result, null);
         }

         result = new ValidationResult(railMLDocument);
         Validation validation = new Validation();
         this.railMLVersion = railMLVersion;

schemaControl.SetDefaultLocationFromVersion(railMLVersion.Ver
sionFromNamespace);

         //Using custom resolver
         CustomXmlUrlResolver resolver = new
CustomXmlUrlResolver(result, railMLVersion);
         XmlSchemaSet schemaSet = new XmlSchemaSet();
         schemaSet.XmlResolver = resolver;

         try
         {
            validation.AddRailML(schemaSet, schemaControl,
railMLVersion);

            resolver._railMLAdded = true;

            //Setting valid non-railML schemas from schemaLocation

schemaControl.SetAndCheckAllSchemas(railMLDocument.docume
nt, result, railMLDocument.path);
            //Adding those schemas to the schema set
            schemaControl.AddOtherShemas(schemaSet);

            //Finding extensions to be able to validate them
            extensions.ProgressUpdated +=
OnExtensionsProgressUpdated;

extensions.FindAndSetExtensions(railMLDocument.document,
schemaControl.SchemaLocations);

            try
            {
               schemaSet.Compile();
            }
            catch (Exception ex)
            {
               result.AddError(ex.Message);
            }

            XmlDocument xmlDocument = new XmlDocument();
            xmlDocument.Load(railMLDocument.path);

            xmlDocument.Schemas.Add(schemaSet);

            List<ValidationEventArgs> issues = new
List<ValidationEventArgs>();
            xmlDocument.Validate((sender, e) =>
            {
               issues.Add(e);

            });

            try
            {
               int i = 1;
               foreach (var issue in issues)
               {
                  OnProgressChanged(i * 100 / issues.Count);
                  i++;
                  validation.HandleIssues(issue, result, extensions);
               }
            }
            catch (Exception ex)
            {
               result.AddError(ex.Message);
            }

         }
         catch (Exception ex)
         {
            result.AddError(ex.Message);
         }
```

```
        return new Tuple<ValidationResult,
railMLExtensions>(result, extensions);
    }

    protected virtual void OnProgressChanged(int percent)
    {
        ValidationProgressChanged?.Invoke(percent);
    }

    private void OnExtensionsProgressUpdated(object sender, int
progressPercentage)
    {
        // Relay percentage progress update
        ExtensionsProgressRelayed?.Invoke(progressPercentage);
    }
  }
}
```

1.21 Текст файлу ValidationXMLReader.cs модуля railVIVID.Validation.XMLReader

```
using System;
using railVIVID.Validation.Extensions;
using railVIVID.Validation.GeneralInformation;
using System.Xml;
using System.Xml.Schema;

namespace railVIVID.Validation.XMLReader
{
  /// <summary>
  /// This class is used for master's thesis of Vyskarka M.. Do not
delete.
  /// </summary>
  public class ValidationXMLReader : IValidator
  {
    private ValidationResult result;
    private SchemaControl schemaControl = new SchemaControl();
    private railMLExtensions extensions = new railMLExtensions();
    private railMLVersion railMLVersion = new railMLVersion();
    List<ValidationEventArgs> issues = new
List<ValidationEventArgs>();

    public event Action<int> ValidationProgressChanged;
    public event Action<int> ExtensionsProgressRelayed;

    /// <summary>
    /// Partial validation. Uses XMLReader.Create.
    /// </summary>
    /// <returns></returns>
    public Tuple<ValidationResult, railMLExtensions>
Validate(railMLDocument railMLDocument, railMLVersion
railMLVersion)
    {
      if (railMLDocument.document == null)
      {
        result.AddError("Document is empty or not properly
loaded!");
        return new Tuple<ValidationResult,
railMLExtensions>(result, null);
      }

      result = new ValidationResult(railMLDocument);
      Validation validation = new Validation();
      this.railMLVersion = railMLVersion;

schemaControl.SetDefaultLocationFromVersion(railMLVersion.Ver
sionFromNamespace);

      //Using custom resolver
      CustomXmlUrlResolver resolver = new
CustomXmlUrlResolver(result, railMLVersion);
      XmlSchemaSet schemaSet = new XmlSchemaSet();
      schemaSet.XmlResolver = resolver;


      try
      {
        validation.AddRailML(schemaSet, schemaControl,
railMLVersion);

        resolver._railMLAdded = true;

        //Setting valid non-railML schemas from schemaLocation

schemaControl.SetAndCheckAllSchemas(railMLDocument.docume
nt, result, railMLDocument.path);
        //Adding those schemas to the schema set
        schemaControl.AddOtherShemas(schemaSet);

        //Finding extensions to be able to validate them
        extensions.ProgressUpdated +=
OnExtensionsProgressUpdated;

extensions.FindAndSetExtensions(railMLDocument.document,
schemaControl.SchemaLocations);

        try
        {
          schemaSet.Compile();
        }
        catch (Exception ex)
        {
          result.AddError(ex.Message);
        }

        XmlReaderSettings settingsReader = new
XmlReaderSettings
        {
          ValidationType = ValidationType.Schema,
          Schemas = schemaSet
        };
        settingsReader.ValidationEventHandler +=
ValidationEventHandler;

        //Validate the XML document directly from the file path
        using (XmlReader reader =
XmlReader.Create(railMLDocument.path, settingsReader))
        {
          try
          {
            while (reader.Read()) { }
          }
          catch (Exception ex)
          {
            result.AddError(ex.Message);
          }
        }

        try
        {
          int i = 1;
          foreach (var issue in issues)
          {
            OnProgressChanged(i * 100 / issues.Count);
```

```
                i++;
                validation.HandleIssues(issue, result, extensions);
            }
        }
        catch (Exception ex)
        {
            result.AddError(ex.Message);
        }

    }
    catch (Exception ex)
    {
        result.AddError(ex.Message);
    }

    return new Tuple<ValidationResult,
railMLExtensions>(result, extensions);
  }

  protected virtual void OnProgressChanged(int percent)
  {
    ValidationProgressChanged?.Invoke(percent);
  }

  private void OnExtensionsProgressUpdated(object sender, int
progressPercentage)
  {
    // Relay percentage progress update
    ExtensionsProgressRelayed?.Invoke(progressPercentage);
  }

  void ValidationEventHandler(object sender,
ValidationEventArgs e)
  {
    issues.Add(e);
  }

 }
}
```

1.22 Текст файлу ValidationTests.cs модуля ValidationTests

```
using railVIVID.Validation;

using railVIVID.Validation.Extensions;

using railVIVID.Validation.GeneralInformation;

using railVIVID.Validation.XDocument;

using railVIVID.Validation.XDocumentSolution;

using railVIVID.Validation.XMLDocument;

using railVIVID.Validation.XMLReader;


namespace ValidationTests

{

  public class UnitTestPrevalidation

  {

    private Prevalidation prevalidation;


    [Theory]

    [InlineData(@"\TestData\Ostsachsen_V220_Good.railml", 0)]


[InlineData(@"\TestData\Ostsachsen_V220_DefaultNamespace.rail
ml", 0)]


[InlineData(@"\TestData\Ostsachsen_V220_Namespace.railml", 0)]

    [InlineData(@"\TestData\Ostsachsen_V220_Formatting.railml",
1)]

    [InlineData(@"\TestData\justtext.txt", 1)]

    [InlineData(@"\TestData\noneexist.railml", 1)]

    public void Prevalidation_Test(string filePath, int expected)

    {

      prevalidation = new Prevalidation();

      string baseDirectory = AppContext.BaseDirectory;

      string projectDirectory =
Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;

      railMLDocument railMLDocument = new
railMLDocument();

      railMLVersion railMLVersion = new railMLVersion();

      ValidationResult result =
prevalidation.Prevalidate(projectDirectory + filePath, out
railMLDocument, out railMLVersion);

      int actual = result.errors.Count();

      Assert.Equal(expected, actual);

    }


    [Theory]

    [InlineData(@"\TestData\Ostsachsen_V220_NoVersion.railml",
"railML namespace is missing in the file")]

    [InlineData(@"\TestData\Ostsachsen_V220_Version.railml",
"railML namespace is missing in the file")]

    [InlineData(@"\TestData\justtext.txt", "Unsupported file
format")]

    public void PrevalidationVersionDescription_Test(string
filePath, string expected)

    {

      Prevalidation prevalidation = new Prevalidation();

      string baseDirectory = AppContext.BaseDirectory;

      string projectDirectory =
Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;

      railMLDocument railMLDocument = new
railMLDocument();

      railMLVersion railMLVersion = new railMLVersion();

      ValidationResult result =
prevalidation.Prevalidate(projectDirectory + filePath, out
railMLDocument, out railMLVersion);

      string actual = result.errors.First().description;

      Assert.Contains(expected, actual);

    }


    [Theory]

    [InlineData("2.1", "http://www.railml.org/schemas/2011",
"2.1")]
```

```
    [InlineData("2.1", "https://www.railml.org/schemas/3.1", "3.1")]
    [InlineData("", "https://www.railml.org/schemas/3.1", "3.1")]
    [InlineData("test", "test", "")]
    public void IdentifyVersion_Test(string version, string railMLNamespace, string expected)
    {
        railMLVersion railMLVersion = new railMLVersion();
        railMLVersion.IdentifyRailMlVersion(version, railMLNamespace);
        string actual = railMLVersion.VersionFromNamespace;
        Assert.Equal(expected, actual);
    }
}

public class UnitTestValidation
{

    /// <summary>
    /// Have to be run individually! Running these tests simultaneously causes different results each time,
    /// most likely due to denial of access or blocking
    /// </summary>
    /// <param name="filePath"></param>
    /// <param name="expectedIssues"></param>
    [Theory]
    [InlineData(@"\TestData\Ostsachsen_V220.xml", false)]
    [InlineData(@"\TestData\Ostsachsen_V200.xml", false)]
    [InlineData(@"\TestData\Ostsachsen_V210.xml", false)]
[InlineData(@"\TestData\161115_BaneNor_NorwayRailNetwork.railml", false)]
[InlineData(@"\TestData\090318_Bahnkonzept_ExampleDataGPSinfradat_NorthUpperRhineNetworkRailML23.railml", false)]
[InlineData(@"\TestData\090318_Bahnkonzept_ExampleDataGPSinfradat_NorthUpperRhineNetworkRailML23.railmlx", false)]
[InlineData(@"\TestData\OpenTrack_to_railML.V_2.2.rollingstock.railml", false)]
[InlineData(@"\TestData\OpenTrack_to_railML.V_2.2.timetable.railml", false)]
    [InlineData(@"\TestData\SimpleExtensionExample.xml", true)]
[InlineData(@"\TestData\SimpleExtensionOneDeclaredOneUndeclared.xml", false)]
    [InlineData(@"\TestData\SimpleExample.xml", true)]
    [InlineData(@"\TestData\SimpleExtensionExample1.xml", true)]
    [InlineData(@"\TestData\SimpleExtensionExample2.xml", true)]
    [InlineData(@"\TestData\Bsp_V220.xml", false)]
    [InlineData(@"\TestData\Bsp_V250.xml", false)]
    [InlineData(@"\TestData\SCHROTT - railML_SimpleExample_v11_railML3-1_01.xml", true)]
    [InlineData(@"\TestData\V221-i4 Stand 2-2-11-89 Suedthueringen mit Vorlage.railml", false)]
[InlineData(@"\TestData\railML_SimpleExample_v11_railML2-3_01.xml", false)]
[InlineData(@"\TestData\railML_SimpleExample_v11_railML2-5_01.xml", false)]
[InlineData(@"\TestData\railML_SimpleExample_v11_railML3-1_04.xml", false)]
[InlineData(@"\TestData\railML_SimpleExample_v11_railML2-4_01.xml", false)]
    public void XDocumentSolutionErrors_Test(string filePath, bool expectedIssues)
    {
        string baseDirectory = AppContext.BaseDirectory;
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;
        Prevalidation prevalidation = new Prevalidation();
        railMLDocument railMLDocument = new railMLDocument();
        railMLVersion railMLVersion = new railMLVersion();
        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);
        IValidator methodOne = new ValidationXDocumentSolution();
        Tuple<ValidationResult, railMLExtensions> results = methodOne.Validate(railMLDocument, railMLVersion);
        ValidationResult result = results.Item1;
        int actualErrors = result.errors.Count();
        bool errorsPresent = false;
        if (actualErrors > 0)
        {
            errorsPresent = true;
        }
        Assert.Equal(expectedIssues, errorsPresent);
    }

    /// <summary>
    /// Have to be run individually! Running these tests simultaneously causes different results each time,
    /// most likely due to denial of access or blocking
    /// </summary>
    /// <param name="filePath"></param>
    /// <param name="expectedIssues"></param>
    [Theory]
    [InlineData(@"\TestData\Ostsachsen_V220.xml", false)]
    [InlineData(@"\TestData\Ostsachsen_V200.xml", false)]
```

```
[InlineData(@"\TestData\Ostsachsen_V210.xml", false)]

[InlineData(@"\TestData\161115_BaneNor_NorwayRailNetwork.railml", false)]

[InlineData(@"\TestData\090318_Bahnkonzept_ExampleDataGPSInfradat_NorthUpperRhineNetworkRailML23.railml", false)]

[InlineData(@"\TestData\090318_Bahnkonzept_ExampleDataGPSInfradat_NorthUpperRhineNetworkRailML23.railmlx", false)]

[InlineData(@"\TestData\OpenTrack_to_railML.V_2.2.rollingstock.railml", false)]

[InlineData(@"\TestData\OpenTrack_to_railML.V_2.2.timetable.railml", false)]

[InlineData(@"\TestData\SimpleExtensionExample.xml", false)]

[InlineData(@"\TestData\SimpleExtensionOneDeclaredOneUndeclared.xml", true)]

[InlineData(@"\TestData\SimpleExample.xml", false)]

[InlineData(@"\TestData\SimpleExtensionExample1.xml", false)]

[InlineData(@"\TestData\SimpleExtensionExample2.xml", true)]

[InlineData(@"\TestData\Bsp_V220.xml", false)]

[InlineData(@"\TestData\Bsp_V250.xml", false)]

[InlineData(@"\TestData\SCHROTT - railML_SimpleExample_v11_railML3-1_01.xml", false)]

[InlineData(@"\TestData\V221-i4 Stand 2-2-11-89 Suedthueringen mit Vorlage.railml", false)]

[InlineData(@"\TestData\railML_SimpleExample_v11_railML2-3_01.xml", false)]

[InlineData(@"\TestData\railML_SimpleExample_v11_railML2-5_01.xml", false)]

[InlineData(@"\TestData\railML_SimpleExample_v11_railML3-1_04.xml", false)]

[InlineData(@"\TestData\railML_SimpleExample_v11_railML2-4_01.xml", true)]

public void ValidationMethod1Warnings_Test(string filePath, bool expectedIssues)

{

string baseDirectory = AppContext.BaseDirectory;

string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;

Prevalidation prevalidation = new Prevalidation();

railMLDocument railMLDocument = new railMLDocument();

railMLVersion railMLVersion = new railMLVersion();

prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);

IValidator methodOne = new ValidationXDocumentSolution();

Tuple<ValidationResult, railMLExtensions> results = methodOne.Validate(railMLDocument, railMLVersion);

ValidationResult result = results.Item1;

int actualWarnings = result.warnings.Count();

bool presentWarnings = false;

if (actualWarnings > 0)

{

presentWarnings = true;

}

Assert.Equal(expectedIssues, presentWarnings);

}


//Have to be run individually! Running these tests simultaneously causes different results each time,

//most likely due to denial of access or blocking

[Theory]

[InlineData(@"\TestData\Ostsachsen_V220.xml", false, false)]

[InlineData(@"\TestData\Ostsachsen_V200.xml", false, false)]

[InlineData(@"\TestData\Ostsachsen_V210.xml", false, false)]

[InlineData(@"\TestData\161115_BaneNor_NorwayRailNetwork.railml", false, false)]

[InlineData(@"\TestData\090318_Bahnkonzept_ExampleDataGPSInfradat_NorthUpperRhineNetworkRailML23.railml", false, false)]

[InlineData(@"\TestData\OpenTrack_to_railML.V_2.2.rollingstock.railml", false, false)]

[InlineData(@"\TestData\OpenTrack_to_railML.V_2.2.timetable.railml", false, false)]

[InlineData(@"\TestData\SimpleExtensionExample.xml", true, false)]

[InlineData(@"\TestData\SimpleExtensionOneDeclaredOneUndeclared.xml", false, true)]

[InlineData(@"\TestData\SimpleExample.xml", true, false)]

[InlineData(@"\TestData\SimpleExtensionExample1.xml", true, false)]

[InlineData(@"\TestData\SimpleExtensionExample2.xml", true, true)]

[InlineData(@"\TestData\Bsp_V220.xml", false, false)]

[InlineData(@"\TestData\Bsp_V250.xml", false, false)]

[InlineData(@"\TestData\SCHROTT - railML_SimpleExample_v11_railML3-1_01.xml", true, false)]

[InlineData(@"\TestData\V221-i4 Stand 2-2-11-89 SЂdthЂringen mit Vorlage.railml", false, false)]

[InlineData(@"\TestData\railML_SimpleExample_v11_railML2-3_01.xml", false, false)]

[InlineData(@"\TestData\railML_SimpleExample_v11_railML2-5_01.xml", false, false)]
```

```
[InlineData(@"\TestData\railML_SimpleExample_v11_railML3-1_04.xml", false, false)]

[InlineData(@"\TestData\railML_SimpleExample_v11_railML2-4_01.xml", false, true)]
    public void XDocument_Test(string filePath, bool expectedErros, bool expectedWarnings)
    {
        string baseDirectory = AppContext.BaseDirectory;
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;
        railMLDocument railMLDocument = new railMLDocument();
        railMLVersion railMLVersion = new railMLVersion();
        Prevalidation prevalidation = new Prevalidation();
        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);
        IValidator validator = new ValidationXDocument();
        Tuple<ValidationResult, railMLExtensions> results = validator.Validate(railMLDocument, railMLVersion);
        ValidationResult result = results.Item1;
        int actualWarnings = result.warnings.Count();
        int actualErrors = result.errors.Count();
        bool presentWarnings = false;
        bool presentErrors = false;
        if (actualWarnings > 0)
        {
            presentWarnings = true;
        }
        if(actualErrors > 0)
        {
            presentErrors = true;
        }
        Assert.Equal(expectedErros, presentErrors);
        Assert.Equal(expectedWarnings, presentWarnings);
    }

    //Have to be run individually! Running these tests simultaneously causes different results each time,
    //most likely due to denial of access or blocking
    [Theory]
    [InlineData(@"\TestData\Ostsachsen_V220.xml", false, false)]
    [InlineData(@"\TestData\Ostsachsen_V200.xml", false, false)]
    [InlineData(@"\TestData\Ostsachsen_V210.xml", false, false)]

[InlineData(@"\TestData\161115_BaneNor_NorwayRailNetwork.railml", false, false)]

[InlineData(@"\TestData\090318_Bahnkonzept_ExampleDataGPSinfradat_NorthUpperRhineNetworkRailML23.railml", false, false)]

[InlineData(@"\TestData\OpenTrack_to_railML.V_2.2.rollingstock.railml", false, false)]

[InlineData(@"\TestData\OpenTrack_to_railML.V_2.2.timetable.railml", false, false)]
    [InlineData(@"\TestData\SimpleExtensionExample.xml", true, false)]

[InlineData(@"\TestData\SimpleExtensionOneDeclaredOneUndeclared.xml", false, true)]
    [InlineData(@"\TestData\SimpleExample.xml", true, false)]
    [InlineData(@"\TestData\SimpleExtensionExample1.xml", true, false)]
    [InlineData(@"\TestData\SimpleExtensionExample2.xml", true, true)]
    [InlineData(@"\TestData\Bsp_V220.xml", false, false)]
    [InlineData(@"\TestData\Bsp_V250.xml", false, false)]
    [InlineData(@"\TestData\SCHROTT - railML_SimpleExample_v11_railML3-1_01.xml", true, false)]
    [InlineData(@"\TestData\V221-i4 Stand 2-2-11-89 SЬdthЬringen mit Vorlage.railml", false, false)]

[InlineData(@"\TestData\railML_SimpleExample_v11_railML2-3_01.xml", false, false)]

[InlineData(@"\TestData\railML_SimpleExample_v11_railML2-5_01.xml", false, false)]

[InlineData(@"\TestData\railML_SimpleExample_v11_railML3-1_04.xml", false, false)]

[InlineData(@"\TestData\railML_SimpleExample_v11_railML2-4_01.xml", false, true)]
    public void XMLDocument_Test(string filePath, bool expectedErros, bool expectedWarnings)
    {
        string baseDirectory = AppContext.BaseDirectory;
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;
        railMLDocument railMLDocument = new railMLDocument();
        railMLVersion railMLVersion = new railMLVersion();
        Prevalidation prevalidation = new Prevalidation();
        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);
        IValidator validator = new ValidationXMLDocument();
        Tuple<ValidationResult, railMLExtensions> results = validator.Validate(railMLDocument, railMLVersion);
        ValidationResult result = results.Item1;
        int actualWarnings = result.warnings.Count();
        int actualErrors = result.errors.Count();
        bool presentWarnings = false;
        bool presentErrors = false;
        if (actualWarnings > 0)
```

```
        {
            presentWarnings = true;
        }
        if (actualErrors > 0)
        {
            presentErrors = true;
        }
        Assert.Equal(expectedErros, presentErrors);
        Assert.Equal(expectedWarnings, presentWarnings);
    }

    //Have to be run individually! Running these tests simultaneously causes different results each time,
    //most likely due to denial of access or blocking
    [Theory]
    [InlineData(@"\TestData\Ostsachsen_V220.xml", false, false)]
    [InlineData(@"\TestData\Ostsachsen_V200.xml", false, false)]
    [InlineData(@"\TestData\Ostsachsen_V210.xml", false, false)]

[InlineData(@"\TestData\161115_BaneNor_NorwayRailNetwork.railml", false, false)]

[InlineData(@"\TestData\090318_Bahnkonzept_ExampleDataGPSinfradat_NorthUpperRhineNetworkRailML23.railml", false, false)]

[InlineData(@"\TestData\OpenTrack_to_railML.V_2.2.rollingstock.railml", false, false)]

[InlineData(@"\TestData\OpenTrack_to_railML.V_2.2.timetable.railml", false, false)]
    [InlineData(@"\TestData\SimpleExtensionExample.xml", true, false)]

[InlineData(@"\TestData\SimpleExtensionOneDeclaredOneUndeclared.xml", false, true)]
    [InlineData(@"\TestData\SimpleExample.xml", true, false)]
    [InlineData(@"\TestData\SimpleExtensionExample1.xml", true, false)]
    [InlineData(@"\TestData\SimpleExtensionExample2.xml", true, true)]
    [InlineData(@"\TestData\Bsp_V220.xml", false, false)]
    [InlineData(@"\TestData\Bsp_V250.xml", false, false)]
    [InlineData(@"\TestData\SCHROTT - railML_SimpleExample_v11_railML3-1_01.xml", true, false)]
    [InlineData(@"\TestData\V221-i4 Stand 2-2-11-89 SЬdthЬringen mit Vorlage.railml", false, false)]

[InlineData(@"\TestData\railML_SimpleExample_v11_railML2-3_01.xml", false, false)]

[InlineData(@"\TestData\railML_SimpleExample_v11_railML2-5_01.xml", false, false)]

[InlineData(@"\TestData\railML_SimpleExample_v11_railML3-1_04.xml", false, false)]

[InlineData(@"\TestData\railML_SimpleExample_v11_railML2-4_01.xml", false, true)]
    public void XMLReader_Test(string filePath, bool expectedErros, bool expectedWarnings)
    {
        string baseDirectory = AppContext.BaseDirectory;
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;
        railMLDocument railMLDocument = new railMLDocument();
        railMLVersion railMLVersion = new railMLVersion();
        Prevalidation prevalidation = new Prevalidation();
        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);
        IValidator validator = new ValidationXMLReader();
        Tuple<ValidationResult, railMLExtensions> results = validator.Validate(railMLDocument, railMLVersion);
        ValidationResult result = results.Item1;
        int actualWarnings = result.warnings.Count();
        int actualErrors = result.errors.Count();
        bool presentWarnings = false;
        bool presentErrors = false;
        if (actualWarnings > 0)
        {
            presentWarnings = true;
        }
        if (actualErrors > 0)
        {
            presentErrors = true;
        }
        Assert.Equal(expectedErros, presentErrors);
        Assert.Equal(expectedWarnings, presentWarnings);
    }

    [Theory]
    [InlineData("XDocument", @"\TestData\Examples\2_Attribute_NotAllowed_Defined.xml")]
    [InlineData("XMLDocument", @"\TestData\Examples\2_Attribute_NotAllowed_Defined.xml")]
    [InlineData("XMLReader", @"\TestData\Examples\2_Attribute_NotAllowed_Defined.xml")]
    [InlineData("XDocumentSolution", @"\TestData\Examples\2_Attribute_NotAllowed_Defined.xml")]
    public void ValidationSituation5_Test(string ValidationMethod, string filePath)
    {
        string baseDirectory = AppContext.BaseDirectory;
```

```
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;

        railMLDocument railMLDocument = new railMLDocument();

        railMLVersion railMLVersion = new railMLVersion();

        Prevalidation prevalidation = new Prevalidation();

        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);

        ValidationResult result = new ValidationResult(railMLDocument);

        switch (ValidationMethod)

        {

           case "XDocument":

              IValidator validator1 = new ValidationXDocument();

              Tuple<ValidationResult, railMLExtensions> results1 = validator1.Validate(railMLDocument, railMLVersion);

              result = results1.Item1;

              break;

           case "XMLDocument":

              IValidator validator2 = new ValidationXMLDocument();

              Tuple<ValidationResult, railMLExtensions> results2 = validator2.Validate(railMLDocument, railMLVersion);

              result = results2.Item1;

              break;

           case "XMLReader":

              IValidator validator3 = new ValidationXMLReader();

              Tuple<ValidationResult, railMLExtensions> results3 = validator3.Validate(railMLDocument, railMLVersion);

              result = results3.Item1;

              break;

           case "XDocumentSolution":

              IValidator validator4 = new ValidationXDocumentSolution();

              Tuple<ValidationResult, railMLExtensions> results4 = validator4.Validate(railMLDocument, railMLVersion);

              result = results4.Item1;

              break;

        }

        int actualErrors = result.errors.Count();

        bool presentErrors = false;

        if (actualErrors > 0)

        {

           presentErrors = true;

        }

        Assert.True(presentErrors);

     }


     [Theory]

     [InlineData("XDocument", @"\TestData\Examples\2_Attribute_NotAllowed_NotDefined.xml")]

     [InlineData("XMLDocument", @"\TestData\Examples\2_Attribute_NotAllowed_NotDefined.xml")]

     [InlineData("XMLReader", @"\TestData\Examples\2_Attribute_NotAllowed_NotDefined.xml")]

     [InlineData("XDocumentSolution", @"\TestData\Examples\2_Attribute_NotAllowed_NotDefined.xml")]

     public void ValidationSituation6_Test(string ValidationMethod, string filePath)

     {

        string baseDirectory = AppContext.BaseDirectory;

        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;

        railMLDocument railMLDocument = new railMLDocument();

        railMLVersion railMLVersion = new railMLVersion();

        Prevalidation prevalidation = new Prevalidation();

        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);

        ValidationResult result = new ValidationResult(railMLDocument);

        switch (ValidationMethod)

        {

           case "XDocument":

              IValidator validator1 = new ValidationXDocument();

              Tuple<ValidationResult, railMLExtensions> results1 = validator1.Validate(railMLDocument, railMLVersion);

              result = results1.Item1;

              break;

           case "XMLDocument":

              IValidator validator2 = new ValidationXMLDocument();

              Tuple<ValidationResult, railMLExtensions> results2 = validator2.Validate(railMLDocument, railMLVersion);

              result = results2.Item1;

              break;

           case "XMLReader":

              IValidator validator3 = new ValidationXMLReader();

              Tuple<ValidationResult, railMLExtensions> results3 = validator3.Validate(railMLDocument, railMLVersion);

              result = results3.Item1;

              break;

           case "XDocumentSolution":

              IValidator validator4 = new ValidationXDocumentSolution();

              Tuple<ValidationResult, railMLExtensions> results4 = validator4.Validate(railMLDocument, railMLVersion);

              result = results4.Item1;

              break;

        }

        int actualErrors = result.errors.Count();

        bool presentErrors = false;

        if (actualErrors > 0)
```

```
        {
            presentErrors = true;
        }
        Assert.True(presentErrors);
    }

    [Theory]
    [InlineData("XDocument",
@"\TestData\Examples\2_Element_Allowed.xml")]
    [InlineData("XMLDocument",
@"\TestData\Examples\2_Element_Allowed.xml")]
    [InlineData("XMLReader",
@"\TestData\Examples\2_Element_Allowed.xml")]
    [InlineData("XDocumentSolution",
@"\TestData\Examples\2_Element_Allowed.xml")]
    public void ValidationSituation1Full_Test(string
ValidationMethod, string filePath)
    {
        string baseDirectory = AppContext.BaseDirectory;
        string projectDirectory =
Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;
        railMLDocument railMLDocument = new
railMLDocument();
        railMLVersion railMLVersion = new railMLVersion();
        Prevalidation prevalidation = new Prevalidation();
        prevalidation.Prevalidate(projectDirectory + filePath, out
railMLDocument, out railMLVersion);
        ValidationResult result = new
ValidationResult(railMLDocument);
        switch (ValidationMethod)
        {
            case "XDocument":
                IValidator validator1 = new ValidationXDocument();
                Tuple<ValidationResult, railMLExtensions> results1 =
validator1.Validate(railMLDocument, railMLVersion);
                result = results1.Item1;
                break;
            case "XMLDocument":
                IValidator validator2 = new ValidationXMLDocument();
                Tuple<ValidationResult, railMLExtensions> results2 =
validator2.Validate(railMLDocument, railMLVersion);
                result = results2.Item1;
                break;
            case "XMLReader":
                IValidator validator3 = new ValidationXMLReader();
                Tuple<ValidationResult, railMLExtensions> results3 =
validator3.Validate(railMLDocument, railMLVersion);
                result = results3.Item1;
                break;
            case "XDocumentSolution":
                IValidator validator4 = new
ValidationXDocumentSolution();
                Tuple<ValidationResult, railMLExtensions> results4 =
validator4.Validate(railMLDocument, railMLVersion);
                result = results4.Item1;
                break;
        }
        bool errorsPresent = false;
        if(result.errors.Count() > 0)
        {
            errorsPresent = true;
        }
        Assert.False(errorsPresent);
        bool declarationPresent = false;
        foreach( var warning in result.warnings )
        {
            if(warning.description.Contains("not declared") ||
warning.description.Contains("declaration"))
            {
                declarationPresent = true;
            }
        }

        Assert.True(declarationPresent);
    }

    [Theory]
    [InlineData("XDocument",
@"\TestData\Examples\2_Element_NotAllowed_Defined.xml")]
    [InlineData("XMLDocument",
@"\TestData\Examples\2_Element_NotAllowed_Defined.xml")]
    [InlineData("XMLReader",
@"\TestData\Examples\2_Element_NotAllowed_Defined.xml")]
    [InlineData("XDocumentSolution",
@"\TestData\Examples\2_Element_NotAllowed_Defined.xml")]
    public void ValidationSituation2Full_Test(string
ValidationMethod, string filePath)
    {
        string baseDirectory = AppContext.BaseDirectory;
        string projectDirectory =
Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;
        railMLDocument railMLDocument = new
railMLDocument();
        railMLVersion railMLVersion = new railMLVersion();
        Prevalidation prevalidation = new Prevalidation();
        prevalidation.Prevalidate(projectDirectory + filePath, out
railMLDocument, out railMLVersion);
        ValidationResult result = new
ValidationResult(railMLDocument);
        switch (ValidationMethod)
        {
```

```
            case "XDocument":
                IValidator validator1 = new ValidationXDocument();
                Tuple<ValidationResult, railMLExtensions> results1 = validator1.Validate(railMLDocument, railMLVersion);
                result = results1.Item1;
                break;
            case "XMLDocument":
                IValidator validator2 = new ValidationXMLDocument();
                Tuple<ValidationResult, railMLExtensions> results2 = validator2.Validate(railMLDocument, railMLVersion);
                result = results2.Item1;
                break;
            case "XMLReader":
                IValidator validator3 = new ValidationXMLReader();
                Tuple<ValidationResult, railMLExtensions> results3 = validator3.Validate(railMLDocument, railMLVersion);
                result = results3.Item1;
                break;
            case "XDocumentSolution":
                IValidator validator4 = new ValidationXDocumentSolution();
                Tuple<ValidationResult, railMLExtensions> results4 = validator4.Validate(railMLDocument, railMLVersion);
                result = results4.Item1;
                break;
        }
        bool errorsPresent = false;
        if (result.errors.Count() > 0)
        {
            errorsPresent = true;
        }
        Assert.True(errorsPresent);
        bool notAllowedPresent = false;
        foreach (var error in result.errors)
        {
            if (error.description.Contains("not allowed") || error.description.Contains("invalid child"))
            {
                notAllowedPresent = true;
            }
        }

        Assert.True(notAllowedPresent);
    }

    [Theory]
    [InlineData("XDocument", @"\TestData\Examples\2_Element_NotAllowed_NotDefined.xml")]
    [InlineData("XMLDocument", @"\TestData\Examples\2_Element_NotAllowed_NotDefined.xml")]
    [InlineData("XMLReader", @"\TestData\Examples\2_Element_NotAllowed_NotDefined.xml")]
    [InlineData("XDocumentSolution", @"\TestData\Examples\2_Element_NotAllowed_NotDefined.xml")]
    public void ValidationSituation3Full_Test(string ValidationMethod, string filePath)
    {
        string baseDirectory = AppContext.BaseDirectory;
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;
        railMLDocument railMLDocument = new railMLDocument();
        railMLVersion railMLVersion = new railMLVersion();
        Prevalidation prevalidation = new Prevalidation();
        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);
        ValidationResult result = new ValidationResult(railMLDocument);
        switch (ValidationMethod)
        {
            case "XDocument":
                IValidator validator1 = new ValidationXDocument();
                Tuple<ValidationResult, railMLExtensions> results1 = validator1.Validate(railMLDocument, railMLVersion);
                result = results1.Item1;
                break;
            case "XMLDocument":
                IValidator validator2 = new ValidationXMLDocument();
                Tuple<ValidationResult, railMLExtensions> results2 = validator2.Validate(railMLDocument, railMLVersion);
                result = results2.Item1;
                break;
            case "XMLReader":
                IValidator validator3 = new ValidationXMLReader();
                Tuple<ValidationResult, railMLExtensions> results3 = validator3.Validate(railMLDocument, railMLVersion);
                result = results3.Item1;
                break;
            case "XDocumentSolution":
                IValidator validator4 = new ValidationXDocumentSolution();
                Tuple<ValidationResult, railMLExtensions> results4 = validator4.Validate(railMLDocument, railMLVersion);
                result = results4.Item1;
                break;
        }
        bool errorsPresent = false;
        if (result.errors.Count() > 0)
        {
```

```
            errorsPresent = true;
        }
        Assert.True(errorsPresent);
        bool notAllowedPresent = false;
        foreach (var error in result.errors)
        {
            if (error.description.Contains("not allowed") || error.description.Contains("invalid child")
                || error.description.Contains("cannot contain"))
            {
                notAllowedPresent = true;
            }
        }

        Assert.True(notAllowedPresent);
    }

    [Theory]
    [InlineData("XDocument", @"\TestData\Examples\2_Attribute_Allowed.xml")]
    [InlineData("XMLDocument", @"\TestData\Examples\2_Attribute_Allowed.xml")]
    [InlineData("XMLReader", @"\TestData\Examples\2_Attribute_Allowed.xml")]
    [InlineData("XDocumentSolution", @"\TestData\Examples\2_Attribute_Allowed.xml")]
    public void ValidationSituation4Full_Test(string ValidationMethod, string filePath)
    {
        string baseDirectory = AppContext.BaseDirectory;
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;
        railMLDocument railMLDocument = new railMLDocument();
        railMLVersion railMLVersion = new railMLVersion();
        Prevalidation prevalidation = new Prevalidation();
        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);
        ValidationResult result = new ValidationResult(railMLDocument);
        switch (ValidationMethod)
        {
            case "XDocument":
                IValidator validator1 = new ValidationXDocument();
                Tuple<ValidationResult, railMLExtensions> results1 = validator1.Validate(railMLDocument, railMLVersion);
                result = results1.Item1;
                break;
            case "XMLDocument":
                IValidator validator2 = new ValidationXMLDocument();
                Tuple<ValidationResult, railMLExtensions> results2 = validator2.Validate(railMLDocument, railMLVersion);
                result = results2.Item1;
                break;
            case "XMLReader":
                IValidator validator3 = new ValidationXMLReader();
                Tuple<ValidationResult, railMLExtensions> results3 = validator3.Validate(railMLDocument, railMLVersion);
                result = results3.Item1;
                break;
            case "XDocumentSolution":
                IValidator validator4 = new ValidationXDocumentSolution();
                Tuple<ValidationResult, railMLExtensions> results4 = validator4.Validate(railMLDocument, railMLVersion);
                result = results4.Item1;
                break;
        }
        bool errorsPresent = false;
        if (result.errors.Count() > 0)
        {
            errorsPresent = true;
        }
        Assert.False(errorsPresent);
        bool declarationPresent = false;
        foreach (var warning in result.warnings)
        {
            if (warning.description.Contains("not declared") || warning.description.Contains("declaration"))
            {
                declarationPresent = true;
            }
        }

        Assert.True(declarationPresent);
    }

    [Theory]
    [InlineData("XDocument", @"\TestData\Examples\2_Attribute_NotAllowed_Defined.xml")]
    [InlineData("XMLDocument", @"\TestData\Examples\2_Attribute_NotAllowed_Defined.xml")]
    [InlineData("XMLReader", @"\TestData\Examples\2_Attribute_NotAllowed_Defined.xml")]
    [InlineData("XDocumentSolution", @"\TestData\Examples\2_Attribute_NotAllowed_Defined.xml")]
    public void ValidationSituation5Text_Test(string ValidationMethod, string filePath)
    {
        string baseDirectory = AppContext.BaseDirectory;
```

```
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;

        railMLDocument railMLDocument = new railMLDocument();

        railMLVersion railMLVersion = new railMLVersion();

        Prevalidation prevalidation = new Prevalidation();

        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);

        ValidationResult result = new ValidationResult(railMLDocument);

        switch (ValidationMethod)

        {

           case "XDocument":

              IValidator validator1 = new ValidationXDocument();

              Tuple<ValidationResult, railMLExtensions> results1 = validator1.Validate(railMLDocument, railMLVersion);

              result = results1.Item1;

              break;

           case "XMLDocument":

              IValidator validator2 = new ValidationXMLDocument();

              Tuple<ValidationResult, railMLExtensions> results2 = validator2.Validate(railMLDocument, railMLVersion);

              result = results2.Item1;

              break;

           case "XMLReader":

              IValidator validator3 = new ValidationXMLReader();

              Tuple<ValidationResult, railMLExtensions> results3 = validator3.Validate(railMLDocument, railMLVersion);

              result = results3.Item1;

              break;

           case "XDocumentSolution":

              IValidator validator4 = new ValidationXDocumentSolution();

              Tuple<ValidationResult, railMLExtensions> results4 = validator4.Validate(railMLDocument, railMLVersion);

              result = results4.Item1;

              break;

        }

        string actualErrors = null;

        if (result.errors.Count > 0)

        {

           actualErrors = result.errors.First().description;

           Assert.Contains("not allowed", actualErrors);

        }

        string actualWarnings = null;

        if (result.warnings.Count > 0)

        {

           actualWarnings = result.warnings.First().description;

           Assert.Contains("not allowed", actualWarnings);

        }

     }


     [Theory]

     [InlineData("XDocument", @"\TestData\Examples\2_AttributeWithSchema_NotAllowed_Defined.xml")]

     [InlineData("XMLDocument", @"\TestData\Examples\2_AttributeWithSchema_NotAllowed_Defined.xml")]

     [InlineData("XMLReader", @"\TestData\Examples\2_AttributeWithSchema_NotAllowed_Defined.xml")]

     [InlineData("XDocumentSolution", @"\TestData\Examples\2_AttributeWithSchema_NotAllowed_Defined.xml")]

     public void ValidationSituation5WithSchemaText_Test(string ValidationMethod, string filePath)

     {

        string baseDirectory = AppContext.BaseDirectory;

        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;

        railMLDocument railMLDocument = new railMLDocument();

        railMLVersion railMLVersion = new railMLVersion();

        Prevalidation prevalidation = new Prevalidation();

        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);

        ValidationResult result = new ValidationResult(railMLDocument);

        switch (ValidationMethod)

        {

           case "XDocument":

              IValidator validator1 = new ValidationXDocument();

              Tuple<ValidationResult, railMLExtensions> results1 = validator1.Validate(railMLDocument, railMLVersion);

              result = results1.Item1;

              break;

           case "XMLDocument":

              IValidator validator2 = new ValidationXMLDocument();

              Tuple<ValidationResult, railMLExtensions> results2 = validator2.Validate(railMLDocument, railMLVersion);

              result = results2.Item1;

              break;

           case "XMLReader":

              IValidator validator3 = new ValidationXMLReader();

              Tuple<ValidationResult, railMLExtensions> results3 = validator3.Validate(railMLDocument, railMLVersion);

              result = results3.Item1;

              break;

           case "XDocumentSolution":

              IValidator validator4 = new ValidationXDocumentSolution();
```

```
            Tuple<ValidationResult, railMLExtensions> results4 = validator4.Validate(railMLDocument, railMLVersion);
            result = results4.Item1;
            break;
        }
        string actualErrors = null;
        if (result.errors.Count > 0)
        {
            actualErrors = result.errors.First().description;
            Assert.Contains("not declared", actualErrors);
        }
        string actualWarnings = null;
        if (result.warnings.Count > 0)
        {
            actualWarnings = result.warnings.First().description;
            Assert.Contains("not declared", actualWarnings);
        }
    }

    [Theory]
    [InlineData("XDocument", @"\TestData\Examples\2_Attribute_NotAllowed_NotDefined.xml")]
    [InlineData("XMLDocument", @"\TestData\Examples\2_Attribute_NotAllowed_NotDefined.xml")]
    [InlineData("XMLReader", @"\TestData\Examples\2_Attribute_NotAllowed_NotDefined.xml")]
    [InlineData("XDocumentSolution", @"\TestData\Examples\2_Attribute_NotAllowed_NotDefined.xml")]
    public void ValidationSituation6Text_Test(string ValidationMethod, string filePath)
    {
        string baseDirectory = AppContext.BaseDirectory;
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;
        railMLDocument railMLDocument = new railMLDocument();
        railMLVersion railMLVersion = new railMLVersion();
        Prevalidation prevalidation = new Prevalidation();
        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);
        ValidationResult result = new ValidationResult(railMLDocument);
        switch (ValidationMethod)
        {
            case "XDocument":
                IValidator validator1 = new ValidationXDocument();
                Tuple<ValidationResult, railMLExtensions> results1 = validator1.Validate(railMLDocument, railMLVersion);
                result = results1.Item1;
                break;
            case "XMLDocument":
                IValidator validator2 = new ValidationXMLDocument();
                Tuple<ValidationResult, railMLExtensions> results2 = validator2.Validate(railMLDocument, railMLVersion);
                result = results2.Item1;
                break;
            case "XMLReader":
                IValidator validator3 = new ValidationXMLReader();
                Tuple<ValidationResult, railMLExtensions> results3 = validator3.Validate(railMLDocument, railMLVersion);
                result = results3.Item1;
                break;
            case "XDocumentSolution":
                IValidator validator4 = new ValidationXDocumentSolution();
                Tuple<ValidationResult, railMLExtensions> results4 = validator4.Validate(railMLDocument, railMLVersion);
                result = results4.Item1;
                break;
        }
        string actualErrors = null;
        if (result.errors.Count > 0)
        {
            actualErrors = result.errors.First().description;
            Assert.Contains("not allowed", actualErrors);
        }
        string actualWarnings = null;
        if (result.warnings.Count > 0)
        {
            actualWarnings = result.warnings.First().description;
            Assert.Contains("not allowed", actualWarnings);
        }
    }

    [Theory]
    [InlineData("XDocument", @"\TestData\Examples\2_AttributeWithSchema_NotAllowed_NotDefined.xml")]
    [InlineData("XMLDocument", @"\TestData\Examples\2_AttributeWithSchema_NotAllowed_NotDefined.xml")]
    [InlineData("XMLReader", @"\TestData\Examples\2_AttributeWithSchema_NotAllowed_NotDefined.xml")]
    [InlineData("XDocumentSolution", @"\TestData\Examples\2_AttributeWithSchema_NotAllowed_NotDefined.xml")]
    public void ValidationSituation6WithSchemaText_Test(string ValidationMethod, string filePath)
    {
        string baseDirectory = AppContext.BaseDirectory;
```

```
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;

        railMLDocument railMLDocument = new railMLDocument();

        railMLVersion railMLVersion = new railMLVersion();

        Prevalidation prevalidation = new Prevalidation();

        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);

        ValidationResult result = new ValidationResult(railMLDocument);

        switch (ValidationMethod)

        {

           case "XDocument":

              IValidator validator1 = new ValidationXDocument();

              Tuple<ValidationResult, railMLExtensions> results1 = validator1.Validate(railMLDocument, railMLVersion);

              result = results1.Item1;

              break;

           case "XMLDocument":

              IValidator validator2 = new ValidationXMLDocument();

              Tuple<ValidationResult, railMLExtensions> results2 = validator2.Validate(railMLDocument, railMLVersion);

              result = results2.Item1;

              break;

           case "XMLReader":

              IValidator validator3 = new ValidationXMLReader();

              Tuple<ValidationResult, railMLExtensions> results3 = validator3.Validate(railMLDocument, railMLVersion);

              result = results3.Item1;

              break;

           case "XDocumentSolution":

              IValidator validator4 = new ValidationXDocumentSolution();

              Tuple<ValidationResult, railMLExtensions> results4 = validator4.Validate(railMLDocument, railMLVersion);

              result = results4.Item1;

              break;

        }

        string actualErrors = null;

        if (result.errors.Count > 0)

        {

           actualErrors = result.errors.First().description;

           Assert.Contains("not declared", actualErrors);

        }

        string actualWarnings = null;

        if (result.warnings.Count > 0)

        {

           actualWarnings = result.warnings.First().description;

           Assert.Contains("not declared", actualWarnings);

        }

     }



     [Theory]

     [InlineData("XDocument", @"\TestData\Examples\3_ElementAttribute_Allowed.xml")]

     [InlineData("XMLDocument", @"\TestData\Examples\3_ElementAttribute_Allowed.xml")]

     [InlineData("XMLReader", @"\TestData\Examples\3_ElementAttribute_Allowed.xml")]

     [InlineData("XDocumentSolution", @"\TestData\Examples\3_ElementAttribute_Allowed.xml")]

     public void ValidationSituation7Full_Test(string ValidationMethod, string filePath)

     {

        string baseDirectory = AppContext.BaseDirectory;

        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;

        railMLDocument railMLDocument = new railMLDocument();

        railMLVersion railMLVersion = new railMLVersion();

        Prevalidation prevalidation = new Prevalidation();

        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);

        ValidationResult result = new ValidationResult(railMLDocument);

        switch (ValidationMethod)

        {

           case "XDocument":

              IValidator validator1 = new ValidationXDocument();

              Tuple<ValidationResult, railMLExtensions> results1 = validator1.Validate(railMLDocument, railMLVersion);

              result = results1.Item1;

              break;

           case "XMLDocument":

              IValidator validator2 = new ValidationXMLDocument();

              Tuple<ValidationResult, railMLExtensions> results2 = validator2.Validate(railMLDocument, railMLVersion);

              result = results2.Item1;

              break;

           case "XMLReader":

              IValidator validator3 = new ValidationXMLReader();

              Tuple<ValidationResult, railMLExtensions> results3 = validator3.Validate(railMLDocument, railMLVersion);

              result = results3.Item1;

              break;

           case "XDocumentSolution":

              IValidator validator4 = new ValidationXDocumentSolution();
```

```
            Tuple<ValidationResult, railMLExtensions> results4 = validator4.Validate(railMLDocument, railMLVersion);
            result = results4.Item1;
            break;
        }
        bool errorsPresent = false;
        if (result.errors.Count() > 0)
        {
            errorsPresent = true;
        }
        Assert.True(errorsPresent);
        bool typePresent = false;
        foreach (var error in result.errors)
        {
            if (error.description.Contains("invalid xsi:type"))
            {
                typePresent = true;
            }
        }

        Assert.True(typePresent);
    }

}

public class UnitTestExtensions
{
    [Theory]
    [InlineData(@"\TestData\Ostsachsen_V220_Good.railml", false)]
    [InlineData(@"\TestData\Ostsachsen_V200.xml", false)]
    [InlineData(@"\TestData\Ostsachsen_V210.xml", false)]

[InlineData(@"\TestData\161115_BaneNor_NorwayRailNetwork.railml", false)]

[InlineData(@"\TestData\090318_Bahnkonzept_ExampleDataGPSinfradat_NorthUpperRhineNetworkRailML23.railml", true)]

[InlineData(@"\TestData\OpenTrack_to_railML.V_2.2.rollingstock.railml", false)]

[InlineData(@"\TestData\OpenTrack_to_railML.V_2.2.timetable.railml", false)]
    [InlineData(@"\TestData\SimpleExtensionExample.xml", true)]

[InlineData(@"\TestData\SimpleExtensionOneDeclaredOneUndeclared.xml", true)]
    [InlineData(@"\TestData\SimpleExample.xml", false)]
    [InlineData(@"\TestData\SimpleExtensionExample2.xml", true)]
    public void ExtensionsDetected_Test(string filePath, bool present)
    {
        string baseDirectory = AppContext.BaseDirectory;
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;
        railMLDocument railMLDocument = new railMLDocument();
        Prevalidation prevalidation = new Prevalidation();
        railMLVersion railMLVersion = new railMLVersion();
        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);
        railMLExtensions railMLExtensions = new railMLExtensions();

railMLExtensions.FindAndSetExtensions(railMLDocument.document, null);
        bool presentActual = false;
        foreach (var kvp in railMLExtensions.Extensions)
        {
            var namespaceName = kvp.Key;
            var elementsList = kvp.Value.Item2;
            if (elementsList.Count > 0)
            {
                presentActual = true;
            }
        }
        Assert.Equal(present, presentActual);
    }

    [Theory]
    [InlineData(@"\TestData\SimpleExample.xml", 0)]
    [InlineData(@"\TestData\SimpleExtensionExample.xml", 1)]

[InlineData(@"\TestData\SimpleExtensionOneDeclaredOneUndeclared.xml", 2)]
    public void ExtensionsNamespaces_Test(string filePath, int expectedNamespaces)
    {
        string baseDirectory = AppContext.BaseDirectory;
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;
        Prevalidation prevalidation = new Prevalidation();
        railMLDocument railMLDocument = new railMLDocument();
        railMLVersion railMLVersion = new railMLVersion();
        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);
        railMLExtensions railMLExtensions = new railMLExtensions();
```

```
railMLExtensions.FindAndSetExtensions(railMLDocument.document, null);
        int actualNamespaces = railMLExtensions.Extensions.Count();
        Assert.Equal(expectedNamespaces, actualNamespaces);
    }

    [Fact]
    public void ExtensionsSchemas_Test()
    {
        string filePath = @"\TestData\SimpleExtensionOneDeclaredOneUndeclared.xml";
        string expectedShema1 = "SimpleExtensionSchema.xsd";
        string expectedShema2 = null;
        string baseDirectory = AppContext.BaseDirectory;
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;
        Prevalidation prevalidation = new Prevalidation();
        railMLDocument railMLDocument = new railMLDocument();
        railMLVersion railMLVersion = new railMLVersion();
        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);
        railMLExtensions railMLExtensions = new railMLExtensions();

railMLExtensions.FindAndSetExtensions(railMLDocument.document, null);
        var matchingEntry1 = railMLExtensions.Extensions.FirstOrDefault(kvp => kvp.Key == "https://www.my-company.com/my-department/");
        string ActualSchema1 = matchingEntry1.Value.Item1;
        var matchingEntry2 = railMLExtensions.Extensions.FirstOrDefault(kvp => kvp.Key == "https://www.my-company.com/my-other-department/");
        string ActualSchema2 = matchingEntry2.Value.Item1;
        Assert.Equal(expectedShema1, ActualSchema1);
        Assert.Equal(expectedShema2, ActualSchema2);
    }

    [Fact]
    public void ExtensionsResult1_Test()
    {
        string filePath = @"\TestData\SimpleExtensionOneDeclaredOneUndeclared.xml";
        string baseDirectory = AppContext.BaseDirectory;
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;
        Prevalidation prevalidation = new Prevalidation();
        railMLDocument railMLDocument = new railMLDocument();
        railMLVersion railMLVersion = new railMLVersion();
        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);
        ValidationXDocumentSolution methodOne = new ValidationXDocumentSolution();
        Tuple<ValidationResult, railMLExtensions> results = methodOne.Validate(railMLDocument, railMLVersion);
        railMLExtensions railMLExtensions = results.Item2;
        int iter = 0;
        foreach (var kvp in railMLExtensions.Extensions)
        {
            var namespaceName = kvp.Key;
            var elementsList = kvp.Value.Item2;

            foreach (var element in elementsList)
            {
                switch (iter)
                {
                    case 0:
                        Assert.Equal("SAP-Number", element.Name);
                        Assert.Equal(8, element.Position);
                        Assert.True(element.Validated);
                        break;
                    case 1:
                        Assert.Equal("variant", element.Name);
                        Assert.Equal(8, element.Position);
                        Assert.True(element.Validated);
                        break;
                    case 2:
                        Assert.Equal("something", element.Name);
                        Assert.Equal(10, element.Position);
                        Assert.False(element.Validated);
                        break;
                    case 3:
                        Assert.Equal("something", element.Name);
                        Assert.Equal(11, element.Position);
                        Assert.False(element.Validated);
                        break;
                }
                iter++;
            }

        }
    }

    [Fact]
    public void ExtensionsResult2_Test()
```

```
    {
        string filePath = @"\TestData\SimpleExtensionExample.xml";
        string baseDirectory = AppContext.BaseDirectory;
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;
        Prevalidation prevalidation = new Prevalidation();
        railMLDocument railMLDocument = new railMLDocument();
        railMLVersion railMLVersion = new railMLVersion();
        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);
        ValidationXDocumentSolution methodOne = new ValidationXDocumentSolution();
        Tuple<ValidationResult, railMLExtensions> results = methodOne.Validate(railMLDocument, railMLVersion);
        railMLExtensions railMLExtensions = results.Item2;

        int iter = 0;
        foreach (var kvp in railMLExtensions.Extensions)
        {
            var namespaceName = kvp.Key;
            var elementsList = kvp.Value.Item2;

            foreach (var element in elementsList)
            {
                switch (iter)
                {
                    case 0:
                        Assert.Equal("SAP-Number", element.Name);
                        Assert.Equal(7, element.Position);
                        Assert.True(element.Validated);
                        break;
                    case 1:
                        Assert.Equal("variant", element.Name);
                        Assert.Equal(7, element.Position);
                        Assert.True(element.Validated);
                        break;
                    case 2:
                        Assert.Equal("colour", element.Name);
                        Assert.Equal(8, element.Position);
                        Assert.False(element.Validated);
                        break;
                    case 3:
                        Assert.Equal("colour", element.Name);
                        Assert.Equal(9, element.Position);
                        Assert.True(element.Validated);
                        break;
                }
                iter++;
            }

        }

    }
}

public class UnitTestGeneralInformation
{
    [Theory]
    [InlineData(@"\TestData\Ostsachsen_V210.xml", 6)]
[InlineData(@"\TestData\090318_Bahnkonzept_ExampleDataGPSinfradat_NorthUpperRhineNetworkRailML23.railml", 5)]
[InlineData(@"\TestData\OpenTrack_to_railML.V_2.2.rollingstock.railml", 2)]
[InlineData(@"\TestData\OpenTrack_to_railML.V_2.2.timetable.railml", 2)]
    public void UnitTestMetadataAmount(string filePath, int expectedAmount)
    {
        string baseDirectory = AppContext.BaseDirectory;
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;
        Prevalidation prevalidation = new Prevalidation();
        railMLDocument railMLDocument = new railMLDocument();
        railMLVersion railMLVersion = new railMLVersion();
        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);
        Metadata metadata = new Metadata();
        metadata.LoadMetadata(railMLDocument.document);
        int actualAmount = metadata.dataPairs.Count;
        Assert.Equal(expectedAmount, actualAmount);

    }

    [Theory]
    [InlineData(@"\TestData\Ostsachsen_V210.xml", "2.1.0", "2016-04-13T20:08:25", "4", "iRFP", "1252  (ANSI - Lateinisch I)",
        "iPLAN.exe V1.5.2.851 NtzIntf_RailML2.dll V2.2.4.68")]
    public void UnitTestMetadataFullData(string filePath, string exFormat, string exDate, string exIdentifier, string exCreator,
        string exLanguage, string exSource)
    {
        string baseDirectory = AppContext.BaseDirectory;
```

```
        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;

        Prevalidation prevalidation = new Prevalidation();

        railMLDocument railMLDocument = new railMLDocument();

        railMLVersion railMLVersion = new railMLVersion();

        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);

        Metadata metadata = new Metadata();

        metadata.LoadMetadata(railMLDocument.document);

        Assert.Equal(exFormat, metadata.dataPairs["Format"]);

        Assert.Equal(exDate, metadata.dataPairs["DateTime"]);

        Assert.Equal(exIdentifier, metadata.dataPairs["Identifier"]);

        Assert.Equal(exCreator, metadata.dataPairs["Creator"]);

        Assert.Equal(exLanguage, metadata.dataPairs["Language"]);

        Assert.Equal(exSource, metadata.dataPairs["Source"]);



    }


    [Theory]

    [InlineData(@"\TestData\Ostsachsen_V210.xml", 4)]


[InlineData(@"\TestData\090318_Bahnkonzept_ExampleDataGPSinfradat_NorthUpperRhineNetworkRailML23.railml", 2)]


[InlineData(@"\TestData\OpenTrack_to_railML.V_2.2.rollingstock.railml", 2)]


[InlineData(@"\TestData\OpenTrack_to_railML.V_2.2.timetable.railml", 4)]

    public void UnitTestFileContentsAmount(string filePath, int expectedAmount)

    {

        string baseDirectory = AppContext.BaseDirectory;

        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;

        Prevalidation prevalidation = new Prevalidation();

        railMLDocument railMLDocument = new railMLDocument();

        railMLVersion railMLVersion = new railMLVersion();

        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);

        FileContents contents = new FileContents();

        contents.LoadContent(railMLDocument.document);

        int actualAmount = contents.Contents.Count;

        Assert.Equal(expectedAmount, actualAmount);

    }


    [Theory]

    [InlineData(@"\TestData\Ostsachsen_V210.xml")]

    public void UnitTestFileContentsFullContent(string filePath)

    {

        List<string> possibleContent = new List<string>() { "metadata", "infrastructure", "rollingstock", "timetable" };

        string baseDirectory = AppContext.BaseDirectory;

        string projectDirectory = Directory.GetParent(baseDirectory).Parent.Parent.Parent.FullName;

        Prevalidation prevalidation = new Prevalidation();

        railMLDocument railMLDocument = new railMLDocument();

        railMLVersion railMLVersion = new railMLVersion();

        prevalidation.Prevalidate(projectDirectory + filePath, out railMLDocument, out railMLVersion);

        FileContents contents = new FileContents();

        contents.LoadContent(railMLDocument.document);

        foreach(var element in contents.Contents)

        {

            Assert.Contains(element, possibleContent);

        }

    }

  }

}
```

1.23 Текст файлу Form1.cs модуля Validator

```
namespace Validator
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
    }

    private void buttonValidate_Click(object sender, EventArgs e)
    {
      if (label1.Text.Length == 0)
      {
        MessageBox.Show("Please choose file!");
      }else if(comboBox1.SelectedIndex == -1)
      {
        MessageBox.Show("Please choose validation method!");
      }
      else
      {
        UIController uIController = new UIController();

uIController.OpenValidator(comboBox1.GetItemText(comboBox1.SelectedItem), textBox1.Text);
      }

    }

    private void button1_Click(object sender, EventArgs e)
    {
      using (OpenFileDialog openFileDialog = new OpenFileDialog())
      {
        openFileDialog.ShowDialog();
```

```
            textBox1.Text = openFileDialog.FileName;
        }
    }
```

1.24 Текст файлу UIController.cs модуля Validator

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using railVIVID.Validation;
using railVIVID.Validation.GeneralInformation;
using railVIVID.Validation.XDocument;
using railVIVID.Validation.XMLDocument;
using railVIVID.Validation.XMLReader;
using railVIVID.Validation.XDocumentSolution;
using System.Drawing.Text;
using railVIVID.Validation.Extensions;

namespace Validator
{
    internal class UIController
    {
        public void OpenValidator(string type, string path)
        {
            railMLDocument railMLDocument = new
railMLDocument();
            railMLVersion railMLVersion = new railMLVersion();
            Prevalidation prevalidation = new Prevalidation();
            ValidationResult prevalResult =
prevalidation.Prevalidate(path, out railMLDocument, out
railMLVersion);
            string prevalResultText = "";
            if (prevalResult.errors.Count > 0 ||
prevalResult.warnings.Count > 0)
            {
                foreach(var error in prevalResult.errors)
                {
                    prevalResultText += error.description;

                }

                foreach (var warning in prevalResult.warnings)
                {
                    prevalResultText += warning.description;
                }

                MessageBox.Show(prevalResultText);
            }
            else
            {
                ValidationResult result = new
ValidationResult(railMLDocument);
                railMLExtensions extensions = new railMLExtensions();
                switch (type)
                {
                    case "XDocument.Validate()":
                        IValidator validator1 = new ValidationXDocument();
                        Tuple<ValidationResult, railMLExtensions> results1
= validator1.Validate(railMLDocument, railMLVersion);
                        result = results1.Item1;
                        extensions = results1.Item2;
                        break;
                    case "XMLDocument.Validate()":
                        IValidator validator2 = new
ValidationXMLDocument();
                        Tuple<ValidationResult, railMLExtensions> results2
= validator2.Validate(railMLDocument, railMLVersion);
                        result = results2.Item1;
                        extensions = results2.Item2;
                        break;
                    case "XMLReader":
                        IValidator validator3 = new ValidationXMLReader();
                        Tuple<ValidationResult, railMLExtensions> results3
= validator3.Validate(railMLDocument, railMLVersion);
                        result = results3.Item1;
                        extensions = results3.Item2;
                        break;
                    case "XDocument.Validate() Solution":
                        IValidator validator4 = new
ValidationXDocumentSolution();
                        Tuple<ValidationResult, railMLExtensions> results4
= validator4.Validate(railMLDocument, railMLVersion);
                        result = results4.Item1;
                        extensions = results4.Item2;
                        break;
                }

                HandleResult(type, path, result, extensions,
railMLDocument);
            }
        }

        private void HandleResult(string type, string path,
ValidationResult result, railMLExtensions extensions,
railMLDocument document)
        {
            ValidatorView validatorView = new ValidatorView(type,
path);
            string success;
            if(result.errors.Count == 0)
            {
                success = "Document is valid!";
            }
            else {
                success = "Document is not valid!";
            }
            string errors = GetIssues(result.errors);
            string warnings = GetIssues(result.warnings);

            string resultVal = success + "\nFound errors: " +
result.errors.Count.ToString() +
              "\nFound warnings: " + result.warnings.Count.ToString();

            string extensionsText = "";

            int i = 1;
            foreach (var keyValuePair in extensions.Extensions)
            {
                extensionsText += $"{i}. {keyValuePair.Key}";
                if(keyValuePair.Value.Item1 != null)
                {
                    extensionsText += " " +
keyValuePair.Value.Item1.ToString();
                }
                extensionsText += " contains:";
                foreach (ExtensionElement element in
keyValuePair.Value.Item2)
                {
                    string valid;
                    if (element.Validated)
                    {
                        valid = "valid";
                    }
                    else
                    {
                        valid = "not valid";
    }
}
```

```
                }
                extensionsText += $"\n\tLine {element.Position}:
attribute/element '{element.Name}' {valid}";
                extensionsText += "\n\t\tContent: " +
document.lines[element.Position - 1];
            }
            extensionsText += "\n";
            i++;
        }

        validatorView.ShowValidationResults(resultVal, errors,
warnings, extensionsText);
        validatorView.Show();
    }

    private string GetIssues(List<ValidationIssueBasic> issuesList)
    {
        string issuesText = "";
        int i = 1;
        foreach(var issue in issuesList)
        {
            issuesText += $"{i}. {issue.description}";
            if(issue is ValidationIssueDetailed issueDetailed)
            {
                foreach (var kvp in issueDetailed.lines)
                {
                    issuesText += "\n\tLine: " + kvp.Item1.ToString();
                    issuesText += "\n\t\tContent: " +
kvp.Item2.ToString();
                }
                issuesText += "\n";
            }
            else issuesText += "\n";
            i++;
        }
        return issuesText;
    }

  }
}
```

1.25 Текст файлу ValidatorView.cs модуля Validator

```
namespace Validator
{
  public partial class ValidatorView : Form
  {
    public ValidatorView()
    {
      InitializeComponent();
    }

    public ValidatorView(string type, string file)
    {
      InitializeComponent();
      this.Text = type;
      labelFileName.Text = file;
    }

    public void ShowValidationResults(string result, string errors,
string warnings, string extensions)
    {
      richTextBoxErrorCount.Text = result;
      richTextBoxErrorDesc.Text = errors;
      richTextBoxWarnings.Text = warnings;
      richTextBoxExtensions.Text = extensions;
    }

    private void ValidatorView_SizeChanged(object sender,
EventArgs e)
    {
      labelFileName.MaximumSize = new Size(this.Width - 160,
0);
    }
  }
}
```

ДОДАТОК В

Технічно завдання

ЗАТВЕРДЖУЮ
Проректор Українського державного
університету науки і технологій
Анатолій РАДКЕВИЧ

«ДОСЛІДЖЕННЯ МЕТОДІВ ВАЛІДАЦІЇ RAILML ДОКУМЕНТІВ В СЕРЕДОВИЩІ .NET»
Керівництво користувача
44165850.01451 – 01 І3 01

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Олександр ІВАНОВ
Виконавець
_____ Маргарита ВИСКАРКА
Нормоконтролер
_____Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО
44165850.1451-01 І3 01

«ДОСЛІДЖЕННЯ МЕТОДІВ ВАЛІДАЦІЇ RAILML ДОКУМЕНТІВ В СЕРЕДОВИЩІ .NET»

Керівництво користувача
44165850.1451 – 01 І3 01

Листів 9

2025

АНОТАЦІЯ

Документ 1116130.1451 – 01 І3 01 «Дослідження методів валідації railML документів в середовищі .NET. Керівництво користувача».

Застосунок написаний на мові C# з використанням стандартних бібліотек валідації у .NET у програмному середовищі Visual Studio 2022.

ЗМІСТ

1 ВВЕДЕННЯ

Програмний засіб “Дослідження методів валідації railML документів в середовищі .NET” призначений для дослідження доступних стандартних методів валідації railML документів середовища .NET та розробленого рішення на основі одного з обраних методів..

Головною метою розробки є надання можливості перегляду та порівняння результатів роботи обраних валідаторів середовища .NET.

Даний продукт призначений для використання науковцями для дослідження різних методів валідації XML та railML для визначення бібліотеки, яка найбільше відповідає вимогам railML.org.

Не вимагає ознайомлення із будь-якою додатковою експлуатаційною документацією.

2 ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

Функціональним призначенням застосунку є проведення аналізу результатів роботи стандартних методів валідації XML та railML, та створеного на основі одного з них рішення виявленої в .NET проблеми.

Експлуатаційне призначення застосунку “Дослідження методів валідації railML документів в середовищі .NET” – визначення валідатора, результати якого відповідають вимогам raiML.org.

Додаток, що розробляється, розрахований на використання на пристроях що мають:

- процесор з тактовою частотою 2 ГГЦ або вище;
- не менше 2ГБ оперативної пам’яті.

Для функціонування настільного застосунку неохідна система Windows 10 версії і вище.

3 ПІДГОТОВКА ДО РОБОТИ

Для початку роботи застосунку необхідно запустити .exe файл.

4 ОПИС ОПЕРАЦІЙ

Після встановлення та запуску застосунку на основній формі будуть присутні такі елементи:

- кнопка вибору файлу, який буде перевірятись;
- поле зі шляхом до обраного файлу;
- список доступних методів валідації, таких як XDocument, XDocumentSolution, XMLDocument та XMLReader;
- кнопка запуску процесу валідації.

Вікно перевірки результатів валідації надає таку інформацію:

- текстове поле з назвою файлу;
- текстове поле з загальним результатом валідації;
- текстове поле з кількістю помилок;
- текстове поле з списком та деталями помилок;
- текстове поле з списком та деталями попереджень;
- текстове поле з списком та деталями розширень.

5 АВАРІЙНІ СИТУАЦІЇ

Якщо програмний засіб під час роботи буде поводити некоректно чи із помилкою, необхідно перезапустити настільний застосунок для продовження роботи.

6 РЕКОМЕНДАЦІЇ ЩОДО ЗАСТОСУВАННЯ

Після встановлення застосунку на пристрій його необхідно запустити через виконуючий файл.

Після запуску застосунку перед користувачем відкриється головний екран, на якому він може обрати необхідний файл та спосіб його валідації.

Користувач має можливість паралельної перевірки результатів декількох валідаторів шляхом вибору нового іншого валідатора та повторного натиску кнопки для початку процесу перевірки файлу.