



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Model Driven Engineering 2024-2025

Assignment 4

EMF-Based Airport Management System

Name	Matriculation number	Email
Prachi	301124	prachi.prachi@student.univaq.it
Eric Rode	301145	eric.rode@student.univaq.it
David Aron Vigh	301150	davidaron.vigh@student.univaq.it

Introduction

Airport management systems are complex infrastructures that require meticulous design to manage resources such as terminals, airside areas, gates, and flights. Using model-driven engineering (MDE) principles, this project leverages EMF to define a metamodel that encapsulates the airport's operational domain. The metamodel specifies entities, relationships, and constraints, enabling efficient validation and behavior testing.

The system is developed to include derived fields and validation constraints for real-world relevance. For example, the `isInternational` attribute of a flight is derived from its associated city, and operations such as `findGateByName` encapsulate domain-specific logic.

Metamodel Design

The metamodel is specified in Ecore, representing the structural hierarchy of an airport. Key components include:

1. **Terminal:** The central entity managing various functional and spatial components, such as airside areas, landside areas, and transportation services.
2. **Airside and Landside:** Subdivisions of the terminal, responsible for operations such as gate management and passenger transitions.
3. **GateArea:** A crucial component for managing boarding gates and linking flights.
4. **Flight:** Represents individual flights with attributes like flight number, type (arrival or departure), and derived attributes like `isInternational`.
5. **Validation Rules:** Integrated into the metamodel using Object Constraint Language (OCL). For instance, `ValidTime` ensures that flight times conform to the HH:mm format.

To validate its applicability, the metamodel is instantiated in two implementations:

1. **Debrecen International Airport:** This scenario emphasizes features such as multiple floors, Schengen and non-Schengen gate areas, and comprehensive landside and airside facilities.
2. **Pescara Airport Terminal 1:** This implementation highlights a simpler structure with a focus on a single-floor terminal, basic gate area configurations, and essential transportation services.

Implementation

Model Instantiation and Object Validation

The system uses the EMF-generated factory classes to instantiate model elements. For instance, the AirportFactory is utilized to create Terminal, GateArea, Flight, and other entities. Object validation is performed using the EMF Diagnostician to ensure compliance with defined constraints.

The following code illustrates the creation and validation of model objects:

```
Terminal terminal = factory.createTerminal();  
terminal.setName("Main Terminal");  
validateObject(terminal);
```

Validation results are presented using diagnostic messages. This ensures that all objects conform to their constraints before deployment.

Derived Attributes and Operations

The metamodel includes derived attributes such as *isInternational* in the Flight class, which evaluates whether a flight is international based on its associated city. Additionally, the GateArea class features operations like *calculateTotalGates* and *findGateByName*, implemented in Java for runtime execution.

For example, the *findGateByName* operation searches for a gate with a specific name within a GateArea:

```
Gate foundGate = gateArea.findGateByName("Gate A");
```

Testing Derived Fields

A test suite validates the correctness of derived attributes and operations. The DerivedFieldsTest class demonstrates how changes in the model impact derived fields and ensures they behave as expected.

```
flight.setCity("LocalCity");  
System.out.println("Is International: " + flight.isIsInternational()); // Expected: false
```

Behavior Testing

Functional behavior is tested using unit tests, such as validating the total number of gates in a GateArea and searching for specific gates. The GateAreaTest class ensures that methods like calculateTotalGates yield accurate results.

Results and Validation

The system effectively demonstrates the utility of EMF for modeling complex domains. Validation tests confirm that constraints such as flight time format and valid counter associations are enforced. Derived attributes provide real-time insights into the system's state, ensuring accurate and consistent behavior.

Testing the GateArea operations resulted in the following outcomes:

- calculateTotalGates correctly counted gates.
- findGateByName successfully located gates by their names.

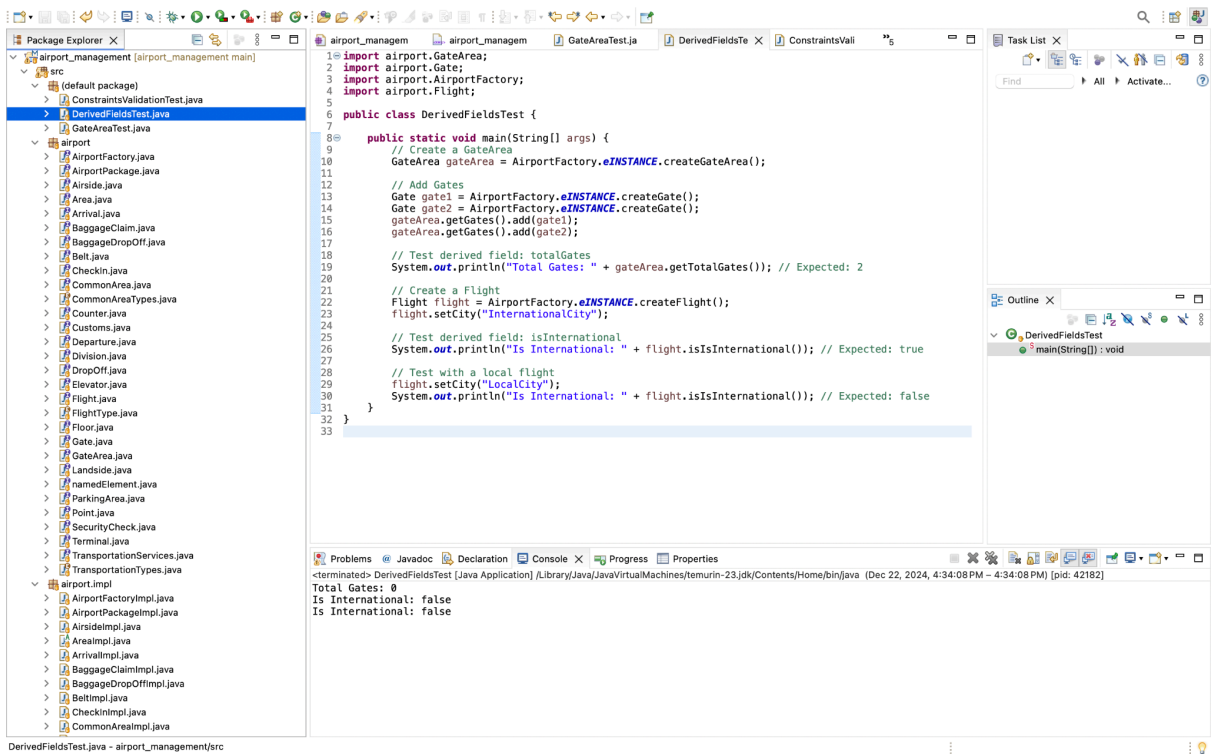
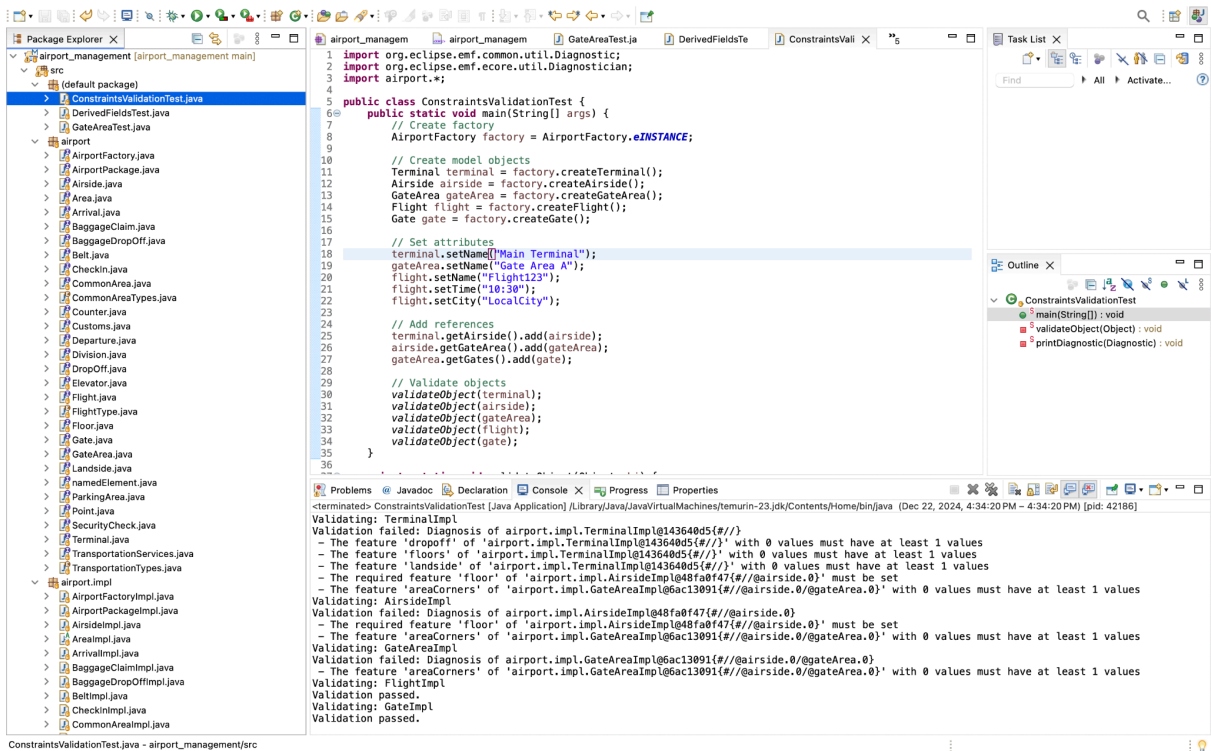
Appendix

Implemented Tests

The screenshot displays an IDE with the following components:

- Package Explorer:** Shows the project structure for `airport_management`. The `src` folder contains a `default package` with various classes. The `GateAreaTest.java` file is selected.
- Editor:** Displays the code for `GateAreaTest.java`. The code includes imports for `airport.GateArea`, `airport.Gate`, and `airport.AirportFactory`. It defines a `GateAreaTest` class with a `main` method. The `main` method performs the following steps:
 - Step 1: Create a `GateArea` instance using `AirportFactory.eINSTANCE.createGateArea()`.
 - Step 2: Create `Gate` instances using `AirportFactory.eINSTANCE.createGate()`.
 - Step 3: Add the created `Gate` instances to the `GateArea` using `gateArea.getGates().add(gate1)` and `gateArea.getGates().add(gate2)`.
 - Step 4: Invoke the `calculateTotalGates()` operation and print the result using `System.out.println("Total gates in GateArea: " + totalGates);`. The expected output is 2.
 - Step 5: Invoke the `findGateByName()` operation and print the result using `System.out.println("Found gate: " + foundGate.getName());`. The expected output is Gate A.
 - Step 6: Print the result of the `findGateByName()` operation using `System.out.println("Gate not found!");`.
- Task List:** Shows the `GateAreaTest` class and its `main(String[])` method.
- Outline:** Shows the `GateAreaTest` class and its `main(String[])` method.
- Problems:** Shows the execution results of the `GateAreaTest` class. The output is:

```
Total gates in GateArea: 2
Found gate: Gate A
Gate not found for name: Gate C
```



Implemented Models

pescara.airport X debrecen.airport

Resource Set

platform:/resource/examples/pescara.airport

Terminal Terminal 1 Pescara Airport

Drop Off Kiss&Leave

Point 0

Point 0

Point 10

Point 10

Floor GroundFloor

Airside

Common Area SittingArea

Floor GroundFloor

Gate Area Gate Area 1

Point 0

Point 0

Point 20

Point 30

Gate Gate 1

Gate Gate 2

Gate Gate 3

Gate Gate 4

Gate Gate 5

Landside

Common Area Restroom 1

Common Area Restroom 2

Common Area Restroom 3

Common Area Restroom 4

Common Area Restroom 5

Common Area DAngelo

Common Area Concorde

Common Area Iannone

Common Area Evangelista

Transportation Services Bus

Point 30































Point 30

Point 40

Point 40

- ▼ ◆ Baggage Claim Schengen arrival
 - ◆ Point 80
 - ◆ Point 80
 - ◆ Point 100
 - ◆ Point 100
 - ◆ Belt 1
- ▼ ◆ Baggage Claim Non-Schengen arrival
 - ◆ Point 100
 - ◆ Point 100
 - ◆ Point 110
 - ◆ Point 110
 - ◆ Belt 2
 - ◆ Customs Customs
- ◆ Flight LH1687
- ◆ Flight IZ702
- ◆ Flight W92701
- ◆ Flight D4703
- ◆ Flight LH1686
- ◆ Flight IZ701
- ◆ Flight W92702
- ▼ ◆ Parking Area Main Parking
 - ◆ Point 0
 - ◆ Point 20
 - ◆ Point 20
 - ◆ Point 0
- ▼ ◆ Transportation Services Taxi
 - ◆ Point 20
 - ◆ Point 20
 - ◆ Point 30
 - ◆ Point 30

- ▼ ◆ Landside
 - ▼ ◆ Departure Main Departure
 - ◆ Point 40
 - ◆ Point 40
 - ◆ Point 60
 - ◆ Point 60
 - ◆ Security Check Passesnger Security Check
 - ▼ ◆ Check In Check-in
 - ◆ Counter 1
 - ◆ Counter 2
 - ◆ Counter 3
 - ◆ Counter 4
 - ◆ Counter 5
 - ◆ Counter 6
 - ▼ ◆ Baggage Drop Off DropOff
 - ◆ Point 40
 - ◆ Point 40
 - ◆ Point 60
 - ◆ Point 60
 - ◆ Counter 1
 - ◆ Counter 2
 - ◆ Counter 3
 - ◆ Counter 4
 - ◆ Counter 5
 - ◆ Counter 6
 - ▼ ◆ Arrival Main arrival
 - ◆ Point 60
 - ◆ Point 60
 - ◆ Point 80
 - ◆ Point 80

- ✖  Terminal Debrecen International
 - ✖  Drop Off Repuloter Parkolo
 -  Point 0
 -  Point 0
 -  Point 10
 -  Point 10
 -  Floor Ground Floor
 -  Floor First Floor
- ✖  Airside
 -  Common Area Waiting Area
 -  Common Area Toilet 3
 -  Common Area Toilet 4
 -  Common Area Toilet 5
 -  Common Area Shop
 -  Common Area Snack Bar
 -  Floor Ground Floor
- ✖  Gate Area Schengen Countries
 -  Point 30
 -  Point 30
 -  Point 50
 -  Point 50
 -  Gate 1
 -  Gate 2
- ✖  Gate Area Non-Schengen Countries
 -  Point 50
 -  Point 50
 -  Point 70
 -  Point 70
 -  Gate 3
 -  Gate 4

◆ Belt 3

◆ Belt 4

◆ Customs Customs

◆ Flight FR2294

◆ Flight FR983

◆ Flight FR5013

◆ Flight W45086

◆ Flight FR982

◆ Flight FR2293

◆ Flight FR5016

◆ Flight W45085

▼ ◆ Parking Area

◆ Point 0

◆ Point 0

◆ Point 0

◆ Point 60

▼ ◆ Transportation Services Taxi

◆ Point 0

◆ Point 0

◆ Point 60

◆ Point 70

▼ ◆ Transportation Services Bus

◆ Point 0

◆ Point 0

◆ Point 70

◆ Point 80

▼ ◆ Transportation Services Train

◆ Point 0

◆ Point 0

◆ Point 60

◆ Point 80

- ◆ Common Area SittingArea

- ◆ Floor GroundFloor

- ▼ ◆ Departure Departure 1

- ◆ Point 0

- ◆ Point 0

- ◆ Point 20

- ◆ Point 20

- ◆ Security Check Security

- ▼ ◆ Check In CheckIn

- ◆ Counter 1

- ◆ Counter 2

- ◆ Counter 3

- ▼ ◆ Baggage Drop Off

- ◆ Point 0

- ◆ Point 0

- ◆ Point 20

- ◆ Point 20

- ◆ Counter 1

- ◆ Counter 2

- ◆ Counter 3

- ◆ Counter 4

- ▼ ◆ Arrival Arrival 1

- ◆ Point 30

- ◆ Point 50

- ◆ Point 30

- ◆ Point 50

- ▼ ◆ Baggage Claim BC 1

- ◆ Point 30

- ◆ Point 50

- ◆ Point 30

- ◆ Point 50