Rail to Digital automated up to autonomous train operation

D31.5 – Yard to station automatic movement model report

Due date of deliverable: 30/06/2026

Actual submission date: 13/03/2026

Leader/Responsible of this Deliverable: Thomas JOUSSELIN (SNCF)

Reviewed: Y

Document status		
Revision	Date	Description
01	16/02/2026	1st draft version for WP31.1 review
02	02/03/2026	2nd version for WP31.1 review
03	13/03/2026	3rd version for TMT review
04	31/07/2026	4th version after External review

Project funded from the European Union's Horizon Europe research and innovation programme		
Dissemination Level		
PU	Public	

Start date: 01/12/2022 Duration: 42 months

ACKNOWLEDGEMENTS

This project has received funding from the Europe's Rail Joint Undertaking (ERJU) under the Grant Agreement no. 101102001. The JU receives support from the European Union's Horizon Europe research and innovation programme and the Europe's Rail JU members other than the Union.

REPORT CONTRIBUTORS

Name	Company	Details of Contribution
Thomas JOUSSELIN	SNCF Voyageurs	Leader, Author Modeling process, Speed profile optimization, Speed profile tracking, model architecture
Maxence BOURHIS	SNCF Voyageurs	Author Signal detection
Matthieu THIBAULT	SNCF Voyageurs	Contributor Modelling process, ATO architecture, Architecture of model and driving simulator
Denis CROIZE	SNCF Voyageurs	Contributor Multiphysics modelling with SIMPACK
Fabien BELLEMIN	SNCF Voyageurs	Reviewer SNCF ATO projects, ATO architecture
Nicolas BIED-CHARRETON	SNCF Voyageurs	Reviewer MBD, prototyping
Martin TEBOUL	SHERPA	Contributor Signal converter
Simon FUNKE	DB Cargo	Reviewer
Xiaolu RAO	SBB	Reviewer

Disclaimer

EXECUTIVE SUMMARY

Engineering has constantly evolved its methods to meet a dual requirement: mastering increasing complexity and reducing lead times, costs, and risks, while increasing the level of quality.

In this context, the Model-Based Design (MBD) approach provides a particularly suitable response. Based on the virtual representation of the system and scenario simulation, it allows for better control of the system development, throughout the life cycle. In the specification phase, modeling – coupled with MBSE – helps clarify the system architecture, detect specification gaps or inconsistencies that would otherwise be identified much later, with a negative impact on deadlines and costs. In design and validation, MBD allows for tests with a high level of physical representativeness (Processor-in-the-Loop, Hardware-in-the-Loop) while reducing the logistical, safety, and cost constraints associated with field tests.

This approach is especially relevant for complex systems like the European Rail Traffic Management System (ERTMS), intended to ensure the interoperability of trains in Europe. Alongside the development of the ERTMS system, complementary work is being carried out on key innovations – geolocation, automation, FRMCS (Future Railway Mobile Communication System)– to improve system reliability, capacity (train frequency) and energy performance.

Still uncommon in the railway sector, the MBD approach nevertheless offers many advantages, of which this document aims to be a concrete testimony. Our approach is based on MBD principles and on application cases related to ATO (Automatic Train Operation) GoA2 and GoA4, developed from existing specifications and documentation. The GoA2 (Grade of Automation 2) use case considers a commercial mission between two stations, with the objective of optimizing train energy consumption. The GoA4 use case focuses on a technical movement between a railway depot and a station. Other models have also been developed to illustrate multiphysics modeling tools as well as tools dedicated to model testing and validation. The document also provides example on freight trains modelling, using multiphysics tools.

Deliverable D31.4 presented the target MBD approach, the chosen system architecture, and the main operational scenarios envisaged for these use cases. This document describes the modeling approach actually carried out, the results obtained, and the key lessons learned on MBD framework and on the specific ATO case.

Future work may aim to further increase the realism of the models, both from the point of view of equipment and environment representation and the diversity and refinement of the operational scenarios studied. Emphasis may also be placed on strengthening collaboration with industry stakeholders to align modeling practices with emerging ERTMS and ATO standards.

ABBREVIATIONS AND ACRONYMS

ADM	Automatic Driving Module
APM	Automatic Processing Module
ATO	Automatic Train Operation
ATP	Automatic Train Protection
ATSM	Automatic Train Stopping Management
CCS	Control Command Signalling
EBD	Emergency Brake Deceleration
EBI	Emergency Brake Intervention
ERTMS	European Rail Traffic Management System
ETCS	European Train Control System
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
FRMCS	Future Railway Mobile Communication System
GoA	Grade of Automation
MBD	Model Based Design
MBSE	Model Based System Engineering
MPC	Model Predictive Control
OSP	Operational Speed Profile
R2DATO	Rail to Digital automated up to autonomous train operation
RSM	Release Speed Monitoring
SCV	Signal Converter
SD	Signal Detection
SE	System Engineering
SSEM	Supervised Speed Envelope Management
TCMS	Train Control and Management System
TSI	Technical Specification for Interoperability
TTSM	TimeTable Speed Management
WP	Work Package

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The deployment of the European Train Control System (ETCS), driven by European initiatives, is steadily advancing. Successive updates to ETCS standardization have led to the release of Baseline 4, formalized in the Technical Specifications for Interoperability (TSI) published in 2023. A pivotal development in this version is the introduction of specifications for the Automatic Train Operation (ATO) system, specifically for Grade of Automation Level 2 (GoA2). Under GoA2, trains operate autonomously but remain under the supervision of a driver in the cab. While GoA4—where driving is fully automated without an onboard driver—has not yet been standardized, both systems aim to optimize train operations by delivering significant operational gains.

These gains include improvements in energy efficiency, line capacity, adherence to timetables, and, for GoA4, reduced human intervention. However, ATO systems also introduce substantial complexity. They integrate both onboard and ground-based components, involve multiple stakeholders, and address safety-critical functions. They fundamentally reshape train operation, bringing major considerations and complexity. For instance, GoA2 raises challenges related to human factors, such as the role of the driver in supervising automated functions. Meanwhile, GoA4 presents other hurdles, including environmental monitoring—such as detecting trackside signaling and obstacles—where traditional human oversight is absent.

Modeling involves replicating a real-world system through a simplified representation that captures its key characteristics and dynamics. Simulation, in turn, enables the temporal analysis of this model, illustrating how the system evolves over time. The earliest computational modeling efforts emerged during World War II for simulating neutron behavior, a phenomenon whose experimental study was too complex and costly to implement**.** Since these pioneering simulations, advancements in computational capacity have democratized this approach, unlocking major advantages: reduced development time and cost, and mitigated risk. Beyond serving as a technical tool, the model acts as a shared reference for all stakeholders, enabling them to align their perspectives throughout the development cycle and detect inconsistencies or divergent interpretations of system behavior at an early stage. These benefits are particularly pronounced for large-scale, complex systems, where the proliferation of subsystems, interfaces, and intricate dynamics makes it difficult to grasp the system as a whole.

While modeling and simulation have become standard practice in industries such as automotive engineering, their adoption in the railway sector remains limited. Yet, these methods are particularly well-suited to complex and innovative systems like Automatic Train Operation (ATO), where they could provide critical insights into system behavior, performance optimization, and risk management.

1.1 PURPOSE

WP31 aims to demonstrate the benefits of modeling through ATO-related use cases, in addition to the ERTMS/ETCS L3 use case addressed by WP30. Additionally, WP31 adopts a comprehensive approach spanning the entire development lifecycle, illustrating how Model-Based Design (MBD) can significantly reduce development costs and timelines. The focus is on a primarily software-based prototype (TRL5), which allows for flexible system representation and adaptable testing, particularly in scenario parameterization. WP31 modelling work offers an intermediate level of representativeness between the specifications (WP6) and the demonstrator developed for this use case (WP38). Indeed, modelling enables the specifications to be challenged through

implementation, and allows for more flexible and extensive validation than what is achievable through field testing alone. The methodology described in this work can thus serve as a bridge between specification work and demonstrator development.

The goal of the work carried out in this subtask, which is part of the Task 31.1, is to provide a concrete example of Model-Based Design (MBD) methodology applied across the full development cycle and illustrated in the context of ATO onboard systems, using the use case of a passenger mission for GoA2 and a technical movement for GoA4. It not only demonstrates the tangible benefits of this framework but also aims at offering actionable guidance on best practices, potential pitfalls, and key success factors for deploying MBD.

1.2 SCOPE OF THE DELIVERABLE

The purpose of this deliverable is to provide a practical example of applying the MBD methodology to two key use cases : ATO GoA2 for passenger missions and ATO GoA4 for yard to station movement. Freight train scenarios are not the focus, as they are already covered in the other subtask of Task 31.1 (see deliverable D31.1).

This document centers on the MBD methodology, demonstrating how model components developed for the ATO GoA2 use case can be reused in GoA4 model development. The primary objective of this work is not to compare potential architectures, propose new specifications, or delve into implementation details.

This deliverable (D31.5) follows up on D31.4, which conducted a system-level analysis using Model-Based Systems Engineering (MBSE) tools to define the target architecture for modeling and identify the test cases required, based on the intended scope of the models. D31.4 serves as the foundational input for this deliverable, which presents the implementation of the specified models, their validation based on simulation results, and the key lessons learned from the deployed workflow.

1.3 STRUCTURE OF THE DELIVERABLE

The first section of this document revisits the use cases considered for ATO modeling, as well as the target architecture defined for the models, in alignment with Deliverable D31.4. The second section delves into the various aspects of MBD addressed in this work, demonstrating how they contribute to enhancing performance across the development cycle. Finally, the third section presents the results of the models developed within the project, validating their behaviour and highlighting the observed benefits. The development of these models also provided an opportunity to test a range of MBD tools, including dynamic system modeling, prototyping, test and verification tools, co-simulation, and multiphysics modeling. This process enabled the identification of their respective strengths and limitations, offering valuable insights for future applications.

2 CONTEXT

This chapter aims to introduce the context of the modelling task with the use cases considered and the targeted ATO structure.

2.1 USE CASES

There are different levels of automation for rolling stock, called GoA (Grade of Automation):

- GoA1: The driving is manual and there is an Automatic Train Protection (ATP) system whose role is to protect the train from the risks of overspeed and collision with another train by automatically triggering emergency braking if necessary.
- GoA2: In addition to the presence of ATP, the traction/braking function of the train is automated thanks to ATO. The driver is still present to handle degraded situations and monitor the presence of obstacles on the track. He also has the responsibility to activate (engage) ATO or not.
- GoA3: The driving functions are fully automated throughout the entire mission. The train analyses its environment and signalling data to make driving decisions. However, a staff member is present onboard to tackle deteriorated situations.
- GoA4: Like in GoA3, all driving functions are automated but there is no staff member onboard.

ATO is expected to enhance performance on energy efficiency, punctuality, network stability and, for GoA3 and GoA4 levels, driving resource optimization (see deliverable D31.4 4.1.1 for more detailed analysis).

For ATO GoA2, it has been chosen to analyse the energy benefit an ATO GoA2 can represent. This benefit comes from the driving behaviour optimization particularly when the speed limit is changing and when stopping at a station. Thus, the GoA2 modelling use case is a commercial mission on medium traffic line with multiple stops at station.

ATO GoA4 builds upon ATO GoA2 but with the additional benefit of driving resource optimization. In this perspective, a frequently considered use case is the movement from depot (maintenance facility situated near a station) to station. Today, these technical movements are carried out by specialized drivers who are entitled to this task or by mainline drivers. GoA4 is often contemplated in this use case because it could free drivers for commercial missions and the technical implementation seems less complex than on main lines (short distances, low speed, few level crossings). (see deliverable D31.4 4.1.3 for more detailed analysis) This use case was used for the GoA4 model that has been developed.

This work focuses on ATO-OB modelling for a single train, providing a detailed analysis of its dynamic behaviour, control strategies, and energy optimization within a controlled environment. It

excludes multi-agent interactions, which are critical to studying the system-level performance of ATO.

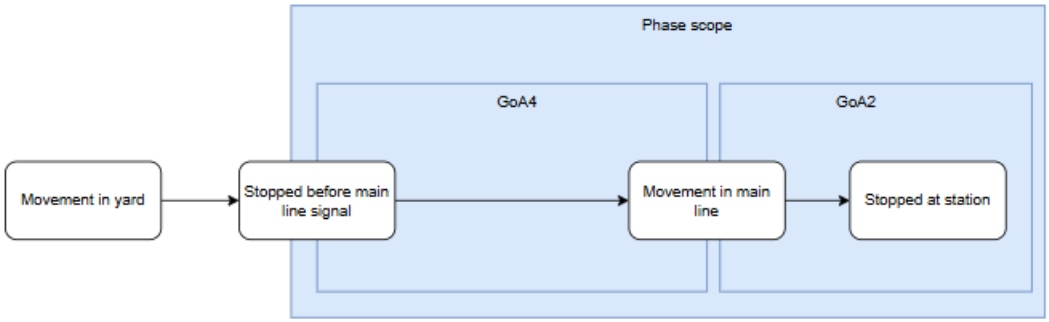


Figure 1: Modelling use case

2.2 ATO MODELS TARGETED ARCHITECTURE

2.2.1 Limitations

The purpose of this document is to illustrate modelling techniques throughout the example of ATO up to GOA4 for a specific and national use case (movement from yard to station based on national lateral signalling). This is not to be considered as an architectural solution to be further elaborated on a European level.

The example demonstrates the effectiveness of the proposed modelling techniques, but is not a specification of a GoA4 architecture.

2.2.2 ATO GoA2

The objective of this model is to quantify the energy savings enabled by an ATO GoA2, focusing on automated traction and braking in nominal operation.

One of the major strengths of modelling is to be able to take hypotheses to simplify the system, considering the targeted output of the model. For this model, several assumptions were considered:

- Signaling and Movement Authority management excluded: The subset 125 describes three distinct profiles the ATO-OB has to consider to elaborate the command value:
    - TTSM (Time Table Speed Management): a profile aimed at optimizing energy consumption while respecting timetable constraints (mission), track and signalling conditions, and the train's performance capabilities.
    - ATSM (Automatic Train Stopping Management): a profile designed to ensure precise stopping at designated stopping points (stations).
    - SSEM (Supervised Speed Envelope Management): a profile that constrains the train's speed to prevent intervention by the ETCS, considering Temporary Speed Restrictions (TSRs) and the Movement Authority (MA) through the calculation of braking curves, as performed by the ETCS.

  As the study focuses on energy performance under nominal traffic without perturbation, signaling and movement authorities mechanisms were not implemented as an input of ATO-OB, and only one train is modeled. Thus, ATO-OB builds up the command value only thanks to TTSM and ATSM.

- Static ATO-TS data: packets from ATO-TS to ATO-OB (e.g., Speed Profile, Journey Profile) are assumed to be constant throughout the simulation.
- The model focuses on the applicative part of the system with only the packets that provide useful data for ATO-OB command elaboration. No deeper OSI layer was modelled.
- Train modeling: The train is assumed to be a punctual mass object subject to gravity, tractive and braking forces and resistance to motion (due to friction with the rails and air). Train dynamics is simply given by fundamental principal of dynamics. The traction (resp. braking) effort repartition among the various traction (resp. braking) actuators has not been modelled.
- Real-mission configuration: The model uses an actual operational mission, including line speed limits, gradients, station stops with timetable constraints, and rolling-stock characteristics.
- Auxiliaries' energy consumption: auxiliaries energy consumption was not included in the model as ATO GoA2 only impacts consumption due to traction/braking commands.
- ATP is out of the modelling scope.

Functional architecture for ATO GoA2 model is given on Figure 2, with 4 major modules:

- ATO-OB is divided into two functional subsystems
  - Operational Speed Profile Generation: Based on mission, track and rolling stock data, this module calculates TTSM and ATSM profiles. The resulting profile is energy efficient and guarantees the validation of timing point constraints. As its input data are assumed static, this module only has to be run one time per simulation.
  - Profile tracking: The aim of this model is to choose a relevant value for the command to track the optimal speed profile.
- Train Model
  - Train motion: longitudinal dynamics including resistances (basic Davis formulation), gradient effects, and traction and braking characteristics (maximum force depending on the train speed).
  - Energy model: computes consumed and recovered energy (if regenerative braking is available), considering efficiencies of propulsion and traction / braking effort.
- ATO-TS
  - Static Journey profile and Segment profile are available for ATO-OB
- Train data
  - Train data including traction and braking characteristics, train mass, Davis force coefficient and traction efficiency, is available for both ATO-OB (for profile generation and tracking) and train model. Thus, the ATO-OB has access to the exact dynamic behavior of the train.

The interfaces between the major building blocks of the model are in keeping with the subsets of the STI-CCS that define them.

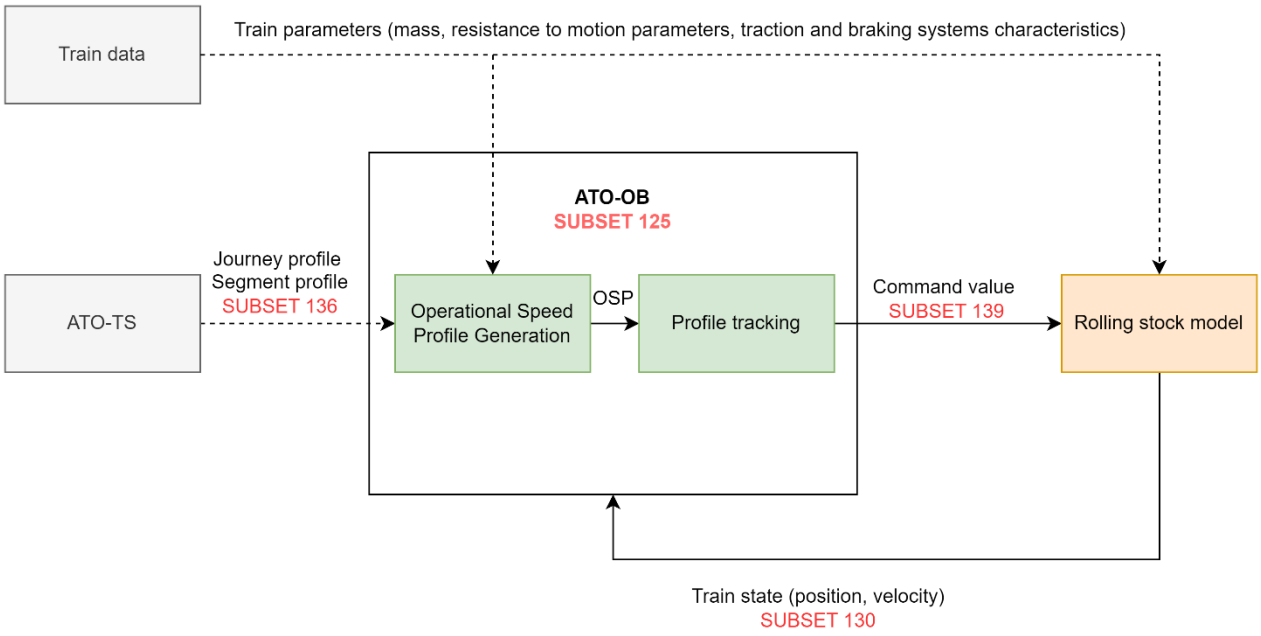


Figure 2: ATO GoA2 architecture for modelling

2.2.3 ATO GoA4

The ATO GoA4 simulation model follows a functional architecture composed of five main modules: PER (Perception), SCV (Signal Converter), ADM (Automatic Driving Module), train model, and APM (Automatic Processing Module), complemented by a data repository (REP). Figure 3 illustrates this architecture and the main data flows between modules.

- The **PER (Perception)** module is responsible for detecting lateral signaling objects, corresponding to the SD block in the WP6 architecture. It relies exclusively on image processing and does not perform any other type of detection, such as obstacle detection. As a consequence, it addresses only one of the perception-system use cases identified by WP5 and does not implement a multimodal fusion of heterogeneous sensors (e.g. LiDAR and camera).
- The **SCV (Signal Converter)** module processes the signaling information perceived from the trackside and converts it into a Movement Authority (MA) and associated data. At its output, a Subset-130 adapter formats this information according to the Subset-130 interface to be consumed by the ADM module.
- The **ADM (Automatic Driving Module)** computes the traction and braking commands applied when ATO GoA4 is engaged. It is decomposed into two sub-modules.
    - The Operational Speed Profile Generation sub-module uses signaling, mission, track and rolling-stock data to compute TTSM, ATSM and SSEM profiles, from which it builds the Operational Speed Profile (OSP). The OSP ensures energy efficiency, compliance with timing constraints, and safety by respecting lineside signaling information. The OSP is recomputed whenever conditions change, for example when the Movement Authority is extended.
      Assumption: The release speed was excluded from the SSEM speed profile calculation, resulting in a deviation from the subset 125.

- The Profile Tracking sub-module, similarly to the GoA2 model, selects appropriate traction and braking commands so that the train tracks the OSP.

- The **train model** module, as in the GoA2 train model, computes the train kinematics along the line by modelling longitudinal dynamics, including basic Davis resistances, gradient effects, and traction and braking characteristics (maximum effort as a function of train speed). The model operates under the idealized assumption of perfect position accuracy, with zero measurement error.
- The **APM (Automatic Processing Module)** is a safety-critical component responsible for reacting to hazardous situations and is therefore intended to operate at a high Safety Integrity Level (SIL). However, the simulation scenarios considered in this work do not include such hazardous situations. Consequently, the APM behavior has been significantly simplified and restricted to managing the ATO GoA4 engagement and disengagement state transitions, as described in Section 4.6.3.4 of deliverable D31.4, based on the work performed in the X2Rail project.

The ATO GoA4 functional model is executed in co-simulation with Unreal Engine, a real-time 3D graphics engine developed by Epic Games and used here to model and simulate the virtual railway environment. Unreal Engine provides a realistic and immersive synthetic world, in which the track layout and lineside signaling are represented and rendered. These virtual assets are either created directly in Unreal Engine or imported from external 3D models, and serve as inputs to the PER module for the perception of lateral signals.

In the X2Rail architecture, the repository (REP) is responsible for sharing data related to signaling (signal types, possible aspects, associated MA characteristics) and mission (predefined data for each scenario). For simplification purposes, the communication between the REP and the other modules is not explicitly modelled in this work. Instead, each functional module is assumed to have direct internal access to the subset of repository data it requires in order to compute its outputs.

ATP is not modeled, but signalling information is incorporated via SSEM profile.

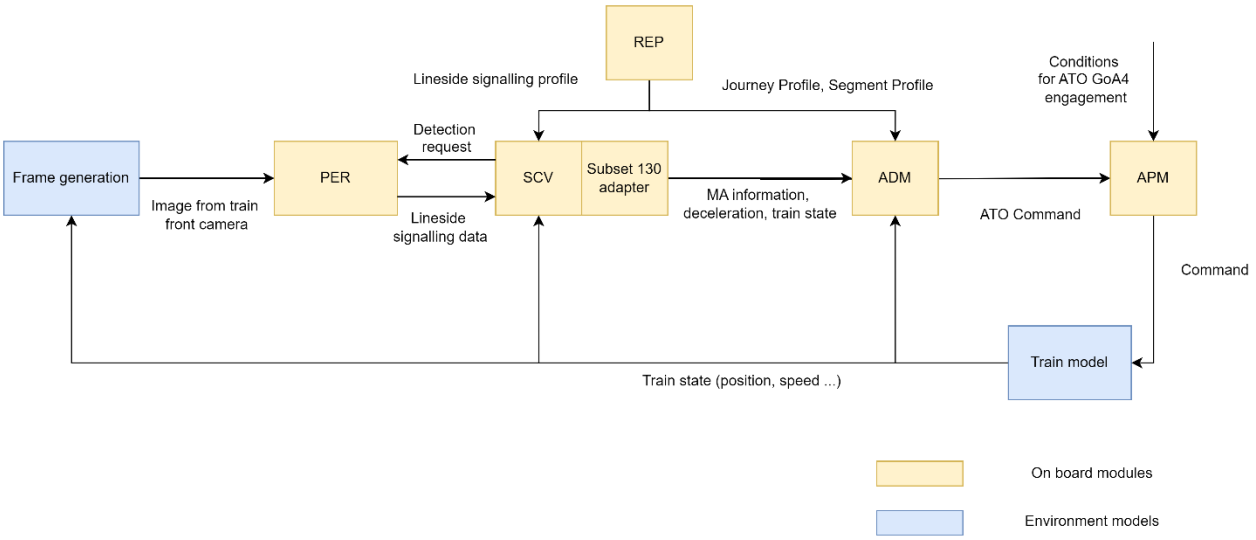


Figure 3: ATO GoA4 architecture for modelling

3 MODELLING TECHNIQUES ACROSS THE SYSTEM DEVELOPMENT CYCLE

This chapter aims to introduce the concept of a model, its implications, the expected benefits, and its practical implementation throughout the development cycle of a system.

3.1 INTRODUCTION

Complex systems such as Control-Command and Signaling (CCS) systems are characterized by numerous interfaces, strong interdependencies between subsystems, and interactions across diverse technical domains including mechanics, electronics, and software systems. These systems present not only technical challenges but also organizational complexities, as the growing number of stakeholders and interactions complicates collaboration, requirements management, and information sharing.

Model-Based Systems Engineering (MBSE) and Model-Based Design (MBD) methodologies directly address these complexities by decomposing intricate problems into modular, manageable components. These approaches enhance efficiency through improved stakeholder collaboration, systematic reuse of validated components, and early-stage verification via rapid prototyping—significantly reducing development risks and accelerating innovation cycles.

When applied consistently across the entire V-cycle of system development, the benefits of this model-based approach become particularly significant. In the early, upstream phases, MBSE supports the construction of a coherent functional and physical architecture of the system, providing a shared, traceable representation around which diverse stakeholders can align. Building on this system-level analysis, MBD then propagates the advantages of modelling into all subsequent phases of the cycle. In the downstream branch of the V-cycle, models enable early feasibility and impact studies (e.g. assessing effects on adjacent systems or quantifying expected performance gains) and allow specifications to be validated by confronting them with executable implementations. Compared with purely textual requirements, model-based specifications can offer a more precise and more accessible medium for communicating intent to industrial partners. This practice is already well established in sectors such as automotive engineering but still relatively uncommon in the railway domain. During design and validation phases, exploiting the capabilities of modelling tools for automated code generation and systematic test-suite construction makes it possible to validate the design while capitalising on the models previously developed.

The inherent flexibility of modelling tools—whether functional or analytical—enables seamless adaptation of models as system design choices evolve. This agility is critical in complex projects like railway systems, where requirements and technical constraints frequently shift. By maintaining live links between models, requirements, and validation artifacts, teams can instantly visualize the cascading effects of a design modification across the entire system. This impact analysis not only reduces the risk of undetected inconsistencies but also ensures that all stakeholders—from engineers to certification bodies—remain aligned with the latest system baseline.

The following subsections detail these key stages of the V-cycle and the modelling tools and techniques associated with each of them.

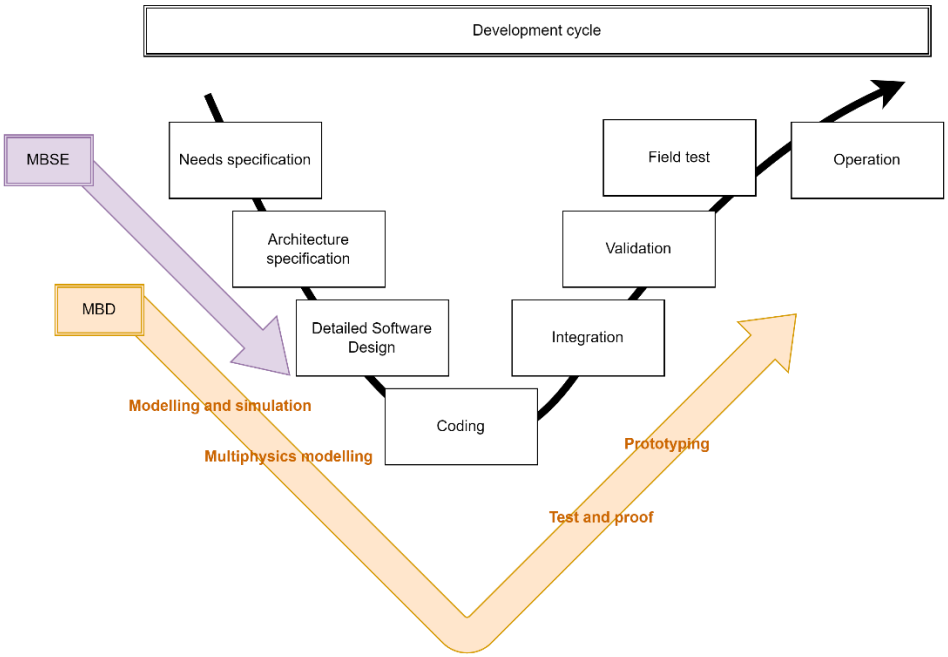


Figure 4: MBSE and MBD along the system development cycle

The application of model-based approaches proves particularly valuable for Automatic Train Operation (ATO) development, where several critical needs converge:

- **Normative validation**: Supporting the verification of emerging standards in a controlled virtual environment
- **Quantifiable benefits assessment**: Objectively evaluating ATO's operational gains, particularly in energy efficiency and system performance
- **Systemic impact analysis**: Examining ATO's functional interactions with other CCS components in a comprehensive virtual ecosystem

This modeling-centric approach becomes essential given the reduced implementation of ATO fully compliant with ERTMS standards, which limits physical testing opportunities. Physical tests are also costly (they mobilize rolling stock, track infrastructure and personnel), are less flexible than simulation (limited scenario coverage, long setup times) and require extensive safety measures and operational coordination. By creating high-fidelity virtual representations, MBSE and MBD enable comprehensive validation of the evolving concepts and requirements of the ATO systems.

3.2 Model-Based System Engineering (MBSE)

Model-Based System Engineering (MBSE) plays a pivotal role in the downstream branch of the V-cycle, where it supports the system-level architectural definition, requirements engineering, and early-stage design activities.

MBSE focuses on the architectural representation of a system, including its structure, behaviour, and data. It considers the system as a whole, considering internal interdependencies rather than treating each subsystem separately. MBSE supports core systems engineering activities and provides outputs for transversal activities, such as RAMS (Reliability, Availability, Maintainability, and Safety), human factor, risk management, configuration management, alternative evaluation and decision making.

The MBSE approach is based on three key pillars:

1. A **standard** for system representation: SysML (Systems Modelling Language) provides a structured way to describe system architecture, behaviour, and interactions.
2. A step-by-step definition **process**: Frameworks like ARCADIA, CESAM, and MagicGrid guide system engineers through progressive modelling stages, ensuring consistency and completeness.
3. A modelling **tool**: Software such as Capella, MagicDraw, and others enable the practical application of MBSE by providing visualization, analysis, and documentation capabilities.

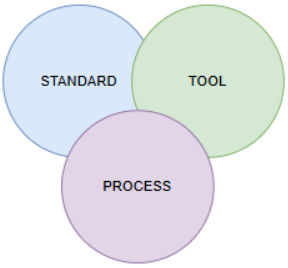


Figure 5: MBSE triad

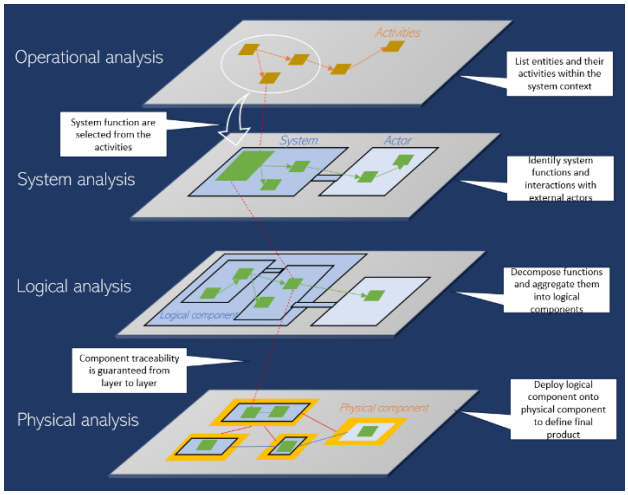


Figure 6: Arcadia analysis layers

The MBSE approach establishes a continuum from requirements to physical implementation, integrating disparate engineering disciplines (e.g., systems, software, mechanical) into a cohesive development framework. By enabling cross-domain impact analysis and maintaining design consistency through a centralized data repository, MBSE eliminates silos and ensures alignment across all development phases. This methodology has proven to significantly reduce costs and timelines while enhancing quality—primarily through automated traceability, early validation, and minimized late-stage modifications. Such benefits are well-documented in both academic literature and industrial applications, particularly in automotive and aerospace sectors.

For a comprehensive overview of the MBSE principles and system analysis methodologies, refer to deliverable D31.4, which lays the foundational analysis preceding the model development and simulation activities detailed in this document.

3.3 MODEL-BASED DESIGN METHODOLOGY ALONG THE DEVELOPMENT CYCLE

3.3.1 Methodology

A model is a simplified representation of a system or phenomenon, designed to replicate its behaviour within a controlled and parameterized environment. By abstracting complexity, models enable engineers and decision-makers to experiment with virtual systems—eliminating risks associated with real-world testing. This capability is particularly valuable in the development of ATO, where simulations allow for rigorous testing of algorithms, infrastructure interactions, and operational scenario.

Model Based Design (MBD) is a systematic approach to engineering that uses dynamic models in support of system specification, design, implementation, and verification. By shifting from document-centric to model-centric development, MBD enables continuous validation throughout the development cycle, significantly improving efficiency, traceability, and system quality. For instance, inconsistencies in system specifications or behavioural anomalies can be identified during the modelling phase, long before they manifest as costly errors in later stages of the V-cycle. Addressing these challenges early not only saves time but also minimizes the financial and operational risks associated with late-stage redesigns.

The proposed MBD workflow is organized into three phases with several feedback loops.

- **Model specification phase**: This step involves translating system specification into model specification. During this phase, engineers select the requirements to be modelled based on the specific objectives of the modelling task. This selection process is critical, as it determines the representativeness of the model: only the requirements that are explicitly modelled will be reflected in the simulation. At this stage, it is already possible to identify inconsistencies or ambiguities in the textual specifications, allowing for early corrections that improve the clarity and feasibility of the design.
- **Model implementation phase**: Once the model specifications are established, the next phase focuses on their computational implementation. Using simulation tools such as MATLAB/Simulink, engineers develop a functional model that mirrors the intended system behaviour. This phase often reveals gaps in the initial specifications: for example, cases where the input data provided is insufficient to generate the expected outputs.

- **Model testing phase**: The final phase involves validating the model through a comprehensive testing process. Test cases are designed based on the system requirements, and their execution helps to identify both implementation errors, such as coding mistakes that cause the model to deviate from its specifications, and specification gaps, where the model's design fails to cover critical operational scenarios. When discrepancies arise, the testing phase provides an opportunity to refine both the model and the underlying system specifications. Additionally, test coverage analysis (see section 3.3.3) can be employed to ensure that the test cases adequately address the full scope of the model's functionality, further enhancing its reliability.

Throughout this workflow, it is critical to maintain explicit documentation of which system specifications have been selected for inclusion in the model. This traceability ensures full awareness of the model's scope and boundaries—clarifying not only what the model represents but also what it intentionally excludes. By systematically tracking these choices, engineers retain precise control over the model's level of representativeness, avoiding misinterpretations of its capabilities and limitations.

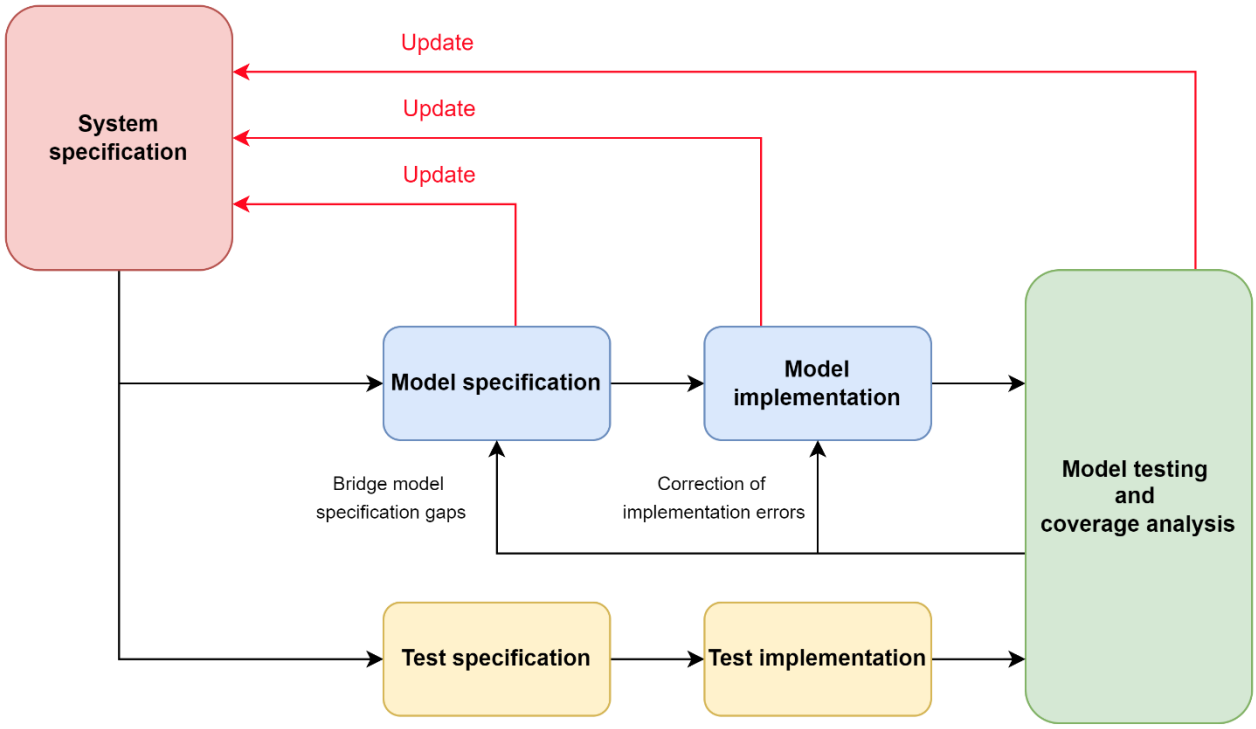


Figure 7: Proposed MBD workflow

Having established the principles and workflow of MBD, we demonstrate in the following sections how this methodology is applied to key stages of the V-cycle.

3.3.2 Upstream study and specification verification

3.3.2.1 Challenges

In the upstream phase of the V-cycle, the primary challenge is to understand and quantify the impact of introducing a new system in its environment, both from a technical and an economic perspective. For ATO at GoA2, this includes, for example, techno-economic assessments of expected energy

savings, as well as analyses of how the new functions interact with existing systems such as signalling, traction, braking, and traffic management.

In parallel, implementing the textual specifications into executable models makes it possible to systematically reveal ambiguities, inconsistencies, or missing requirements, thanks to the feedback loops introduced in the MBD workflow described in Section 3.3.1.

To address these challenges, the question of model representativeness becomes central: the model must be sufficiently accurate to capture the relevant phenomena, while remaining focused on the objectives of the study. The following subsections therefore present two complementary modelling approaches based on MATLAB/Simulink—analytical modelling and multiphysics modelling. These tools are not specific to this phase of the V-cycle; they are reused and refined throughout the design and validation process.

3.3.2.2 MATLAB/Simulink

Analytical modelling consists of representing the system behaviour through mathematical equations that describe physical laws, control algorithms, and operational constraints. This approach allows engineers to simulate continuous or discrete dynamics of the system by numerically solving the underlying differential or algebraic equations over time.

MATLAB/Simulink is a comprehensive development environment widely used in engineering for system modelling, algorithm development, and data analysis. MATLAB, with its matrix-based programming language, excels in numerical computations and algorithmic design, while **Simulink** provides a graphical interface for dynamic system simulation and the implementation of the *Model-Based Design* (MBD) approach. This integrated environment enables users to create complex models by connecting functional blocks from built-in or custom libraries, facilitating both high-level system design and detailed component-level analysis.

MATLAB/Simulink is particularly well-suited for MBD, offering a range of tools that streamline the development lifecycle on multiple aspects:

- **Project organization**: MATLAB provides a project framework that supports collaborative work, with predefined paths and integration with version control systems like Git.
- **Testing and validation**: Built-in tools for verifying models and ensuring compliance with requirements.
- **Multiphysics simulation**: Simscape extension enables the modelling of physical systems across domains (e.g., mechanical, electrical, fluids).
- **Co-simulation**: Compatibility with external software, like Unreal Engine and SIMPACK, to enhance simulation fidelity.
- **Real-time deployment**: Compilation and execution of the models on real-time targets.

These capabilities make MATLAB/Simulink a versatile platform for accelerating the development and validation of complex systems, as demonstrated in the models discussed in this document.

3.3.2.3 Multiphysics Simulation

Railway rolling stock is an inherently multiphysical system, composed of interacting subsystems spanning heterogeneous domains such as mechanical structures, pneumatics, hydraulics, electrotechnics, power electronics, embedded software, and communication systems. The overall

train architecture integrates all subsystems that provide traction, braking, comfort, safety, and control, and relies on tightly coupled components: mechanical assemblies (bogies, chassis, transmission, suspensions), pneumatic equipment (air brakes, doors, auxiliary suspension), hydraulic devices (damping or lifting systems), electrical and electrotechnical subsystems (traction chain, power supply, auxiliaries), power electronic converters (inverters, choppers), onboard computing (control-command, diagnostics, safety systems), as well as HVAC, signalling and communication. This level of complexity requires a rigorous multidisciplinary modelling approach in order to ensure proper integration of the subsystems, as well as overall performance, reliability, maintainability, and compliance with railway safety requirements. Automatic Train Operation (ATO) is deployed on top of this complex physical system and interacts continuously with it.

Multiphysics modelling tools enable a detailed representation of the components that interact directly with the ATO functions, such as the localisation chain, traction and braking systems, and the longitudinal and lateral train dynamics. They make it possible to know how the train would react to the commands generated by the ATO under varying operating conditions and environmental constraints. As a result, the associated modelling challenges are multi-physics and multi-scale (from real-time behaviour to long-term energy and wear assessment), and they require strong interoperability between tools and models. High-fidelity simulation becomes a critical step to validate the ATO concepts and parameterisation before any real-world deployment on operational lines.

To model multiphysics systems, MATLAB and Simulink provide a flexible framework for implementing dynamic system equations. However, for complex physical interactions, dedicated simulation tools offer a more efficient approach by directly representing physical components and their connections, without requiring manual equation implementation. In this deliverable, the discussion focuses on two such tools (Simscape and SIMPACK). It is worth noting that this selection is not exhaustive; other specialized software, such as TrainDY, also exists for train longitudinal dynamics simulation. A comparative analysis of these tools falls outside the scope of this report but could be the subject of a dedicated study.

Within the MATLAB/Simulink environment, Simscape enables multiphysical modelling by assembling components from mechanical, electrical, hydraulic, and other domains, while Simulink orchestrates the simulation workflow. Alternatively, co-simulation with specialized tools like SIMPACK can be employed for high-fidelity mechanical dynamics. In this setup, Simulink hosts control algorithms and subsystem models, while SIMPACK computes detailed multibody dynamics, with real-time data exchange between the two environments.

Simscape

Simscape is an extension of Simulink for modelling and simulating physical systems using networks of physical components rather than purely signal-flow blocks. While Simulink connections carry abstract signals (for example, the train speed), Simscape connections represent physical interactions between components, such as an electrical conductor at a given potential carrying current, or a pipe conveying a fluid with a certain flow rate.

Simscape is organized into several libraries that cover specific physical domains (electrical, mechanical, fluid), and provides interface components that couple these domains, such as an electric motor linking the electrical and mechanical domains.

One of the main benefit of Simscape is its tight integration with Simulink: control variables can directly drive Simscape sources (for example, a controlled voltage source), and sensors can expose physical quantities (such as the rotational speed of a shaft) back to the Simulink environment. This allows engineers to design and test control architectures around a physical model within a single framework, from early-stage concept evaluation to detailed controller design and validation.

SIMPACK

SIMPACK is a multibody system simulation software developed and owned by Dassault Systèmes. It is widely used to model and simulate the dynamic behaviour of mechanical systems composed of multiple interconnected bodies—both rigid and flexible—linked by kinematic joints and force elements. Each body in the system is characterized by its centre of gravity, mass, and inertia matrix, enabling precise representation of its physical properties.

The kinematic connections between solids are modelled using connections defined in SIMPACK, which specify the number of degrees of freedom (DoF) allowed between two solids. Force elements (springs, dampers, bump stops, bushings, etc.) are modelled using elements defined in the SIMPACK toolbox.

Multibody simulation softwares like SIMPACK are grounded in multibody system dynamics theory, making them a cornerstone tool for railway dynamics experts. The software employs advanced numerical methods to solve the complex sets of equations governing the system's motion, including ordinary differential equations (ODEs) and algebraic equations that describe the dynamic behaviour of the complete vehicle-track system.

In this project, SIMPACK is used in co-simulation with MATLAB/Simulink. SIMPACK models can be exported as an S-function and embedded into MATLAB/Simulink. This allows direct interaction with the model through the S-function input/output ports, without requiring a local SIMPACK installation, thus facilitating co-simulation and system-level analysis.

3.3.3 Tests and proof

Software development standards mandate a rigorous approach to testing, requiring the definition, execution, and formal documentation of multiple testing levels. These include unit tests, which validate individual components in isolation; integration tests, which ensure that combined modules interact correctly; and validation tests, which confirm that the system meets its specified requirements and behaves as expected in real-world scenarios.

MBD plays a critical role in streamlining this testing process. Through systematic analysis of test results, teams can rigorously validate the quality of developed models by subjecting them to typical operational scenarios and verifying compliance with specifications. This approach ensures that models not only meet functional requirements but also behave reliably under real-world conditions, while enabling systematic analysis of test outcomes. The structured nature of MBD allows for early detection of inconsistencies or defects, reducing the risk of costly errors later in the development cycle. The tests conducted during the model's specification phase in Model-in-the-Loop (MIL) can also be reused to validate the system's behavior at higher physical integration levels (PIL, HIL) during or after the design process (see section 3.3.4).

The reliability and trustworthiness of a software system are inherently tied to the quality of its testing. A well-tested system inspires confidence among stakeholders, from developers to end-users, by

demonstrating that it performs as intended under various conditions. To objectively assess this quality, MBD provides diagnostic tools that quantify key metrics, such as test coverage rates. These metrics offer tangible insights into the effectiveness of the testing process, highlighting areas that may require additional validation or refinement.

	MBD benefit	MATLAB/Simulink tools
Static program analysis	Identify programming formal errors: overflow, ...	IDE Matlab
Source signals	Wide range of generic signals, interface for signal edition, data import and playback	« Source » library, Signal Editor
Test harness	Encapsulation of the model within a test environment: independence and modularity.	Referenced model
Variant blocks	Evaluate multiple alternatives for a module	Variant model (subsystem, source …)
Test coverage	Analysing test coverage: which modules and decision path are activated par the tests?	Coverage Analyzer
Proof	Analysing assertions	Simulink Design Verifier

Table 1: MBD for testing

3.3.4 Prototyping : from Model-in-the-loop to Harware-in-the-loop

Prototyping leverages the capabilities of modelling tools to automatically generate compiled code and to interfacewith real-time physical systems. By deploying code generated from the model on target hardware or connecting a PC running the environment model to real-time test benches, the same models developed in early phases remain directly exploitable in advanced stages of the V-cycle, from detailed design to integration and validation.

The development process leverages a progressive prototyping approach with increasing levels of hardware integration and system realism. This workflow follows four main stages aligned with the V-cycle development phases:

- **Model-In-the-Loop (MIL)** represents the initial stage where the model executes entirely on a development workstation. This approach enables rapid bug detection and correction through validation tests, without real-time constraints, allowing time acceleration for efficient testing. MIL is primarily used during the early descending phases of the V-cycle, focusing on algorithm development.
- **Software-In-the-Loop (SIL)** maintains the same PC-based execution environment as MIL, but introduces compiled code testing, providing an early verification of code generation and compilation processes.

- **Processor-In-the-Loop (PIL)** marks a significant step toward system representativeness. The compiled code runs on actual microcontroller close to the final target platform. While the surrounding environment can remain simulated on a PC, the connection with the PIL module occurs through dedicated electrical interfaces, operating in real-time conditions. This configuration closely mimics the behaviour of embedded software in the real system, significantly increasing representativeness. Operating in real-time with actual electrical interfaces, PIL testing addresses critical aspects such as network communications, synchronization, and latency. PIL is typically employed during later validation and integration phases to assess system accuracy and performance.
- **Hardware-In-the-Loop (HIL)** represents the highest level of realism, where the code executes on the final target hardware with the complete system also implemented on hardware platforms. This stage is essential for final validation and integration phases, ensuring system performance under realistic conditions.

For this project, the MATLAB/Simulink environment, combined with Speedgoat hardware, facilitates seamless transitions between these prototyping stages, enabling efficient progression from pure simulation to hardware-based validation.

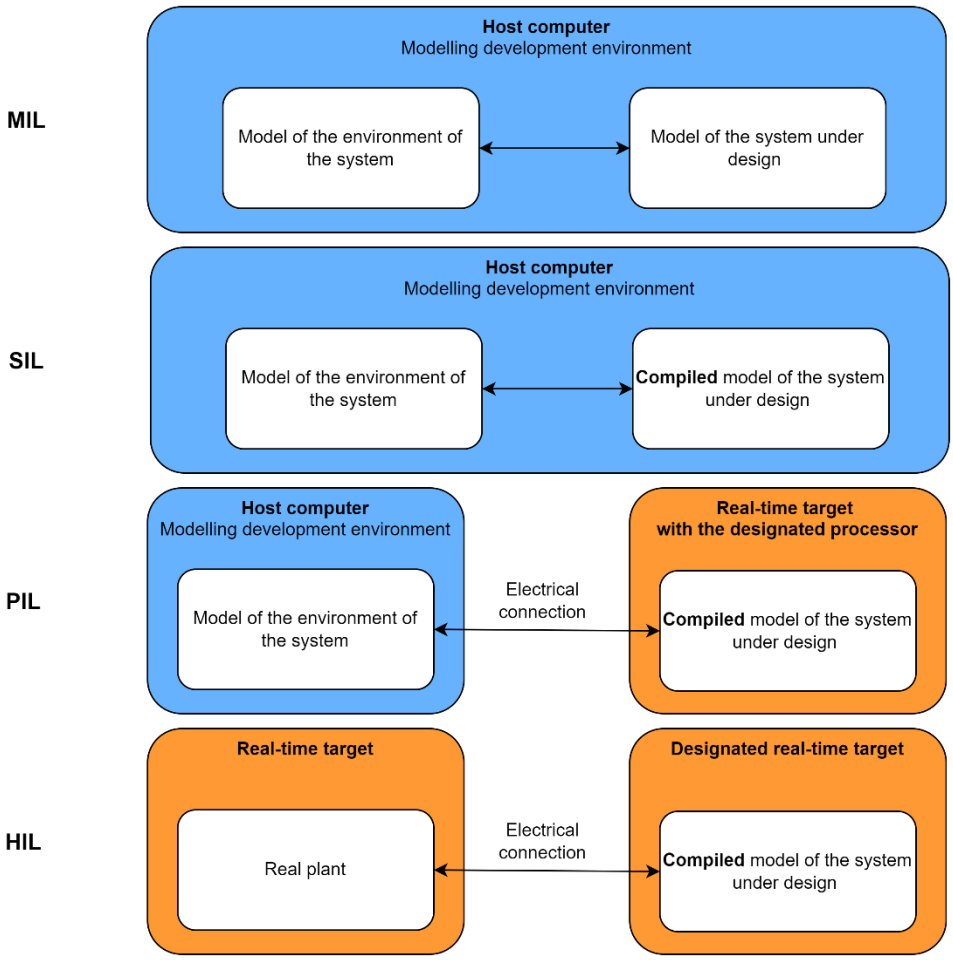


Figure 8: MBD process

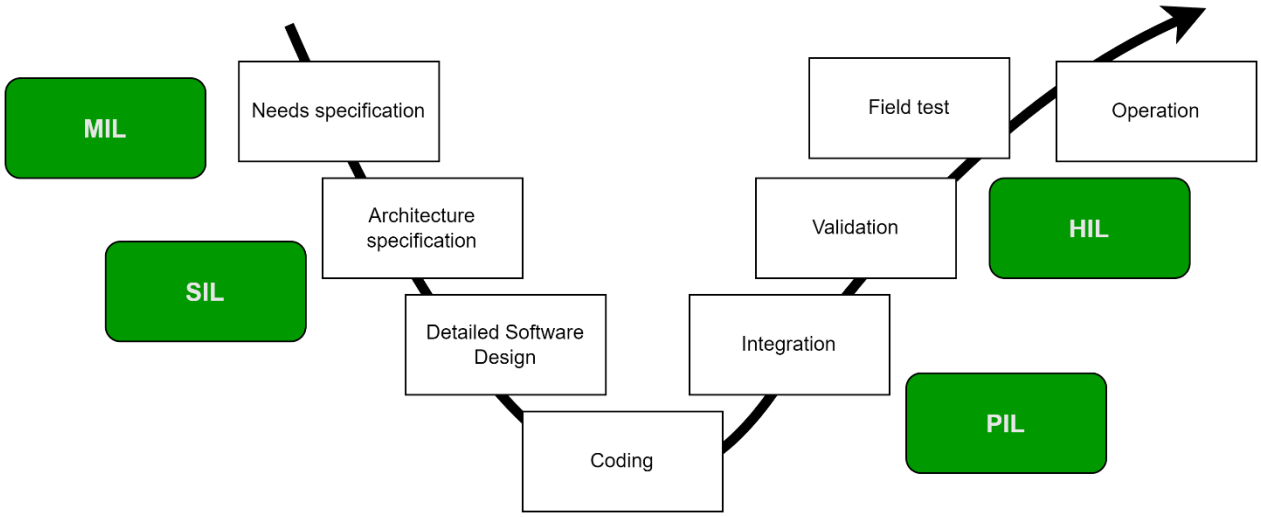


Figure 9: MBD process in development cycle

3.4 BENEFITS

The implementation of this methodology has demonstrated substantial benefits, particularly in terms of development flexibility and operational efficiency. Throughout this project, ,we experienced firsthand how MBD's inherent advantages directly contributed to achieving our technical objectives while optimizing resource allocation.

Modularity

The modular architecture of MBD proved particularly valuable in our project context:

- **Component reusability** was a major asset, allowing previously developed and validated blocks to be integrated into new systems. For instance, the ATO-OB from the GoA2 model has been reused in the GoA4 model development for ADM, saving significant development time.
- **Structured work organization** followed our management plan, with clear file organization enabling parallel development of unit models, individual validation, and seamless integration.
- **Interface isolation capabilities** enabled targeted functional testing by temporarily replacing complex modules with simplified mock-ups. As an example, during the GoA4 model development, the perception module (PER) was frequently replaced by a simplified module providing expected inputs for the SCV. This approach allowed the team to test the correctness of the remaining functional chain behavior (SCV and ADM) without interference from perception module complexities.

Flexibility and Adaptability

The model-based approach provides flexibility, enabling rapid scenario modifications without extensive system reconfiguration.

In the GoA4 model, test scenarios can be easily selected with automatically adapted data (Journey Profile, Speed Profile). Additionally, real-time modifications of environmental conditions—such as

weather parameters and traffic light states—are possible through a dedicated scenario control panel, significantly accelerating validation processes across diverse operational conditions.

Controlled Fidelity

One of MBD's most valuable aspects was its ability to precisely control the level of detail and representativeness of each model component, ensuring alignment with specific objectives while optimizing computational resources and development time.

This principle was applied in the GoA2 model development, where the SSEM was deliberately not modeled in the ATO-OB. By focusing on nominal operational scenarios—where the train does not need to brake due to signaling or incidents—the model remained sufficiently representative to evaluate energy savings potential while maintaining manageable complexity.

Structured Development Process

The MBD methodology brought clear structure to our development workflow through well-defined phases:

GoA2 Development Phases:

- STI data recovery and analysis
- Literature review for speed profile optimization (TTSM construction)
- Design and specification of the optimization algorithm
- Development of a train model with real operational parameters
- Coding and implementation of the tracking algorithm: Model Predictive Control (MPC)
- Integration within a simulator including train model and profile tracking functions

GoA4 Development Phases:

- Architecture design based on MBSE principles and model objectives definition
- Team organization leveraging modular aspects for parallel development
- Precise interface definition between functional blocks
- Individual block development with unit validation
- Final block integration and scenario validation

Cost effectiveness

Developing and validating systems through model-based approaches delivers significant cost advantages compared with extensive physical field testing. Building and running virtual models requires far fewer resources than organizing tests that mobilize rolling stock, track access, specialized personnel, and the logistical coordination to integrate new systems into operational trains—a process that also entails substantial safety measures and associated regulatory overhead.

4 SIMULATION RESULTS

This section presents the outcomes of our modelling efforts, demonstrating how simulation contributes to the broader development lifecycle of automated train operation (ATO) systems. By analysing these findings, we derive key conclusions about the strategic value of model-based approaches, while also identifying future perspective. The results underscore the role of modelling as both a validation tool and a catalyst for iterative improvement, bridging early design phases with final system deployment.

4.1 GoA2 SIMULATION RESULS

4.1.1 Model architecture

The ATO GoA2 model is designed to simulate a complete mission under GoA2 automation, with the primary objective of estimating energy consumption and comparing it against real-world driving profiles for the same mission. This comparison enables a quantitative assessment of the energy savings achieved by GoA2 ATO on the studied mission. The modelling objectives and assumptions were listed in Section 2.2.2.

Unlike the Subset-125 standard, this model does not explicitly simulate the SSEM profile, as the mission is assumed to operate without signaling-related disruptions.

The Optimum Speed Profile (OSP) is generated based on two distinct driving profiles:

- TTSM– Optimized for punctuality and energy efficiency.
- ATSM– Focused on stopping precision at station

Rather than merging these profiles, the model switches dynamically between two associated modes, triggering a recalculation of the corresponding profile whenever a mode change occurs. Additionally, a STOP mode is introduced to represent periods when ATO GoA2 is disengaged at station.

The implementation of these modes is realized through a state machine developed in Simulink/Stateflow, where transitions between modes are defined in Table 2.

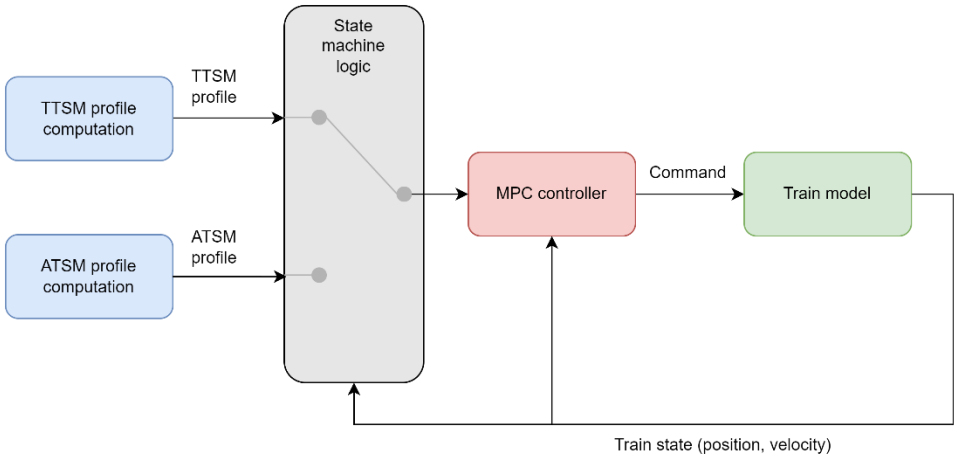


Figure 10: ATO GoA2 model architecture

Origin mode	Target mode	Condition for transition
STOP	TTSM	The scheduled dwell time at the stopping point has elapsed **and** the planned departure time has passed.
TTSM	ATSM	The train is within 500m of the next stopping point **and** the deceleration required to stop at the next stopping point from the current state exceeds 80% of the maximum deceleration capability.
ATSM	STOP	The speed is near zero (below 0.05 m/s) and the train is within 10 m of the next stopping point.

Table 2: Transition between modes

4.1.2 Energy efficient speed profile

4.1.2.1 Methodology

The objective is to minimize energy consumption while respecting timetable and speed limit constraints. After a literature review ([1], [4], [5], [6], [7], [8], [9], [10]), detailed in D31.4 (section 4.7.3.2.1 and Appendix), the selected method is a numerical approach based on graph construction and search [1], which accommodates passing point constraints, i.e. locations where the train must pass within a specified time window without stopping. Notably, a 30-second tolerance is applied before the scheduled arrival time, despite the journey profile does not explicitly define such a window. This tolerance ensures solution feasibility despite the discretization of the search space.

The optimization problem is formulated as follows. The cost function to be minimized is the total energy consumed over the journey, accounting for energy recovery while braking. The constraints are of various types: dynamic behaviour of the train, timetable constraints (at stations and at intermediate points) and speed limits.

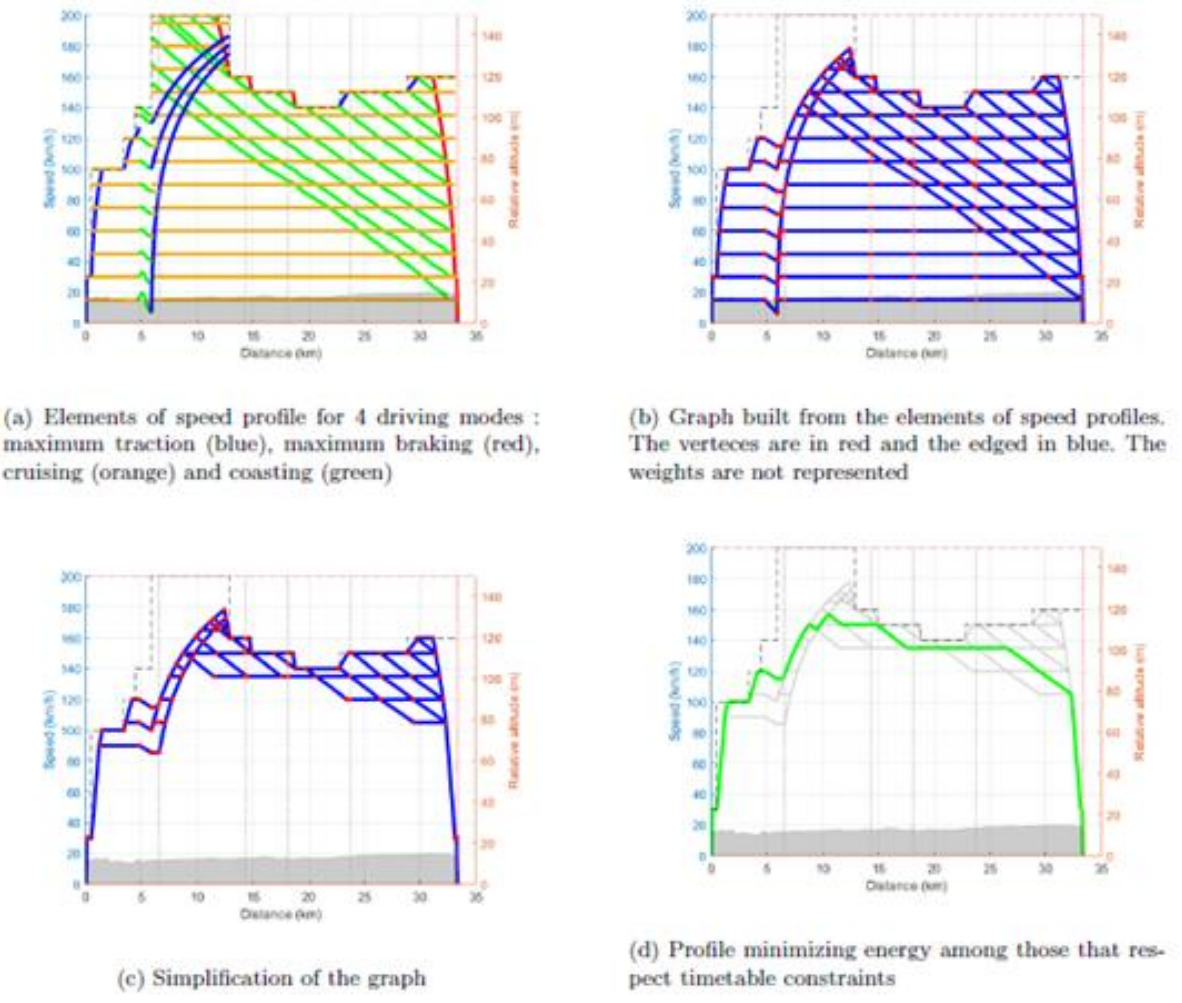
Applying Pontryagin's Maximum Principle [2] to a simplified problem (without intermediate time constraints, speed limits or steep inclines/declines), it can be demonstrated that the optimal solution consists of the succession of four driving modes: maximum traction, maximum braking, cruising (constant speed), and coasting (no traction or braking) [4]. This result is then generalized to the more complex problem described above. Even if this generalization appears valid, it could be theoretically confirmed through further research. Speed profile elements corresponding to these modes are strategically traced between departure and arrival stations, respecting speed limitations and accounting for traction cut-off sections (see Figure 11.a). Discretization steps are involved in building these curves, including the speed intervals between cruising curves and between coasting curves. The choice of these discretization steps must balance computation time against optimization performance. This trade-off is worth exploring as early as the modeling phase to anticipate challenges that may arise during real product development. This topic is addressed in section 4.1.2.3.

A weighted directed graph is constructed, with vertices at curve intersections and edges representing transitions between driving modes. Edge weights reflect time and energy consumption

(see Figure 11.b). The graph is then simplified by removing edges that violate timetable constraints (see Figure 11.c).

The optimization problem is formulated as a Shortest Path Problem with Resource Constraints (SPPRC) [3], solved using dynamic programming to eliminate suboptimal or infeasible paths early, considering intermediate timing points. The algorithm outputs not only the optimal speed profile (see Figure 11.d) but also a set of Pareto-optimal profiles—trajectories for which no alternative consumes both less time and less energy (see Figure 12). Careful selection of algorithm parameters is essential to balance computational cost and solution optimality.

For a comprehensive description of the method used to build the energy-efficient profile, refer to D31.4 (Section 4.7.3.2.1).



(a) Elements of speed profile for 4 driving modes : maximum traction (blue), maximum braking (red), cruising (orange) and coasting (green)

(b) Graph built from the elements of speed profiles. The verteces are in red and the edged in blue. The weights are not represented

(c) Simplification of the graph

(d) Profile minimizing energy among those that respect timetable constraints

Figure 11: Example of the phases of the proposed speed profile optimization heuristic. The grey zone is the elevation profile along the track

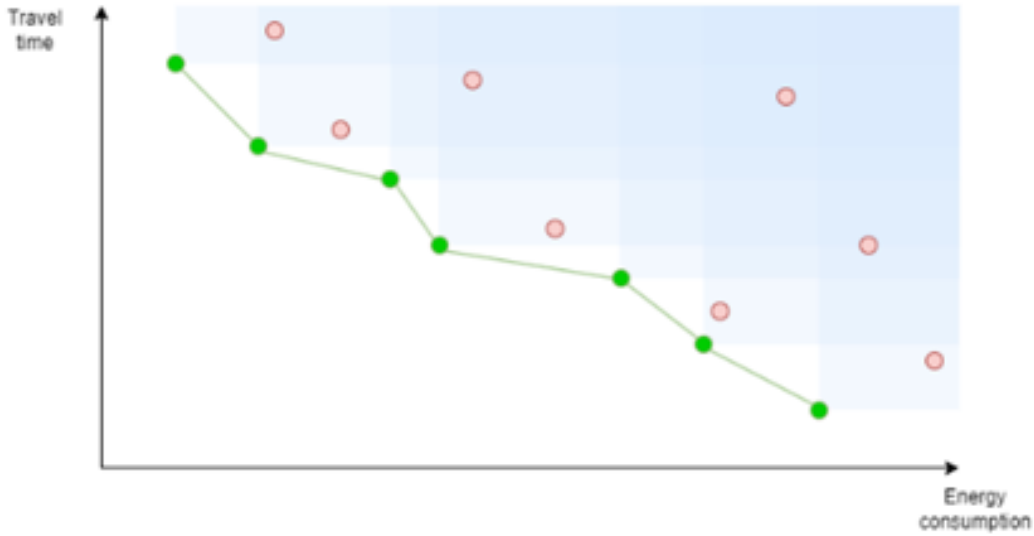


Figure 12: Pareto front. The green points represent Pareto-optimal solutions, i.e.solutions for which no other solution exists that consumes both less energy and less time

4.1.2.2 Scenario

Our modeling effort focused on a specific regional mission connecting three stations. The mission and infrastructure data used in this study were sourced internally from SNCF, encompassing the following key inputs:

- Mission Data
  - Timing Point locations (geographical positions along the route).
  - Scheduled arrival and departure times at stopping points.
- Track Data
  - Track elevation profile (longitudinal gradient).
  - Speed limits (section-specific constraints).
  - Current availability (e.g., location of no power zones).
- Rolling Stock Data
  - Train braking and acceleration characteristics.
  - Train mass and rotational mass coefficient (inertial effects).
  - Running resistance coefficients (from the Davis equation for aerodynamic and mechanical drag).

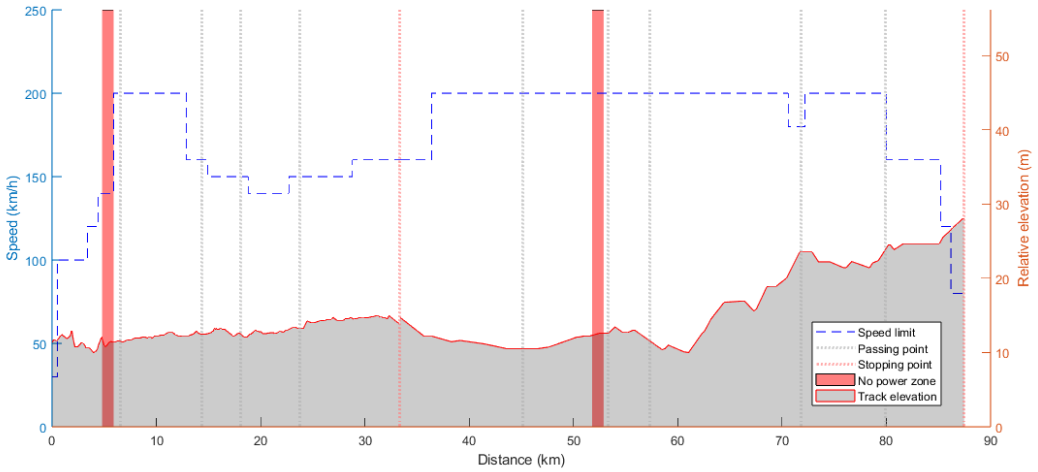


Figure 13: Mission data visualisation

4.1.2.3 Results

Compliance with constraints

The speed curves are directly derived from the train model and systematically generated to never exceed the speed limits defined in the service profile (SP). This ensures inherent compliance with both operational speed restrictions and the train's physical capabilities. The profile's design automatically excludes traction in no-power sections, as the curve generation process integrates power availability constraints. This compliance with track and rolling stock constraints is illustrated on Figure 14.

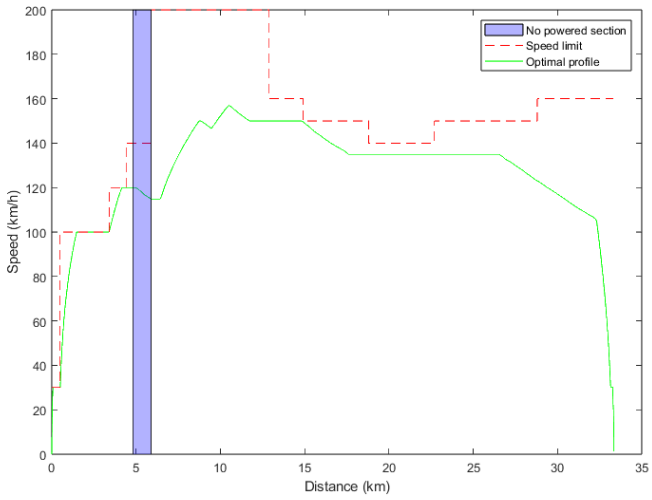


Figure 14: Profile compliance with track constraints

Since any paths that violate timetable constraints are removed during the graph traversal, the algorithm guarantees that the optimal profile satisfies the time windows at passing points as well as

the scheduled times at stopping points. This compliance with timing constraints is shown on the graph in Figure 15.

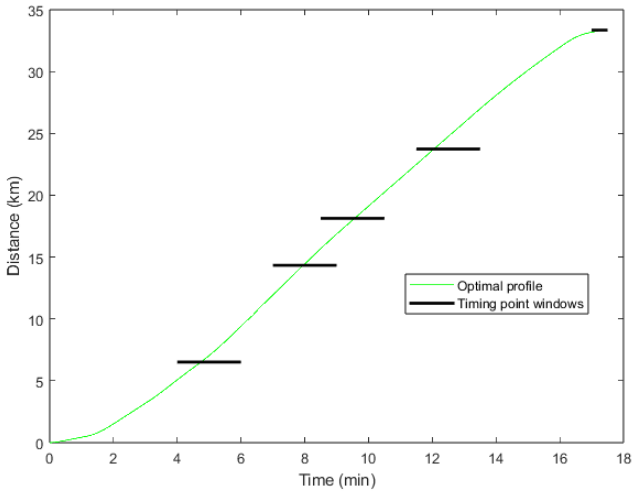


Figure 15: Profile compliance with timetable constraints

Optimality

The algorithm generates a set of Pareto-optimal speed profiles, where no profile in the solution space is both faster and less energy-intensive than another. Among these profiles—illustrated on the Pareto front in Figure 16—we select the minimum-energy profile (highlighted in green). This profile maximizes the use of the allowed travel time, arriving as late as possible while still respecting the scheduled arrival window. For instance, in this case, the optimal profile arrives at the station with less than a 1-second margin, justifying the inclusion of an arrival time window—a constraint not originally specified in the subsets (as discussed in section 4.1.2.1).

The figure also highlights the fastest Pareto-optimal profile in red and an intermediate Pareto-optimal profile in pink.

Discontinuities in the Pareto front arise from the discretization of the solution space, particularly due to the timing of the transition to coasting mode near the end of the profile (around 30 km).

The Pareto front reveals a critical trade-off: small time savings can incur significant energy costs. For example, a 30-second reduction in travel time results in an additional 17 kWh of energy consumption—approximately 8% of the total energy used. This observation raises questions about alternative scheduling strategies, where adherence to timetables might not be a strict constraint, but deviations from the schedule could instead be penalized.

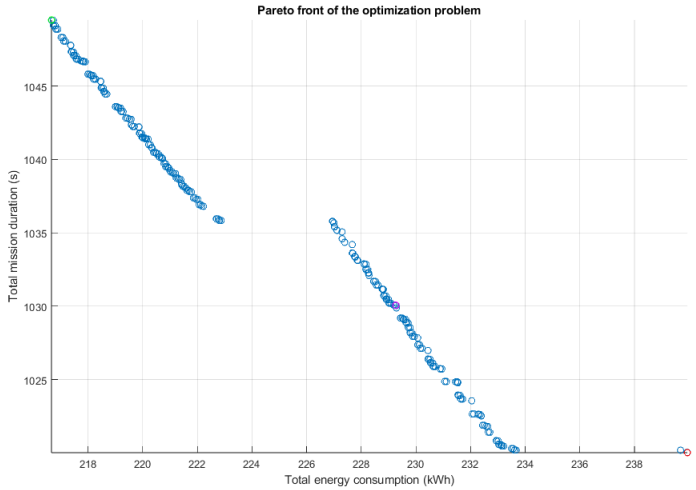


Figure 16: Pareto front for the optimization problem

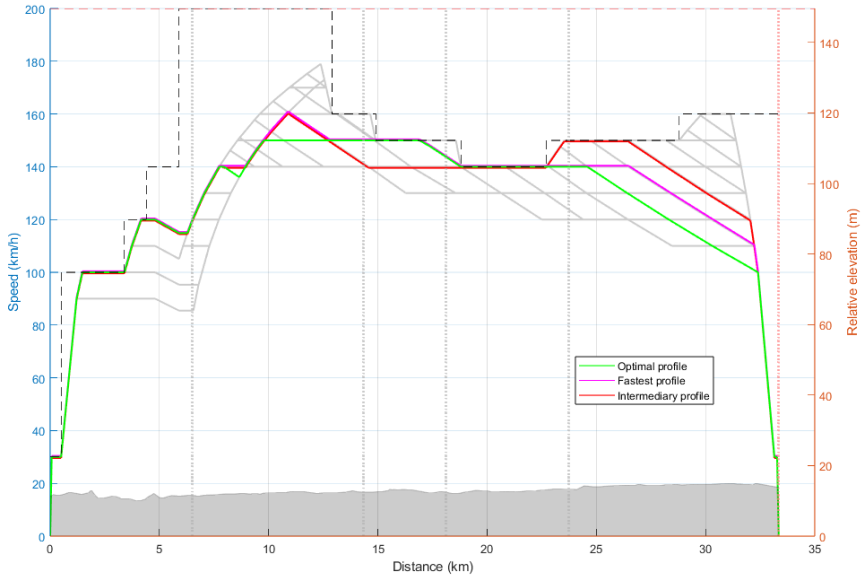


Figure 17: Profiles from the Pareto front

The optimality of the obtained profile is demonstrated by comparison with speed profiles recorded during the actual mission. This comparison is carried out in Section 4.1.4.

Performance (computation time)

A computation time analysis was carried out as a function of two key parameters of the speed profile optimization algorithm which determine the size of the constructed graph:

- The speed increment between consecutive cruising curves.

- The speed increment between consecutive coasting curves (defined along the braking curves).

Reducing these two parameters increases the granularity of the discretization, which leads to a more complex graph and thus allows a larger number of potential solutions to be explored. The outcome of this analysis is shown in Table 3 and Table 4.

The results indicate that decreasing these parameters significantly increases computation time, as it enlarges the graph and therefore the time required to solve the underlying SPPRC. Moreover, the trade-off between computation time and graph complexity is not achieved with a single, universal parameter set: the optimal compromise depends on the specific segment of the mission under consideration.

The trade-off between computation time and optimization quality depends on the application :

- For offline calculations, time constraints are minimal.
- For onboard calculations, computational resources are limited and the application could require faster processing.

COMPUTATION TIME (s)

		Sampling step between cruising curves (km/h)						
		25	20	15	12	10	7	5
Sampling step between coasting curves (km/h)	25	0,13	0,07	0,07	0,19	0,27	0,98	2
	20	0,04	0,09	0,13	0,36	0,37	2,30	5,01
	15	0,08	0,16	0,21	0,70	0,90	2,18	8,70
	12	0,12	0,28	0,25	0,90	1,28	6,19	18,83
	10	0,37	0,78	0,90	2,55	4,05	31,59	49,92
	7	0,75	2,24	2,66	5,30	13,81	34,23	209,28
	5	2,43	7,63	8,87	14,77	28,08	160,90	221,39

ENERGY FOR OPTIMAL PROFILE (kWh)

		Sampling step between cruising curves (km/h)						
		25	20	15	12	10	7	5
Sampling step between coasting curves (km/h)	25	199,52	190,15	190,66	193,35	189,59	189,38	189
	20	191,03	190,14	191,19	190,27	189,61	189,40	189,29
	15	193,56	192,83	192,35	192,63	192,54	188,24	190,31
	12	192,00	188,30	188,81	191,65	188,30	187,55	187,70
	10	187,01	187,43	187,74	188,01	187,73	186,96	187,08
	7	186,83	188,23	187,52	188,95	188,04	186,98	187,02
	5	187,00	187,41	187,33	187,89	187,73	186,75	186,95

Table 3: Results of the performance analysis for the first segment of the mission (station A to station B)

COMPUTATION TIME (s)

Sampling step between coasting curves (km/h)	Sampling step between cruising curves (km/h)						
	25	20	15	12	10	7	5
25	0,12	0,33	0,74	1,49	5,77	26,48	108
20	0,24	1,04	3,23	8,51	15,73	113,04	339,72
15	0,52	2,95	7,34	10,87	45,24	215,63	1003,35
12	1,87	8,53	37,13	68,21	78,80	494,28	3325,60
10	3,19	19,09	60,38	124,02	232,77	1325,18	5731,52
7	12,71	79,55	235,72	965,24	715,83	2763,75	15024,96
5	49,45	383,69	1178,08	2033,08	4785,01	18798,07	/

ENERGY FOR OPTIMAL PROFILE (kWh)

Sampling step between coasting curves (km/h)	Sampling step between cruising curves (km/h)						
	25	20	15	12	10	7	5
25	355,77	356,92	355,22	354,41	353,22	353,42	353
20	357,74	357,96	356,54	356,60	355,45	355,25	354,86
15	355,86	353,38	353,48	353,55	352,98	352,55	352,60
12	357,74	355,02	356,39	354,88	354,39	354,19	354,02
10	357,65	355,29	355,34	355,42	354,75	354,59	354,45
7	356,50	354,30	354,24	353,56	353,48	353,17	353,13
5	354,99	352,91	353,47	352,67	352,11	352,07	/

Table 4: Results of the performance analysis for the second segment of the mission (station B to station C)

4.1.3 Profile tracking: Model Predictive Control

The Optimum Speed Profile (OSP) calculation module was incorporated into a Simulink-based simulator; the system architecture is presented in Figure 10. The simulator uses a closed-loop controller whose purpose is to follow the OSP as closely as possible while respecting physical and operational constraints.

A Model Predictive Controller (MPC) was selected because it can explicitly exploit the available reference information until the next stop and anticipate forthcoming changes in the reference trajectory. At each simulation timestep the MPC formulates and solves a finite-horizon optimization problem: it computes a sequence of control inputs that minimizes a cost function while satisfying a set of constraints. Only the first control input of the computed sequence is applied to the train; at the next timestep the optimization is repeated with updated state information and a shifted prediction horizon (receding-horizon strategy).

While alternative approaches, such as a PID controller, could offer a lighter, computationally simpler solution for onboard deployment, this project specifically aimed to test the implementation of innovative controllers, hence the choice of MPC despite its higher complexity. A comparative analysis of different types of controllers falls outside the scope of this report but could be the subject of a dedicated study.

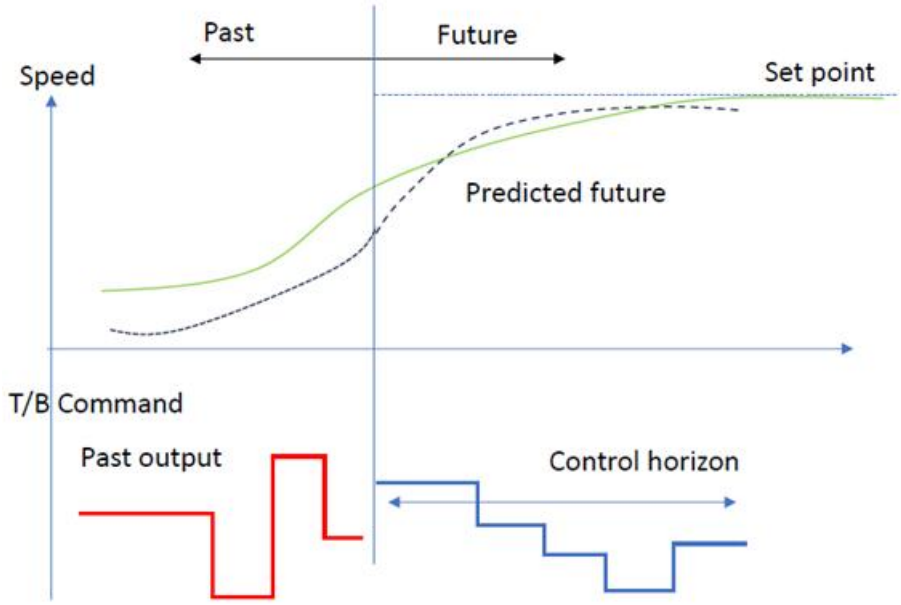


Figure 18: MPC principles (optimal profile in green line and real or predicted profile in black dashed line)

The cost function balances two objectives:

- Accurate tracking of the reference trajectory in both position and velocity, ensuring the vehicle follows the energy-efficient OSP produced by the upstream module.
- Smoothness of the control commands, penalizing large or abrupt variations to preserve passenger comfort and avoid excessive wear on traction and braking equipment.

The optimization is constrained by:

- A linearized model of train dynamics, used to predict future states over the horizon from the current state and the sequence of candidate controls.
- Physical limits on the control inputs, reflecting the maximum available traction and braking forces.
- Speed limits, which ensure the commanded speeds never exceed infrastructure-imposed restrictions.

Results are presented in Figure 19, Figure 20, Figure 21 and Figure 22. Figure 19 shows the overall tracking performance: the controller closely follows the OSP, with a minor overspeed observed around 220 seconds caused by the delay accumulated earlier in the mission that could not be fully recovered so far. Figure 20 shows the variation of the tracking error on position during the mission: the error never exceeds 25m. Figure 21 provides a zoom on a segment and demonstrates the MPC's ability to smooth driving behaviour: the penalization of command variation reduces abrupt accelerations and decelerations, producing a smoother ride, as shown on Figure 22.

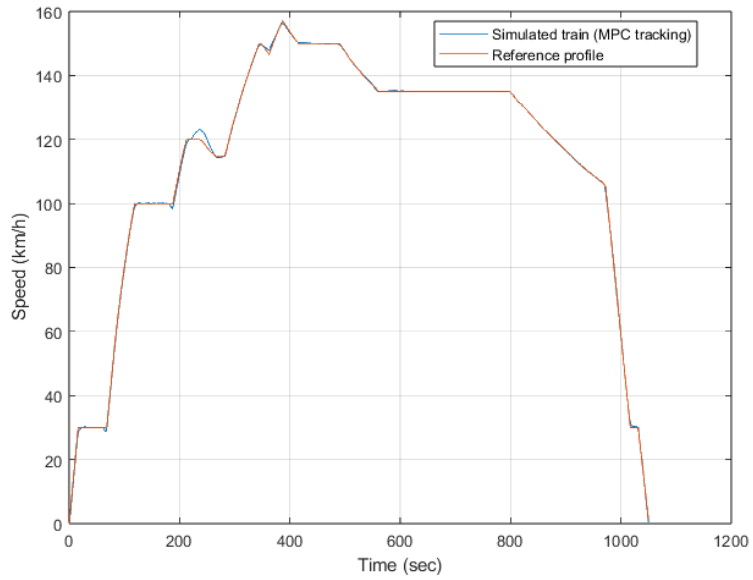


Figure 19: Profile tracking on the entire mission

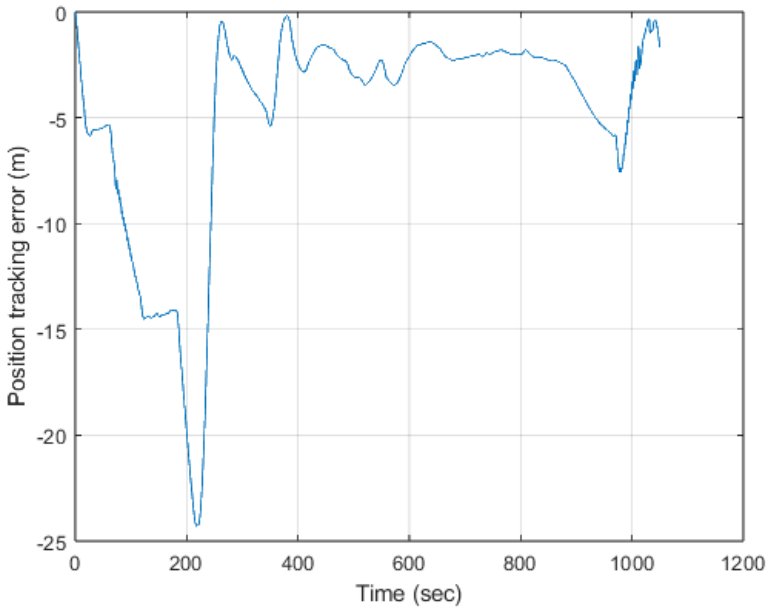


Figure 20: Profile tracking error on the entire mission

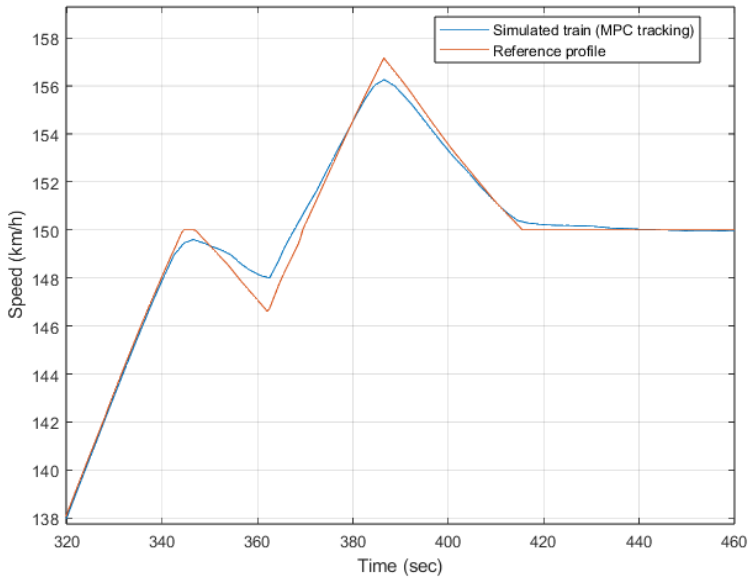


Figure 21: Local profile tracking

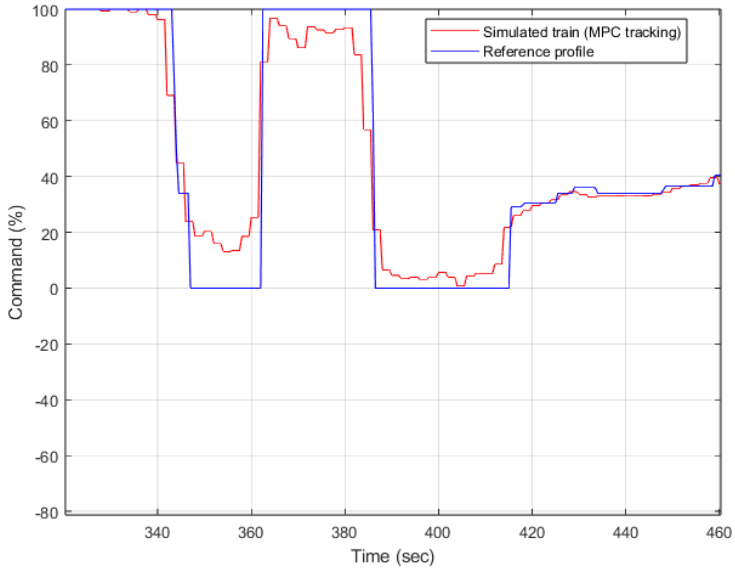


Figure 22: Local simulated command

It is important to note that the MPC parameterization (including the prediction horizon length, sampling time, and weighting matrices) differs depending on whether the system is operating in TTSM or ATSM mode. This distinction arises because the control objectives vary significantly between the two modes:

- In TTSM, the MPC is tuned to gradually recover delays to ensure smooth and energy-efficient operation while maintaining punctuality.
- In ATSM, the focus shifts to precise stopping control, where the MPC prioritizes accurate deceleration to ensure the train comes to a halt exactly at the designated stopping point.

4.1.4 ATO GoA2 energy savings

To assess the energy efficiency gains of the modeled ATO compared to manual driving, we analyze real-world speed recordings (ATESS recordings) from the same mission. However, these recordings include operational anomalies—such as delays, signaling constraints, or other disruptions—that alter driving behavior. To ensure a fair comparison, we filter out outliers, retaining only recordings that strictly adhere to the scheduled timetable. This allows us to isolate energy savings derived solely from eliminating human-induced driving variability.

Figure 23 compares the simulated optimal speed profile against the filtered real-world recordings for the modeled mission. The energy consumption for the recorded profiles was calculated using the train model developed for this study. This computed energy consumption profile was then cross-validated against onboard energy recordings, confirming a sufficient accuracy and reliability of the train model.

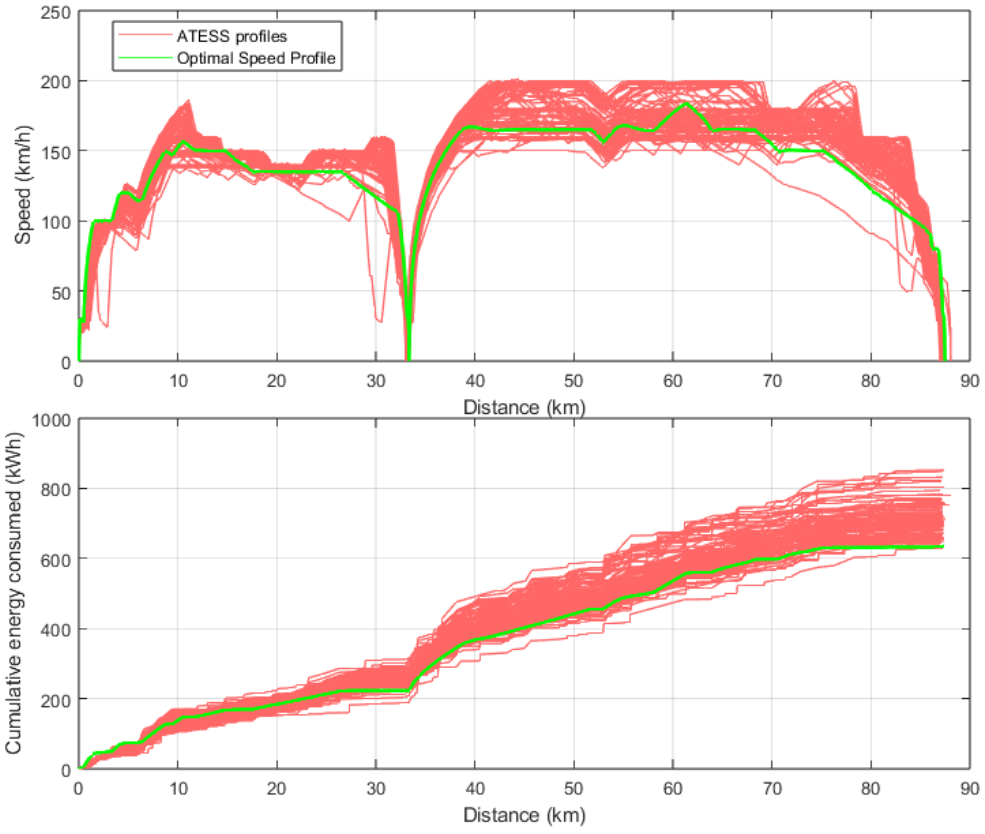


Figure 23: Comparison between optimal speed profile and ATESS recordings

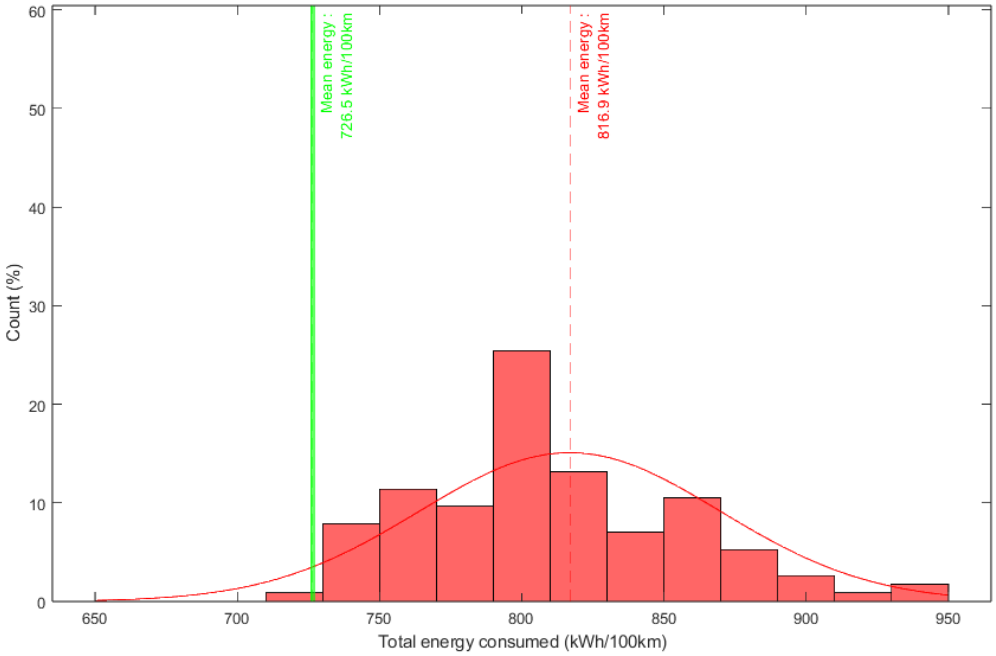


Figure 24: Dispersion on energy consumption

The results demonstrate that the optimized ATO profile achieves energy consumption equivalent to the minimum observed in manual operations, thereby validating the effectiveness of our optimization approach. Overall, in our simulations, the GoA2 ATO allows an average energy saving of about 11% compared with manual driving. This result, however, must be interpreted in the light of the assumptions and the evaluation procedure used. Several factors limit its direct generalization:

- Imperfect filtering of measured speed profiles, which may not fully remove variations linked to signalling or other operational constraints.
- Modelling errors in the train dynamics (e.g. wind, load, adhesion, traction and braking characteristics).
- Measurement errors in the recorded profiles (noise and sampling effects).
- Evaluation performed on a single, specific mission.

Despite these limitations, the model has enabled a quantitative assessment of the energy savings that a GoA2 ATO can realistically achieve on the studied mission.

4.1.5 ATO GoA2 Simulator

The GoA2 ATO model has been integrated into a simulator, implementing new functionalities including a 3D environmental visualization (enabled through co-simulation with Unreal Engine) and the ability to engage or disengage the GoA2 ATO system dynamically.

The simulator's architecture is detailed in Figure 25.

Users can operate the "virtual train" using a traction/braking manipulator and toggle the ATO system via a dedicated button on the controller. Additionally, the simulator incorporates a Driver Advisory System (DAS), which provides real-time driving recommendations (e.g., traction, braking, cruising, or coasting) to help drivers maintain schedules and reduce energy consumption. The DAS can be activated via a separate button on the manipulator.

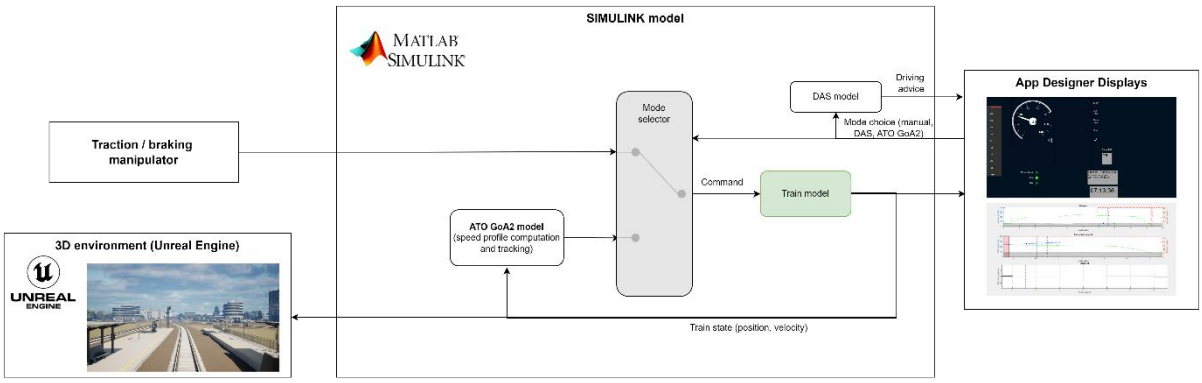


Figure 25: Architecture of ATO GoA2 simulator

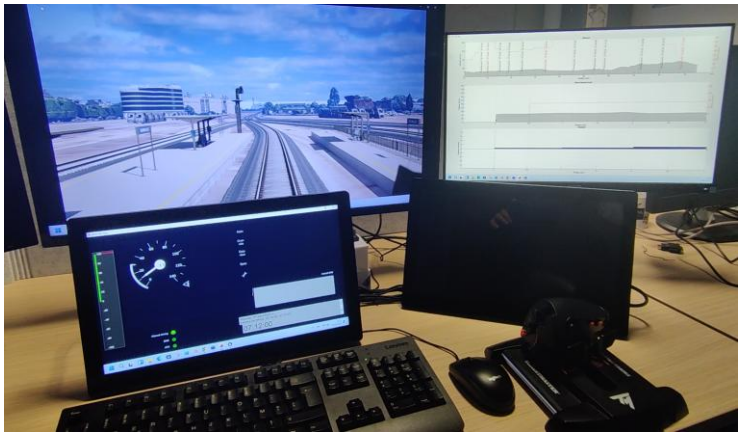


Figure 26: ATO GoA2 simulation workstation

Two custom interfaces were developed using App Designer to support user interaction and analysis:

1. A DMI-like display, showing the applied control command (traction/braking), current speed, distance to the next Timing Point, ATO engagement status and DAS recommendations (if active).

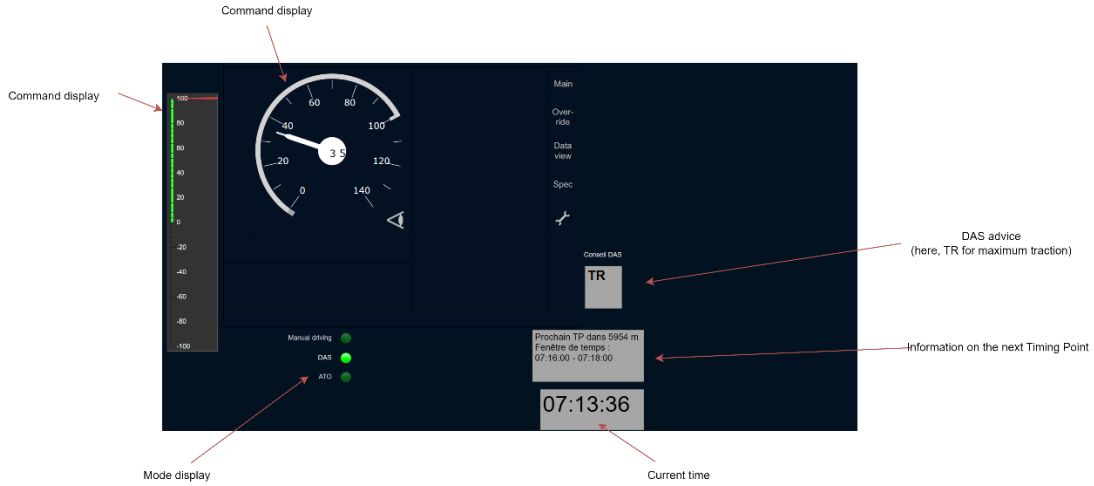


Figure 27: DMI-like interface

2. A mission visualization interface, featuring three synchronized graphs:
    - Top graph: Mission overview, including:
        - Stopping point (red dashed vertical lines) and passing point (black dashed vertical lines).
        - Speed limits (red dashed profile)
        - Track elevation (gray area).
        - Current train position (blue dot).
        - When ATO is active, the optimal speed profile (green) and the train's position following this profile (green dot).
    - Middle graph: Speed profile around the current position, with:
        - The same elements as the top graph.
        - In ATO mode, the optimal profile (green) and the MPC's planned trajectory (blue circular markers beyond the current position).
    - Bottom graph: Track gradient, aligned with the same position scale as the middle graph.

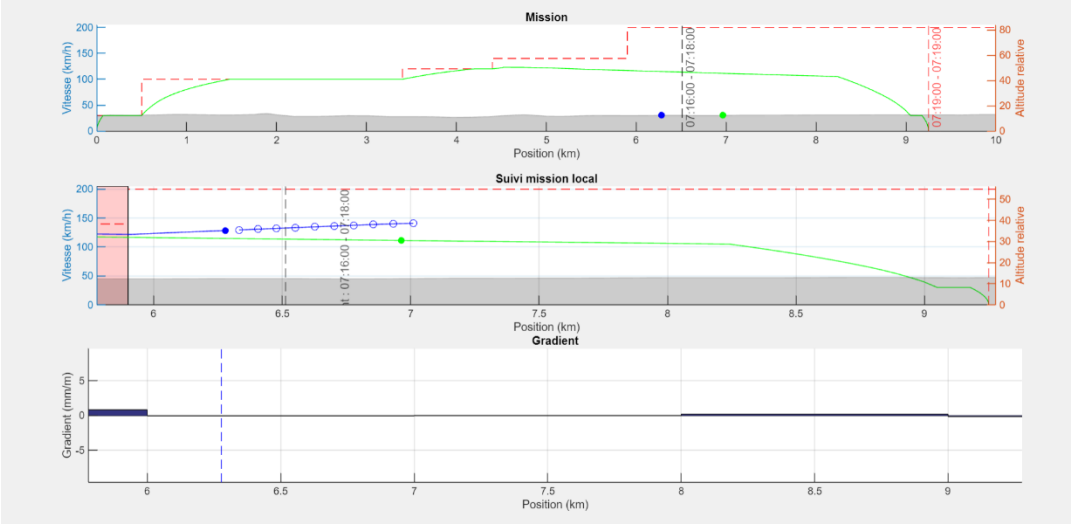


Figure 28: Mission visualization interface

This type of light simulator allows drivers, engineers, and stakeholders to interact with the ATO GoA2 and DAS systems in a controlled, risk-free environment. This facilitates ergonomic assessments of the interfaces and validation of engagement/disengagement logic, ensuring seamless integration with real-world operations.

The simulator also serves as an engaging communication tool: its immersive, interactive interface lets non-specialist audiences explore how an ATO works, experience its effects on driving, and understand the system's key stakes (energy, punctuality, safety).

4.1.6 Prototyping : deployment of MPC controller on Speedgoat real-time target

This prototyping activity aims at demonstrating the capability to transition a simulation model from Model-in-the-Loop (MIL) to Processor-in-the-Loop (PIL) / Hardware-in-the-Loop (HIL) execution, a key step in the V-cycle validation process.

Code Generation

Automatic code generation was successfully performed for the MPC controller and the train dynamics model using Simulink Coder. However, the speed profile generation algorithm could not be compiled, as it relied on functions not supported by the code generation toolchain. Consequently, the reference speed profile was implemented as a static input in the deployed model.

This experience underscores the importance of designing algorithms with code generation in mind from the beginning of the developments. Early consideration of toolchain constraints—such as avoiding unsupported functions or data types—would prevent late-stage redesign and ensure smoother transitions from MIL to PIL/HIL.

Hardware architecture

The prototyping setup consists of two Speedgoat real-time target machines communicating via UDP:

- Speedgoat A: Hosts the train dynamics model, simulating the physical behavior of the train (position, speed) in response to control commands. This target effectively replaces the virtual plant used in MIL simulations.
- Speedgoat B: Runs the MPC controller alongside a preloaded static speed profile. The controller computes optimal control actions in real time and transmits them to Speedgoat A, closing the control loop.

The two targets exchange data via UDP packets, with Speedgoat B sending control commands and Speedgoat A returning the updated train state (position, speed, acceleration). This communication mimics the interface between an onboard controller and a real train, enabling early validation of timing and synchronization requirements.

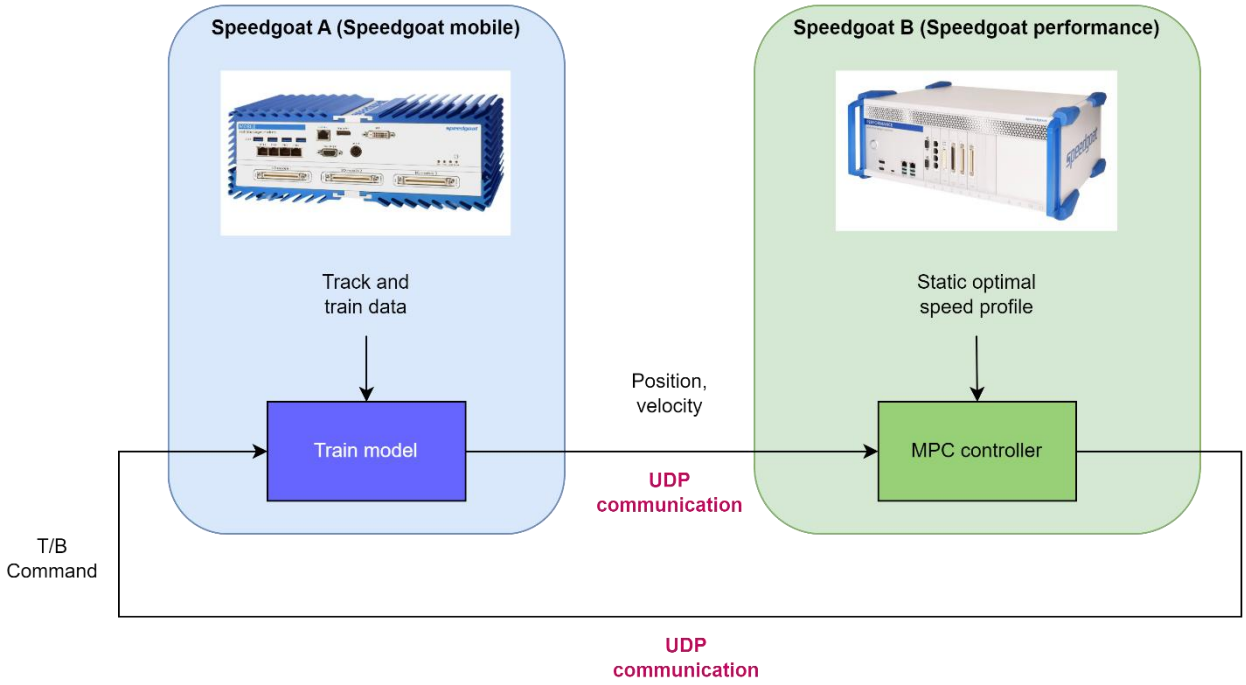


Figure 29: Prototyping demonstration architecture

This setup demonstrates the value of Model-Based Design (MBD) in the V-cycle validation phase. By replacing Speedgoat B with a real industrial ATO system, the platform can be used to stimulate the real system with realistic operational scenarios and verify that system behaviour is compliant with specifications under controlled conditions.

4.1.7 Modelling feedback report

Beyond the energy savings results presented above, the GoA2 ATO modelling work revealed several issues that required interpretation and explicit modelling decisions.

Issue	Description	Modelling decision
Train characterization data	Subset-139 lacks sufficient data for complete train dynamics characterization. Missing parameters include full traction/braking curves across the entire speed range, running resistance coefficients, and the rotative mass coefficient (to account for rotational inertia). Are these parameters intended for ATO-OB configuration?	Assumption that these parameters are part of the ATO-OB configuration. They must be provided as input data in the model.
Time windows at stopping points	Although time windows are defined for passing points, they	A 30-second time window was introduced upstream of the

	are not specified for stopping points. To enable effective optimization, the algorithm must account for a time interval to ensure convergence.	scheduled arrival time at each stopping point to allow algorithm convergence.
Optional stopping points in JP	Including a stopping point in the Journey Profile (JP) is not mandatory. However, this complicates ATO implementation, which benefits from having a clear target point (position and speed). Since speed is not constrained at passing points, they cannot serve as reliable targets.	Test scenarios were designed to always include a stopping point as the final target, ensuring a clear position and speed objective (v=0) for trajectory planning.

Table 5: Specification issues for ATO GoA2

Further extensions of this work could address several directions among which:

- A **multi-agent modelling framework** could be developed to represent multiple trains and their interactions with each other and with the infrastructure, thereby capturing the behaviour of the ATO system as a whole rather than focusing on a single onboard controller.
- The proposed model could be **compared with a real ATO solution** provided by a supplier, in order to validate the modelling assumptions and design choices, and to assess how closely the simulated behaviour matches that of a real deployed system.

4.2 GoA4 SIMULATION RESULS

This section demonstrates how the GoA4 model development leverages a modular and flexible modelling framework to ease system validation, foster collaborative engineering, and seamlessly adapt to diverse test scenarios.

4.2.1 Model architecture and project organization

The architecture of the model reflects the targeted ATO GoA4 architecture, as described in Section 2.2.3, along with the associated assumptions. The model is implemented in Simulink (see Figure 30), and its development follows a structured project organization to ensure modularity, maintainability, and collaborative efficiency.

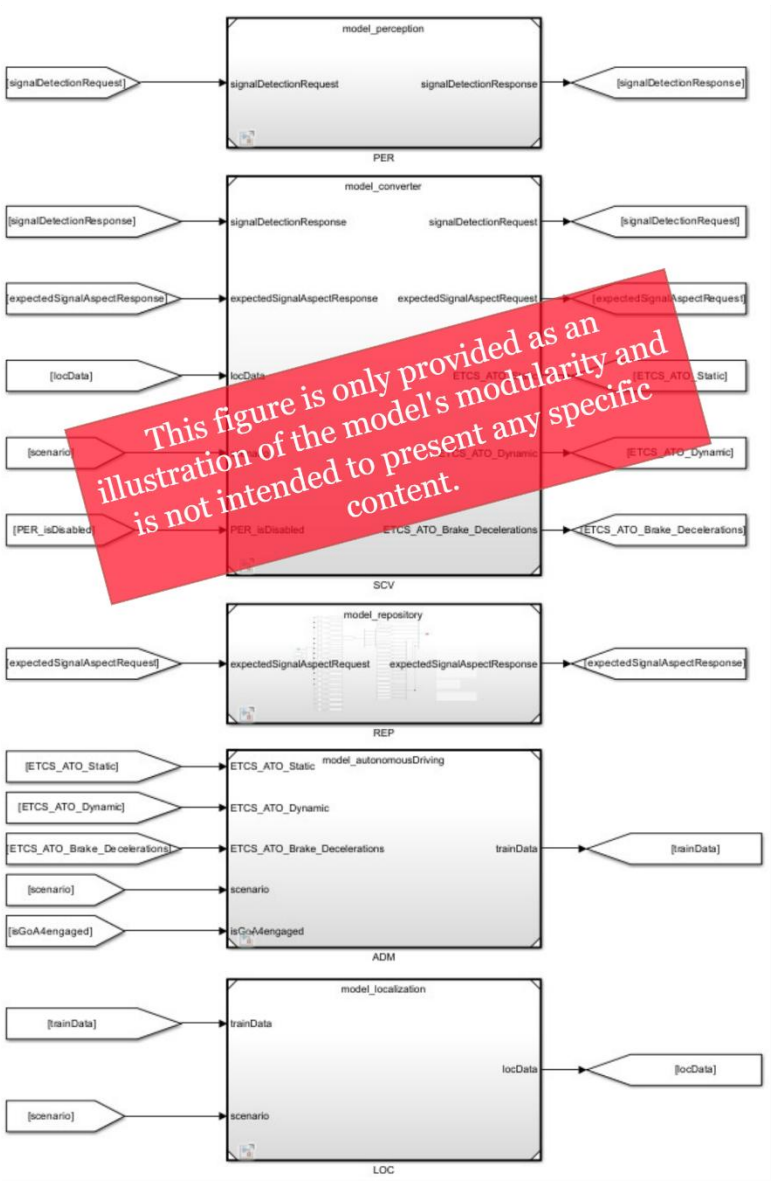


Figure 30: Modular aspect of ATO GoA4 model on Simulink (*model.slx*)

Each subsystem was developed autonomously by different team members, enabling parallel progress. However, because of this additional complexity, clear directory layout is essential for modular development and repeatable integration. The chosen structure separates data, models, scripts and tests to allow each subsystem to be developed and validated independently.

Project Structure (model.prj)

The project is organized into four main directories:

- **Data**: Contains all data required for the global model (*model.slx*), consolidated in a Simulink Data Dictionary (.sldd).
- **Script**: Includes MATLAB scripts (.m), particularly those for simulation initialization.
- **Model**: Houses the global model (*model.slx*) and subdirectories for each subsystem (PER, SCV, ADM, …). The global model references each subsystem via referenced models.
- **Test**: Contains test scripts and models for validation purpose.

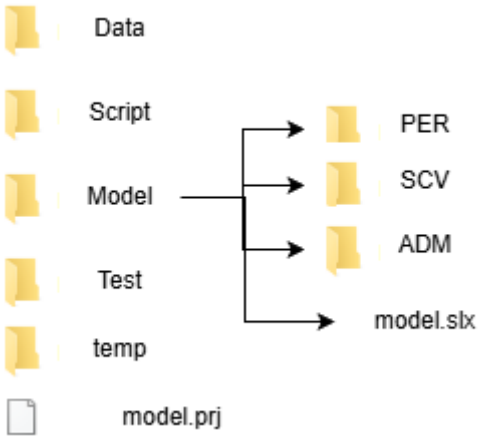


Figure 31: Project structure

Subsystem Organization

Each folder corresponding to a subsystem (PER, SCV, ADM, …) follows a similar project structure:

- **Data**: Subsystem-specific data, stored in a dedicated .sldd file that inherits from the global data dictionary.
- **Script**: MATLAB scripts for subsystem initialization and auxiliary functions.
- **Model**: Contains the subsystem's referenced model, which can be integrated into the global model or used in standalone test harnesses.
- **Test**: Includes test harnesses for subsystem validation.

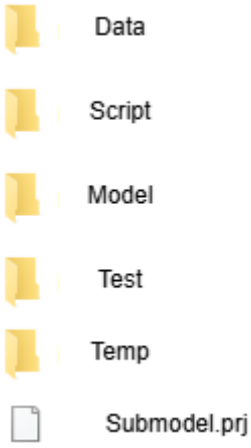


Figure 32: Subsystem project structure

This modular organization offers multiple benefits:

- **Easy Updates**: Subsystems can be updated or replaced by simply swapping their respective folders, facilitating iterative development. The structure also allows independent testing of the algorithmic loop by replacing a subsystem with a simple "fake" module that delivers desired output.
- **Data Consistency**: Inheritance from the global data dictionary ensures shared parameters remain coherent across subsystems.
- **Test Accessibility**: Test harnesses for each subsystem are centrally located.

A key preliminary step involved defining interface data between modules. This ensured clarity on expected data exchanges in various scenarios, limiting integration issues and promoting interoperability.

4.2.2 Validation approach

4.2.2.1 Methodology

The modular architecture of the ATO GoA4 model not only streamlines development but also enables a structured and scalable validation process. By isolating components and progressively integrating them, we ensure robustness at each level of the system. The validation approach is organized into three complementary phases:

1. **Unit Validation**

   Each subsystem (PER, SCV, ADM, …) undergoes standalone validation using handcrafted test inputs that cover nominal and edge cases. Unit validation is carried out through a test harness located in the Test folder of the subsystem. Unit validation focuses on the correctness of individual algorithms and logic and ensuring compliance with subsystem-specific requirements. This step guarantees that each module behaves as expected in isolation before integration, reducing the risk of cascading errors.

2. **Pairwise Interface Validation**

To ensure seamless interaction between subsystems, we perform pairwise validation, where two subsystems are interfaced and tested together. This incremental approach allows us to validate data exchanges and communication protocols between modules and identify and resolve interface mismatches early in the process. Pairwise validation allows to isolate and diagnose errors more efficiently, as the scope of potential issues is limited to the two interfaced subsystems.

3. **Global Chain Validation**

   Once all subsystems are validated individually and in pairs, the full ATO GoA4 model is tested under realistic and degraded conditions by running end-to-end predefined test scenarios, including nominal and off-nominal conditions.

By combining these three validation phases, we ensure a comprehensive and efficient verification process.

4.2.2.2 Scenarios

Validation is carried out through 3 scenarios already mentioned in D31.4

ID: 01 Track free

Description: Depot to station track free, direct track reception

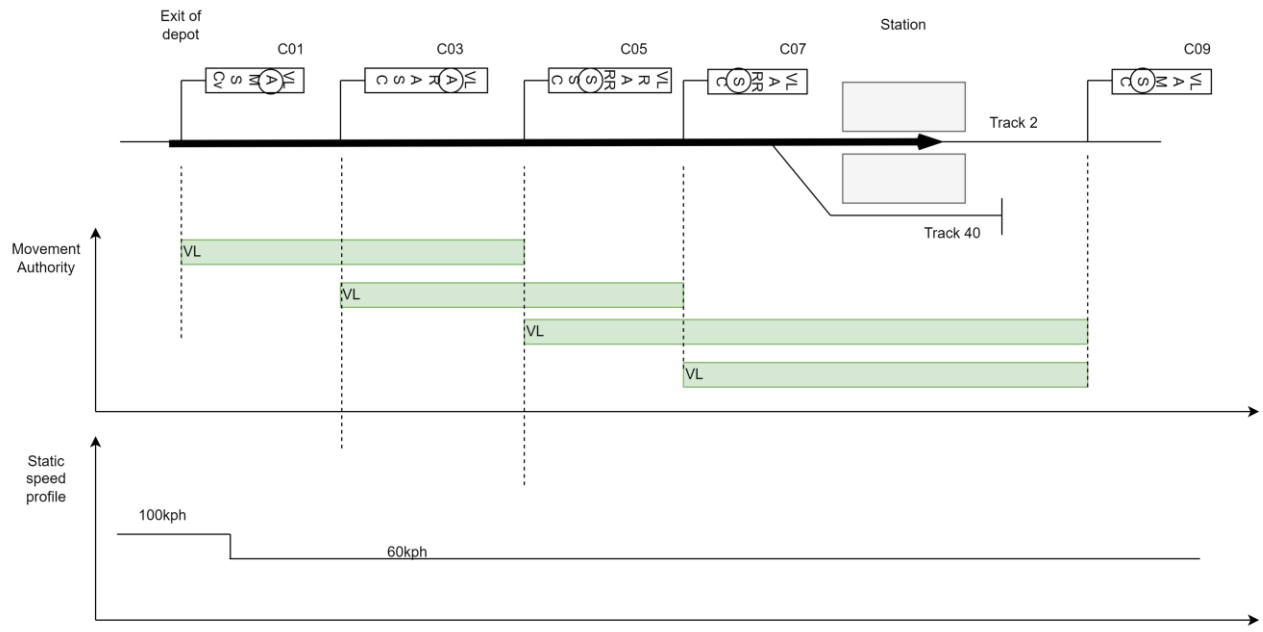


Figure 33: Track free scenario

Expected result:

- Set in motion
- Compliance with maximum static speed constraint
- Detection of track free and no brake
- Stop at platform mark at station

ID: 02 Stop

Description: Depot to station warning and non crossable stop, signal opening, direct track reception

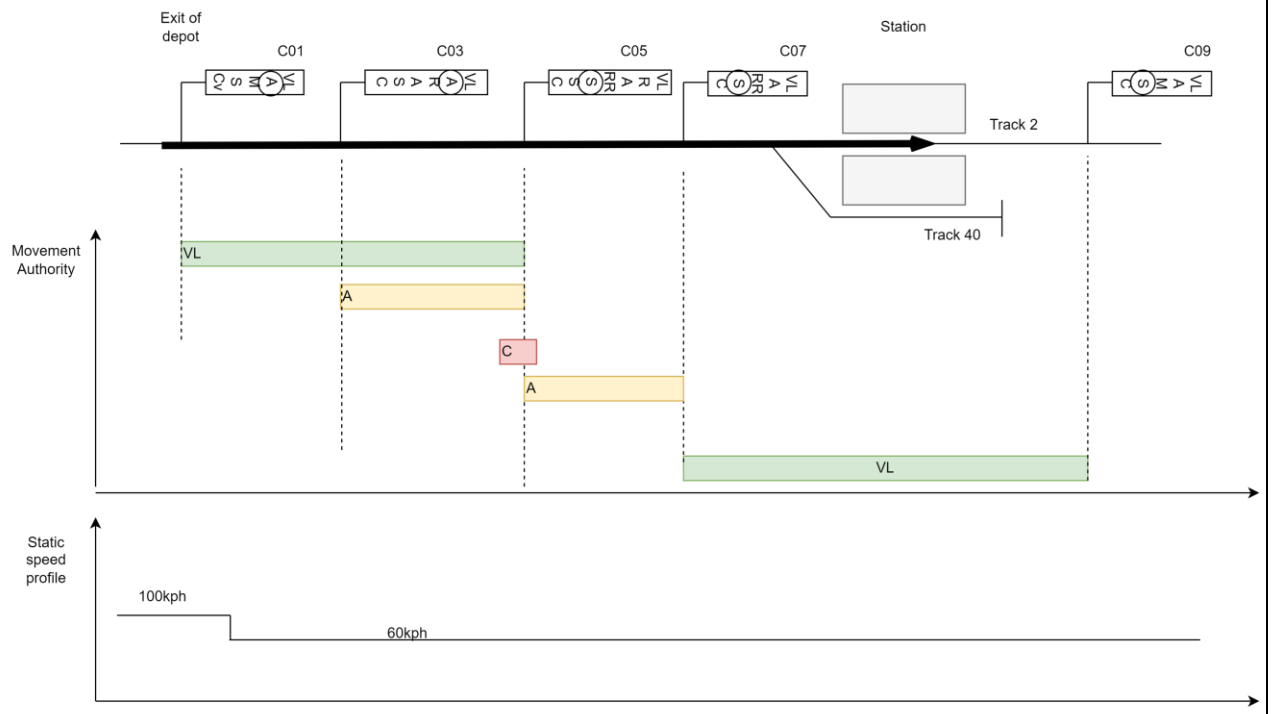


Figure 34: Stop scenario

Expected result:

- Set in motion
- Compliance with maximum static speed constraint
- Warning detection (C03) and speed decrease
- Non crossable stop detection and stop before signal
- Detect change: stop to warning
- Detect track free (C07)
- Stop at platform mark at station

ID: 03 Switch crossing

Description: Depot to station 30km/h announcement, 30km/h execution, reception on deviated track

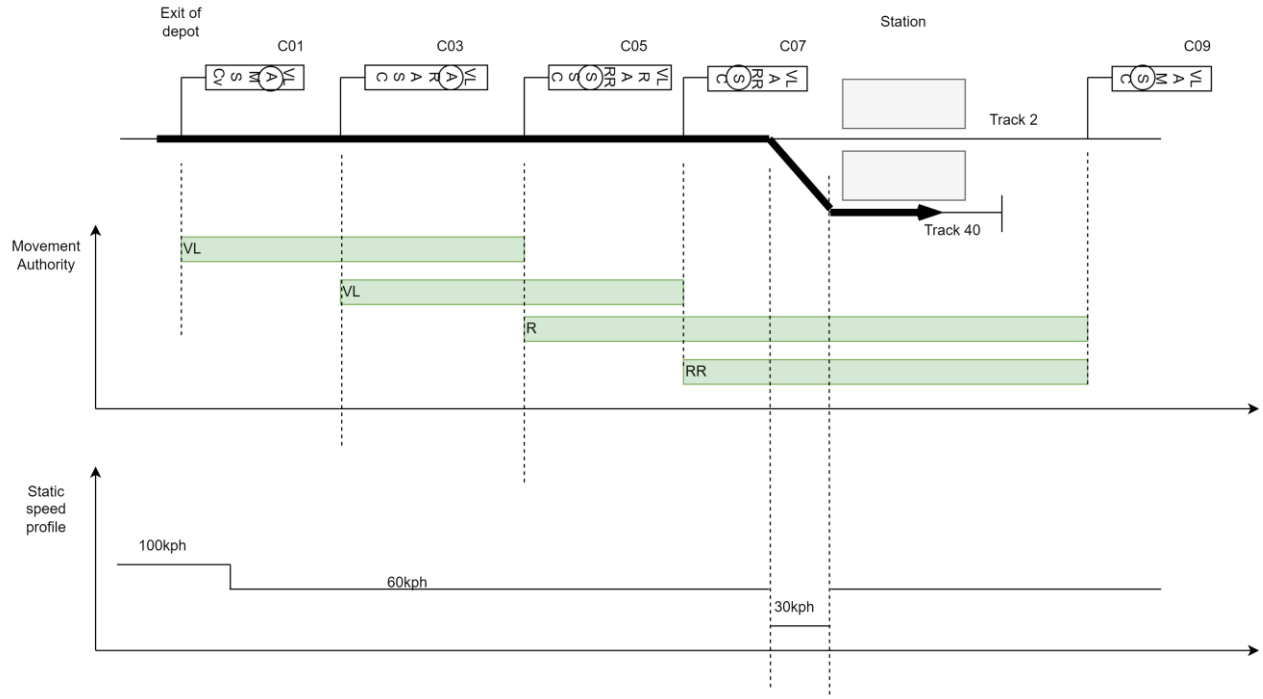


Figure 35: Switch crossing scenario

Expected result:

- Set in motion
- Compliance with maximum static speed constraint
- Detect track free and no brake
- Detect signal announcement for speed reduction to 30kph: reduce speed
- Detect signal speed execution: maintain speed at 30kph on switch
- Stop at platform mark at station

4.2.3 Perception (PER) module validation

4.2.3.1 Overview of the PER block and its environment

The PER block (which corresponds to the SD block of the WP6 architecture) is the module responsible for detecting lateral signaling signs. It should be noted that it is not responsible for any other type of detection, such as obstacle detection, and relies solely on image processing. It is not a multimodal system combining data from multiple sensors such as LiDAR.

Working environment

The PER block detects railway signals using deep-learning models, its development is multi-platform and distributed across four complementary environments:

- **Capella on Windows** *(version 7.0):* functional architecture design.
- **Python on Linux** *(version 3.12):* development and training of computer vision models.
- **MATLAB/Simulink** *(version R2024a):* integration with ATO control logic.
- **Unreal Engine** *(version 5.1):* 3D environment modelling and real-time simulation.

Interfaces

The PER block exchanges data with two simulation components, “3D Environment” and “SCV”, as shown in Figure 36.

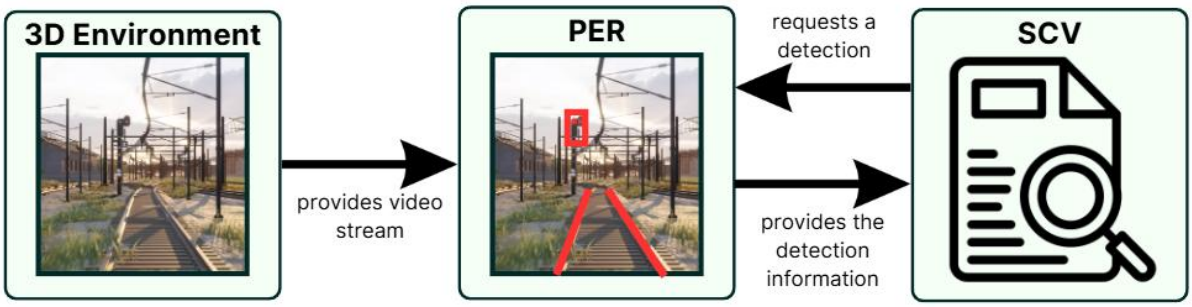


Figure 36: PER module interfaces

The 3D Environment block (Unreal Engine) provides a real-time camera view from the driver’s cab and supports dynamic changes of simulation conditions (weather, illumination, signal states).

The SCV conversion block requests detection when a railway signal is nearby and receives detection responses from PER composed of a railway signal type, a railway signal state and a binary reliability flag (detection confidence indicator).

PER functional pipeline

PER extracts railway signal information from a video stream and detection requests through five processing stages:

- **Data acquisition:** ingest video stream (frame by frame) and detection requests.
- **Pre-processing:** prepare raw data for analysis, including normalization, cropping, etc.
- **Detection:** locate and classify signal using deep-learning models in each frame.
- **Temporal processing:** suppress spurious detections and improve robustness via sequential filtering.
- **Result formatting:** produce structured outputs consumable by downstream blocks.

For more details, the block diagram of the PER module is provided in Figure 121 *(Appendix 1).*

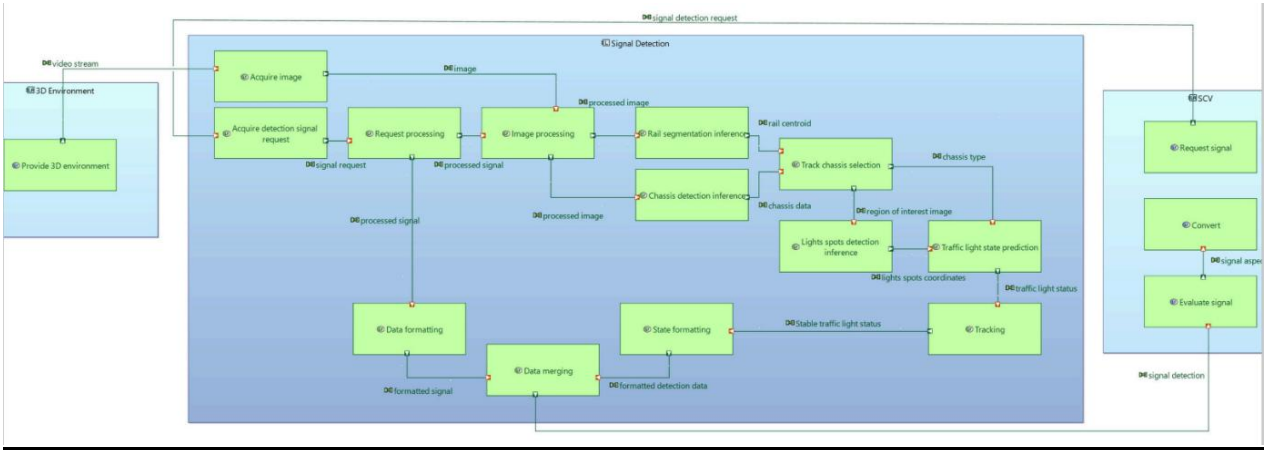


Figure 37: Capella diagram (logical layer)

The functional diagram (Figure 37) developed using the MBSE approach with Capella (see D31.4) served as the foundation for building the PER block architecture in Simulink.

Detection logic

First, the system detects all candidate railway signals in each frame, returning for each candidate the class (signal type), the bounding region (region of interest), and the confidence score. In Figure 38, the railway signalization detection outputs are highlighted in red. A single frame may contain multiple detected signal bounding boxes.

If multiple railway signals are detected, the system performs a rail segmentation step to identify the rail centroid, the candidate closest to the rail centroid is selected as the most relevant signal.

Within the selected signal's region of interest, the system detects individual lamp spots to determine which lamps are illuminated. This information is used to infer the overall signal state.

In Figure 38, you can see the three key steps involving three Deep Learning perception models used to determine the type and state of the signal in each frame.

The results are then consolidated over a 70-frame sliding window, roughly 3 seconds at 25 FPS, which provides a sufficient time span to reliably detect flashing lights and filter detection errors.

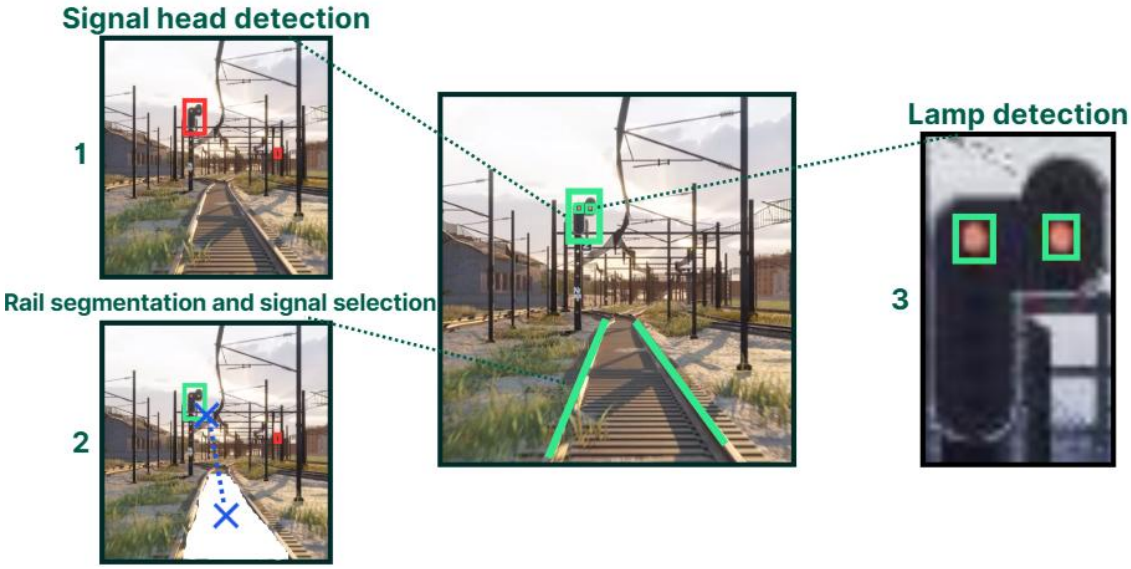


Figure 38: Detection logic schema

Development objectives

The objectives set for the development of the PER module are as follows:

KPI	Description
Detection score per image	Design a model capable of identifying railway signals with a **confidence rate of ≥ 90%** on a real images test set covering 4 signal types and 32 signal states.
Robustness under optimal conditions	Validate the model on **100% of the nominal scenarios** under optimal conditions to ensure reliability and identify potential weaknesses.
End to end FPS	Ensure a throughput of **10 FPS** (frame per second) during real-time demonstrations by optimizing inference time and computational efficiency.
Memory efficiency	Limit the artificial intelligence model size to **less than 50 MB** to ensure smooth and lightweight deployment.

KPI	Description
Degraded scenarios	Ensure that no significant degradation in vision model performance is observed in degraded scenarios, such as adverse or degraded weather conditions.

Table 6: PER development objectives

Validation approach

The goal is to verify the functional compliance and robustness of the system (chain: 3D Environment → PER → SCV) through unit tests, integration tests, and comprehensive scenario testing.

The validation plan is as follows:

1. Unit validation of subsystems and interfaces:
   - Computer vision models: Validation of the models on real-world data.
   - 3D Environment ↔ PER: verification of rendering, real-time video stream, and dynamic environment management.
   - PER ↔ SCV: verification of data exchanges (detection requests and responses).
2. Functional validation of the complete PER model:
   - Validation of the PER module across a series of simulated test scenarios.
   - Verification of the objectives defined for PER module development.

4.2.3.2 Development and unit validation of computer-vision models

This subsection presents the design pipeline for deep-learning perception models, illustrated through the specific models developed for this project. All training, evaluation, and performance metrics are based on real-world data, with the goal of assessing how well models trained on real data generalize to simulated environments.

Problem definition and success metrics

As with any development effort, it is essential to start from a clear objective and a precise idea of the expected outcomes.

The goal is to design a perception pipeline (detection + segmentation) that can operate in near real-time on medium-quality images in French railway environments, using only limited annotated datasets. The technical challenge is to achieve reliable performance (accuracy and robustness) while controlling execution time and model size. In addition, the model must be exportable and integrable into a MATLAB environment using a GPU.The KPI are provided in Table 6.

Defining these elements up front provides a roadmap for what we want to achieve and how we will proceed.

1. **Data collection & preparation**: inventory of real data, targeted annotations, dataset splits.
2. **Baseline & fine-tuning**: select a fast pretrained model as baseline, then transfer and fine-tune on French railway data.
3. **Evaluation**: compute the metrics listed above and run robustness tests.
4. **Optimization**: apply quantization, pruning and GPU acceleration to meet latency and size constraints.
5. **Field validation**: final tests under operational conditions.

Figure 39: Main steps when developing an AI model

Data collection and preparation

A major part of building a reliable perception pipeline is assembling a dataset that is fit for purpose in terms of quality, quantity and diversity. We apply two complementary approaches: direct image acquisition with subsequent annotation, and the use of open datasets available on platforms such as Hugging Face, GitHub or Kaggle. All collection activities comply strictly with regulatory constraints, including GDPR (General Data Protection Regulation).

For the rail segmentation model, an open-source Roboflow dataset of approximately 1,000 images was used. This dataset matches our needs:

- **Viewpoint**: perspectives from the driver's cab.
- **Diversity**: varied lighting, rich environments, and changing weather.
- **Precise annotation**: each rail is carefully segmented, providing a solid base for learning.

For the chassis detection model, the starting point for chassis detection is an internal team dataset that resembles the data described in the FRSign datasheet.

Figure 40: Visualization of image for training the rail segmentation model

For the lamp detection model, we compiled an image corpus from the FRSign and GERALD (German signals) datasets. By extracting ROIs around the chassis, we obtained an initial corpus of roughly 20,000 images.

Figure 41: Visualization of image for training the lamp detection model

After data collection, a thorough analysis has been performed to assess dataset quality. This step aims to identify strengths, weaknesses, potential biases and areas for improvement.

1. **Domain coverage**

The goal is to ensure the dataset covers all targeted classes, a variety of lighting conditions, different viewing angles and multiple seasons and weather contexts. Complete coverage is essential to guarantee model generalization.

2. **Biases and imbalances**

Statistical analysis of the dataset reveals possible over- or under-representations. A notable example concerns the FRSign dataset used for chassis detection, the study identified a strong over-representation of certain signal states, notably classes A and H.

This imbalance introduces bias during training, the model is more exposed to those specific signals and risks overfitting their characteristics at the expense of other classes, as illustrated on Figure 42.

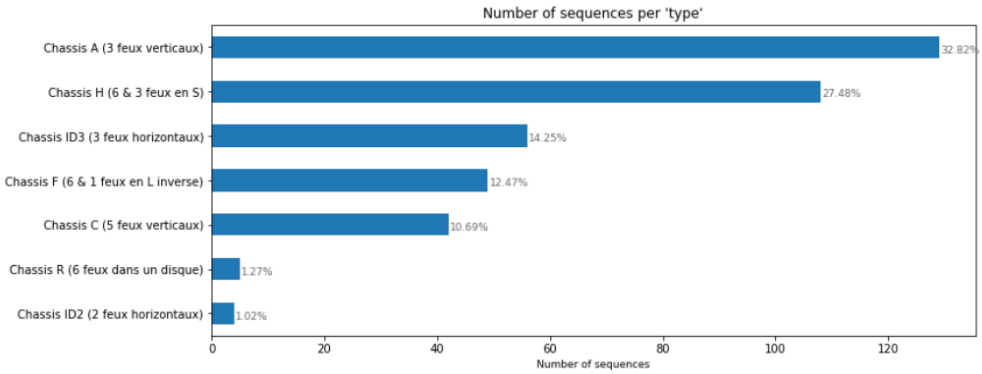


Figure 42: FRSign number of sequences per type

3. Labeling and Data Governance

A rigorous labeling process is essential to produce a usable dataset. Annotation is performed with LabelStudio or an equivalent tool, these platforms provide controls that increase annotation reliability. Datasets are accompanied by essential metadata such as date and version, data origin and standardized manifest (JSON/YAML). This versioning ensures reproducibility of the training pipeline.

Some datasets arrive unlabeled, so we perform the annotation ourselves. The classic approach is to label each image manually, one by one. However, automation techniques can accelerate this task, for example, we applied pseudo-labeling to annotate the lamp-detection dataset. Pseudo-labeling is a semi-supervised technique in which a model trained on a small, labeled set generates labels for unlabeled data.

The workflow used for lamp dataset pseudo-labeling includes:

- **Initial manual annotation**: select and annotate 400 diverse images, chosen carefully according to clustering.
- **Quick training**: train a YOLOv11-nano model on this sample for approximately 50 epochs.
- **Automatic annotation**: apply the lightweight model to ~5,000 images to auto-generate YOLO-format annotations.
- **Manual validation** via a simple Tkinter interface:
  - central pane shows the image with the model's annotation,
  - right-arrow (keyboard) = "accept", left-arrow = "re-annotate",
  - mouse support to redraw boxes if needed.

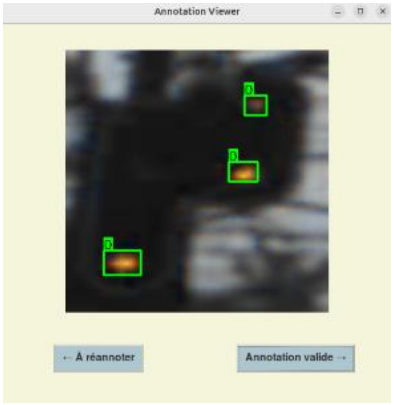


Figure 43: Manual validation interface

The result was a high-quality, manually validated dataset of ~5,000 annotated images ready for analysis.

4. In-depth Analysis & Dataset Corrections

A detailed statistical analysis not only assesses overall quality but also drives corrective actions. The data exploration and preprocessing stage transforms the raw corpus into a clean, consistent dataset ready for training. This phase combines statistical analysis, targeted corrections, and reproducible transformations.

Key tasks during analysis and preprocessing:

- Inspect core distributions: relative object sizes, box counts per image, class frequency, area/aspect-ratio histograms.
- Examine correlations between attributes (object size vs. image position, brightness vs. error rate).
- Identify and visualize outliers (aberrant annotations, excessively noisy images, extreme object positions).

Then, the data must be cleaned:

- **Deduplication**: remove identical or near-identical images (same scenes/frames). For example, the lamp-detection dataset contained many near-duplicates coming from video sequences, which risked biasing training. We removed duplicates using **DBSCAN**, ensuring each retained image contributes new information. DBSCAN is an unsupervised, density-based clustering method that groups similar images without requiring a predefined number of clusters, identified duplicates were removed to reduce repetition and maximize fine-tuning efficiency.
- **Annotation correction**: manual and automated checks to fix badly placed masks/boxes and to harmonize labels (typos, aliases).
- **Format harmonization**: normalize images (RGB vs. grayscale), resolutions, annotation formats and metadata fields.

For example, with FRSign, class distribution analysis revealed heavy concentration on nine principal states, several rare classes were underrepresented, while classes A and H were oversampled, as Figure 44 shows. This imbalance could distort training by causing the model to overfit the dominant states. To address this, we implemented dataset rebalancing strategies, notably targeted data augmentation for underrepresented classes.

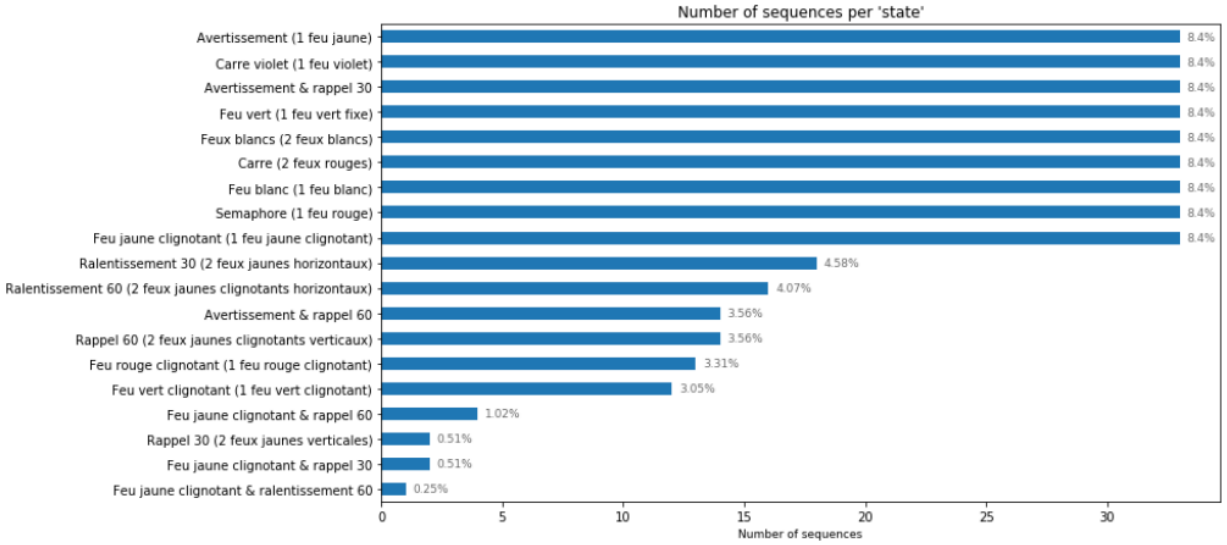


Figure 44: FRSign number of sequences per state

5. Targeted Data Augmentation

Data augmentation is applied in a controlled manner to artificially increase dataset diversity while preserving visual realism. Its objectives are to:

- strengthen under-represented classes,
- improve robustness to contextual variations,
- reduce overfitting to overly frequent configurations

We select augmentation techniques according to their relevance for each object type to avoid transformations that would make scenes unrealistic. Commonly used operations include flips, rotations, blur, brightness/contrast variation, zoom, partial occlusion, and mosaic compositing.

For our detection and segmentation training datasets, the augmentation methods described in Table 7 were used.

Method	Rails	Chassis	Lamps	Notes	Images
Flips	Horizontal only	Horizontal only	Horizontal & vertical	Vertical flip would make rails/chassis silhouettes unrealistic; lamps tolerate both.	
Rotations	±10°	±20°	±180°	Rails are very sensitive to tilt; chassis are more tolerant; circular lamps can rotate fully.	
Brightness	±20%	±20%	±40%	Moderate range for rails/chassis; larger range for lamps to simulate intensity/glare.	
Contrast	±20%	±20%	±35%	Same rationale as brightness.	
Blur / Sharpness	Blur σ ∈ [0, 1.5] / Sharpness ∈ [0.75, 1.25]	Blur σ ∈ [0, 1.8] / Sharpness ∈ [0.7, 1.3]	Blur σ ∈ [0, 2.0] / Sharpness ∈ [0.6, 1.4]	Lamps are more prone to strong blur/glare; avoid extreme blur on rails.	
Translations	±12%	±15%	±20%	Moderate shifts to preserve context; lamps are more mobile in the scene.	
Zoom	[0.9×, 1.15×]	[0.85×, 1.2×]	[0.7×, 1.3×]	Larger zoom range for lamps (point lights); tighter ranges for rails.	

Method	Rails	Chassis	Lamps	Notes	Images
Occlusions / Cutouts	Up to 12%	~8%	~6%	Rails can tolerate higher occlusion; avoid completely masking the object.	
Mosaic (2×2)	Yes (4 images)	Yes (4 images)	Yes (4 images)	Useful to vary context and object scales.	

Table 7: Augmentation methods

After applying these augmentations, the final class distribution shows a significant rebalancing, supporting more homogeneous learning across classes.

6. Adaptations to Maximize Performance

Some preprocessing and transformation steps can be applied to make the model's task easier and improve generalization:

- **Normalization / standardization** of pixel values and derived features.
- **Stable categorical encoding** for any discrete attributes.

For example, in the rail segmentation task we observed that YOLO generalized better when trained on images converted to grayscale.

Figure 45: Rail segmentation

Additional analysis that can be performed depending on the task includes distribution of the number of bounding boxes per image, relative object area within the image and typical object positions in the scene.

The overall objective is to make the dataset as clean, consistent and representative of real operational conditions as possible.

Dataset Splitting Strategy: Training / Validation / Test

The splitting and validation strategy determines the reliability of performance estimates. The choice depends on dataset size, presence of temporal dependencies, and class balance.

A simple, fast approach is to split the dataset into three subsets, commonly:

- Training set (70% of dataset): data used for training the model, i.e. fit the parameters (weights) of the model.
- Validation set (15% of dataset): dataset used to provide an unbiased evaluation of the trained model. It is used for tuning the hyperparameters of the model.
- Test set (15% of dataset): dataset used to provide an unbiased evaluation of the final model, after tuning the hyperparameters.

This method is effective but can be unstable when classes are rare or heavily imbalanced.

For smaller datasets we favor Stratified K-fold (with K = 5 for example) because it provides more robust performance estimates. Stratified K-fold preserves class proportions within each fold, which is essential when class imbalance is significant. The method yields K different train/validation splits and reduces variance in metric estimates compared with a single split.

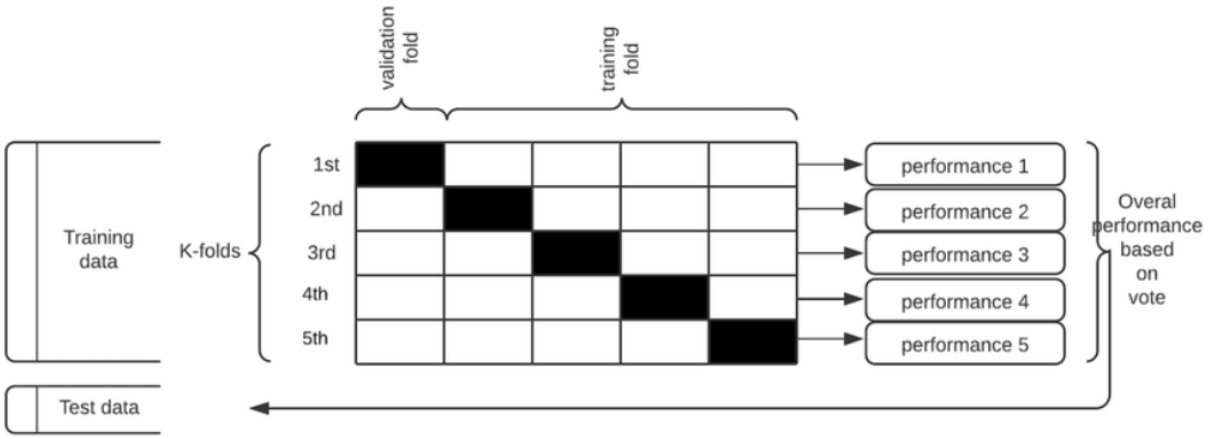


Figure 46: Stratified K-fold method

Important rules for splitting and augmentation

- Ensure that splits maintain representativeness across classes and acquisition conditions (lighting, viewpoints, sequences…).
- Apply all augmentation and dataset manipulations only to the training subset. Validation and test sets must contain only original, non-augmented images to provide unbiased estimates of model generalization.

Baseline and fine tuning

Model architecture selection

First, based on the problem we must choose the appropriate model type (segmentation, detection, classification, re-identification, etc.), since each family has distinct strengths and trade-offs.

- **A segmentation model** produces a per-pixel mask that outlines the target object. It typically requires more compute but provides precise contours and is therefore well suited when exact shape or boundary information is needed. For this reason, we selected a segmentation model for the **rails**.
- **A detection model** returns object locations as bounding boxes plus class and confidence scores. It is generally faster than segmentation and a good fit when exact contours are not required. We therefore chose detection models for chassis and lamps.

Next, we consider the expected model size and capacity according to constraints and objectives:

- For real-time inference or low compute budgets, prefer lightweight architectures with fewer layers and parameters.
- If latency is less critical and ample training data are available, larger architectures with higher capacity can be selected to maximize accuracy and robustness. The architecture decision balances inference speed, memory footprint, and expected performance for the deployment environment.

For our case we selected YOLOv11-Nano, a compact variant designed to offer a good trade-off between performance and efficiency thanks to optimized building blocks.

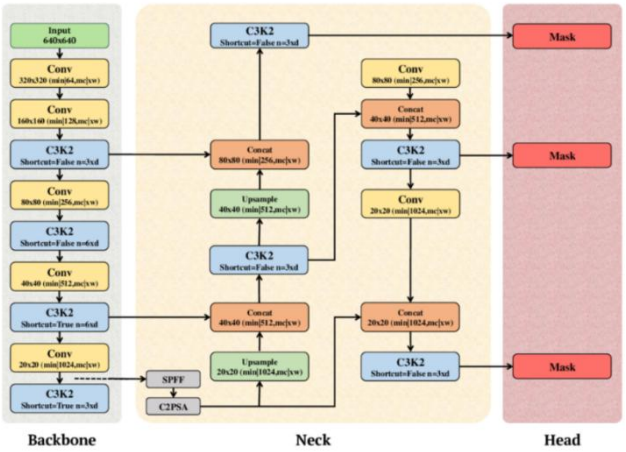


Figure 47: Architecture of the YOLOv11 model

YOLOv11 can be conceptually split into three stages (visualized in Figure 47):

- **Observation (Backbone):** Acts as the model's "eye": it processes the input image and extracts low- and mid-level features.
- **Aggregation (Neck):** Combines features at different scales to provide both detailed and contextual information.
- **Decision (Head):** Produces the final outputs. Depending on the task only the head differs:
  - Detection head: outputs bounding boxes, class labels and confidence scores.

- Segmentation head: outputs pixel-wise masks that assign each pixel to an object class.

The Nano variant contains approximately 2.6 million parameters, making it suitable for low-compute environments. By comparison, a medium variant is already around 20.1 million parameters, trading efficiency for increased capacity and potentially higher accuracy.

Training and validation

Training adjusts the parameters of the pretrained model to optimize detection/segmentation of the target objects and to adapt the model to real acquisition conditions. It is important to structure the process so that it is reproducible, traceable and robust.

To guarantee reproducibility and traceability, the training pipeline should be standardized, for example by enforcing:

- code and dataset versioning,
- a locked virtual environment,
- intermediate checkpoints and the best final checkpoint saved according to the chosen metrics at intermediate stages of training (files containing model architecture and weights),
- learning curves to visualise training dynamics and detect anomalies such as overfitting,
- validation snapshots showing annotated validation images with model predictions and Grad-CAM heatmaps

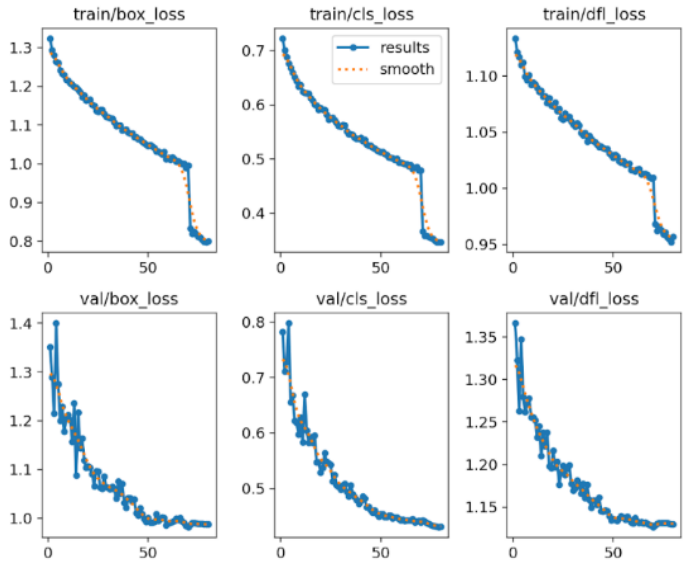


Figure 48: Learning curves for the lamp detection model

Hyperparameter tuning can be automated (for example using Optuna), which intelligently explores the parameter space and identifies promising combinations. In our case, after selecting optimized hyperparameter candidates, we ran the full training while enforcing early stopping and checkpointing mechanisms to guard against overfitting.

Once training is completed, we diagnosed the learning curves. For the lamp detection task, a gap appeared around epoch 70, the box loss decreased on the training set, but this improvement was not reflected in validation, as shown on Figure 48.

This pattern suggests a shortcut learning behaviour or a change in training conditions rather than a genuine, generalizable improvement. As a precaution, we selected the checkpoint immediately before this gap, it offered comparable performance, in the validation sets, while minimizing the risk of choosing a biased iteration.

To validate the computer vision models, we notably relied on two key metrics, conducting tests on data of the same type as that used during training.

- **F1 confidence curve** allowes to visualize how the F1-score (the harmonic mean of precision and recall) evolves with the confidence threshold. This helps identify the optimal operating threshold by balancing precision (how many detected objects are correct) and recall (how many real objects are detected). For example, Figure 49 shows the F1-confidence curve for rail segmentation. The curves has a peak of approximately 0.85 at threshold ≈ 0.37 and a stable plateau between 0.20 and 0.60. To favour recall, a threshold ≈ 0.30 was selected. In our case, recall is prioritized because missing parts of the rail (false negatives) can critically impact the performance. In contrast, false positives are generally tolerable and can be filtered out through post-processing, making a high recall essential to ensure rail continuity and system robustness.

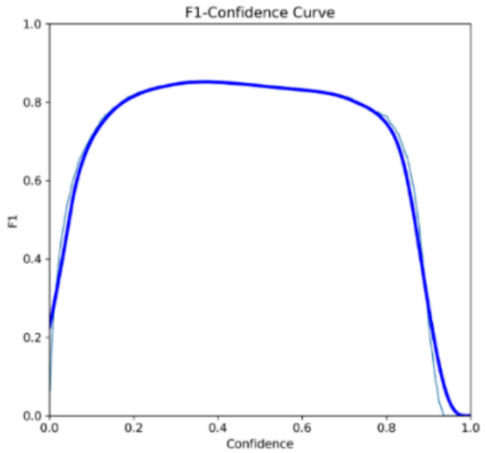


Figure 49: F1-confidence curve for rail segmentation

- **Normalized confusion matrix:** provides class-wise detection performance by showing, for each class, the proportion of correct predictions and the types of misclassifications. For example, the confusion matrix of chassis detection is presented on Figure 50. Correct classification reaches 97% across types A, C, H and F. Inter-class confusion is negligible.

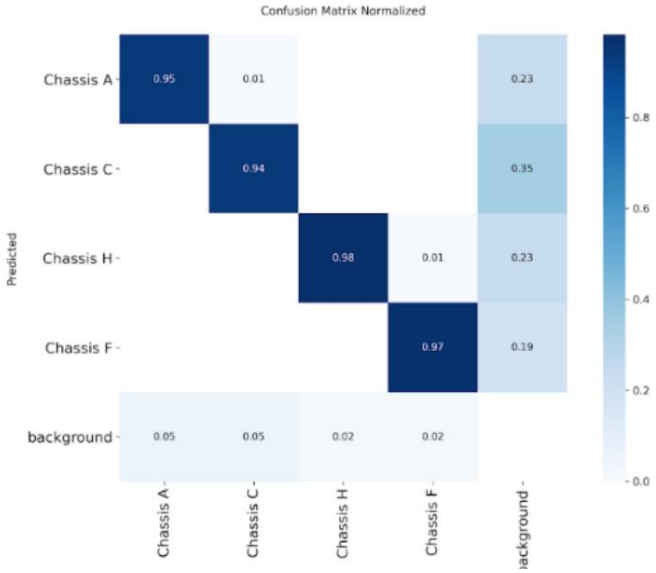


Figure 50: Confusion matrix of chassis detection

All the evaluated metrics help select the most suitable vision models, ensure their performance, and prevent overfitting. They also allow selecting the most appropriate thresholds to achieve optimal performance for the intended use case. The three models are validated with high accuracy and confidence. The next step is to optimize those models further to improve inference speed for deployment.

Evaluation and Optimization

After training, the next phase is optimization: reduce model footprint and latency while preserving task performance. The primary techniques applied in our pipeline are **pruning**, **quantization**, and **model formatting/export**.

- **Pruning**: Remove connections or channels that contribute little to final predictions. Pruning reduces parameter count and model size, which shortens inference time and memory use.

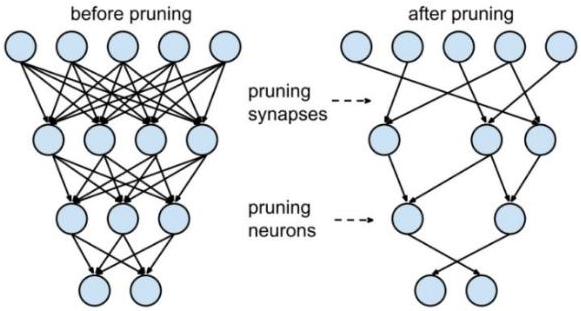


Figure 51: Pruning method

- **Quantization:** Reduce numerical precision of weights and activations to accelerate inference and lower memory usage. In our workflow we used static quantization (calibration on a representative image set) to move from 32-bit floating point (FP32) to 16-bit precision (FP16). This preserves near-original accuracy while improving throughput and reducing memory bandwidth.

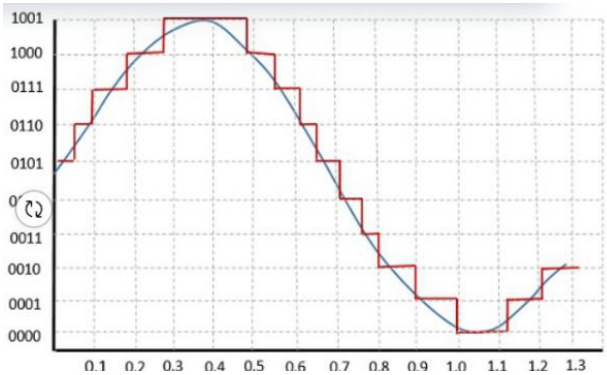


Figure 52: Quantification method

- **ONNX export / model formatting:** Exporting the model to an interoperable format such as ONNX enables easy integration into different runtime environments and hardware backends (MATLAB, inference engines, edge runtimes). ONNX also facilitates further optimization.

Optimization is validated by before/after comparisons on the target platform (here, it is MATLAB/CPU). Typical measures include model size (MB), inference throughput (FPS) and the core task metrics (IoU / Dice / mAP / F1).

Example for the rail segmentation model:

- Optimized model size: **11 MB**, **~5× smaller** than the original (~55 MB).
- Inference speed: **56 FPS**, **~3.5× faster** than the original (~16 FPS).
- Task metrics (IoU, Dice): **practically unchanged** between original and optimized models.

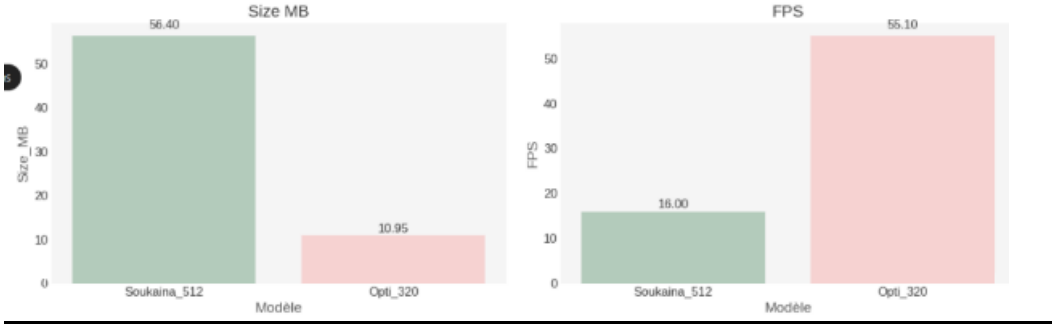


Figure 53: Comparison graph of pre (green) and post (red) optimization models

These results show the optimization achieved a favorable trade-off: substantial gains in size and speed while maintaining segmentation quality.

Interpretability

Interpreting the decisions made by the models is a major concern in AI, it helps verify that the model relies on meaningful features, exposes potential biases, and, when satisfactory, increases confidence in model outputs.

For example, the Grad-CAM method allows users to visualize the attention zone of models during the prediction. With segmentation rail model, attention maps are generally coherent and focus on rails features, like in the first picture of Figure 54. But sometimes, like in the second picture of the figure, overhead catenaries activate attention for rail segmentation due to similar geometry. This was identified as a potential source of bias. No systematic bias was detected, and interpretability analyses support trust in the model's decisions, while also highlighting areas for targeted improvements.

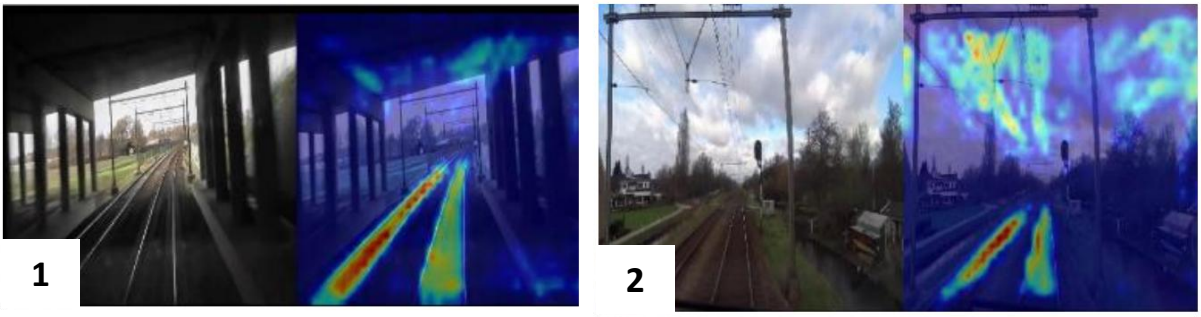


Figure 54: Interpretability of rail segmentation model using Grad-CAM method

Interoperability

After optimization, exporting the model to a standard interoperable format such as ONNX is recommended. ONNX enables deployment across different runtimes and hardware backends and facilitates integration into environments like MATLAB/Simulink.

It's important to know that converting to ONNX does not change the model's core predictive quality (precision, recall, F1, etc.). However, the runtime environment and backend implementation can affect non-functional metrics such as inference latency.

We measured inference times in three distinct contexts to quantify these differences:

- .ONNX executed under Python (typical ONNX runtime or optimized engine)
- .ONNX imported into MATLAB (MATLAB ONNX import/runtime)
- .MAT format executed in MATLAB (native MATLAB model file)

Results are given in Table 8.

Models	Versions	Format	MATLAB environment speed	Python environment speed
Lamp detection	optimized	Matlab	27 FPS	
		ONNX	27 FPS	53 FPS
	not optimized	Matlab	8 FPS	
		ONNX	8 FPS	15 FPS
Rail segmentation	optimized	Matlab	24 FPS	
		ONNX	24 FPS	38 FPS
	not optimized	Matlab	5 FPS	
		ONNX	7 FPS	16 FPS
Chassis detection	optimized	Matlab	26 FPS	
		ONNX	27 FPS	55 FPS
	not optimized	Matlab	15 FPS	
		ONNX	14 FPS	24 FPS

Table 8: Inference time comparison

The Python environment consistently provided the fastest inference times for our models. Differences are primarily due to runtime optimizations and hardware interface efficiency.

4.2.3.3 Interfaces Validation and end-to-end functional validation of PER

This section describes the integration and end-to-end functional validation of the complete perception chain (3D Environment → PER → SCV), including scenario-based evaluations and quantitative analysis of key performance indicators.

Interface: 3D Environment ↔ PER

The first objective is to assess whether the provided Unreal Engine scene is realistic and fit-for-purpose for vision model validation, by visual and analytic checks against a verification checklist covering realism, track geometry, lighting/weather transitions, presence of perturbing objects, and signal fidelity.

We check the following parameters:

- **Realism**: visual inspection of scene elements and material quality.
- **Track geometry**: comparison with the original route track.
- **Weather and lighting**: day / sunset / night / rain / fog…
- **Perturbators**: insertion of confusers like a train with rear lights on.
- **Signal fidelity**: colorimetry and geometric proportions checked against railway norms like the NFF79-189 norm.

The 3D environment met acceptance criteria and was judged compliant for use in model testing.

Figure 55: 3D environment

Figure 56: Disturbance element added to the 3D environment

The second objective is to ensure a stable, exploitable real-time video stream from Unreal Engine to the Simulink ATO model, by continuous stream testing, visual comparison between source frames in Unreal Engine and frames received in Simulink under multiple lighting scenarios.

We observed that stream latency and stability were validated but under low-light conditions, frames received by Simulink appeared darker than the source frames rendered in Unreal Engine.

Thus, we implemented an automatic brightness equalization module in Simulink to restore usable video stream under low-light conditions. A comparison of the images received by Simulink and the images obtained after automatic brightness adjustment is provided in Figure 57.

Figure 57: Automatic brightness adjustment

Video feed is stable and corrected for low light via automatic equalization. However, a noticeable quality gap remains between the Unreal Engine render and the images received in Simulink.

Interface: PER ↔ SCV

The objective is to define and validate a compact, robust exchange format between PER and SCV and verify message compatibility in Simulink.

Our design approach is to normalized exchanges under Capella using domain references such as TAURO, and nonessential variables for our modeling were eliminated to reduce interface complexity while preserving forward extensibility for features like obstacle detection.

The final exchange format is a vector of doubles including:

- **incrementalIndex** : increment index,
- **NID_C** : country identifier,
- **NID_VIP**: virtual information point associated with the light signal,
- **N_FRAME_ITER**: number of frames associated with NID_VIP (the target light signal),
- **NID_FRAME_TYPE**: frame-type code for each detected frame,
- **frame_aspect**: detected frame aspect,
- **M_DETECTION_CONFIDENCE**: detection confidence on the associated image (binary 0/1),
- **M_STABLE**: stability flag on sliding window (binary 0/1).

To validate this format, we executed simulated exchanges for all possible request and response combinations. Message semantics and payloads were verified for conformity and compatibility. This leads to the conclusion that the interface is lightweight, extensible and validated in Simulink.

Verification of PER logic and temporal processing

Temporal filtering: method comparison

The objective is to select the temporal filter that best balances robustness for both blinking and steady lights on 70-frame windows.

A 70-frame sliding window was selected, as it corresponds to approximately 3 seconds of data, which provides an effective compromise for filtering sporadic detection errors while reliably detecting blinking signal patterns.

The following methods have been tested:

- **RLE + short-run removal** (run-length encoding**:** a method that encodes consecutive identical values to compress data, with an additional step to filter very short runs and remove spurious values.)
- **Local mode filter** (each point set to most frequent value in its neighborhood)
- **Custom method** (replace values occurring <5 times with local 10-neighbour mode)

Evaluation metric: RMSE (Root Mean Square Error, measuring the average prediction error) was computed on synthetic 70-point signals containing outliers and missing values across two scenarios to assess the performance of the methods.

Case 1: blinking light: RLE performed poorly. Mode and Custom are comparable and superior.

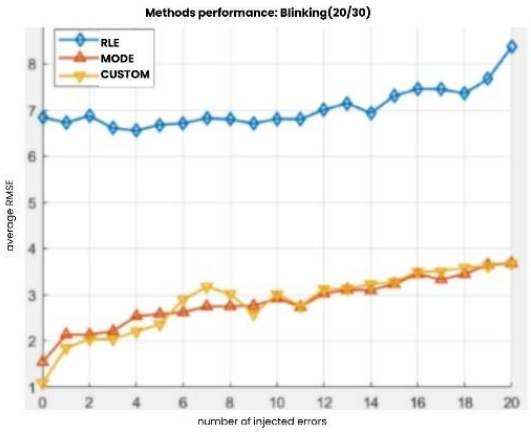


Figure 58: RMSE of the tested method on blinking light

Case 2: steady light: RLE is the best. Mode acceptable and Custom is slightly worse.

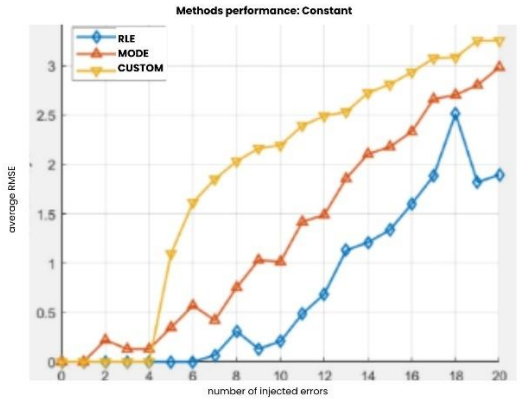


Figure 59: RMSE of the tested methods on steady light

Local mode filter was finally selected as best trade-off for mixed use, blinking and steady, given balanced performance across cases.

Block-by-block logic verification

A block diagram of the PER module logic has been produced and unit tests for each block in Simulink have been executed. This marks the first time our deep learning vision models are being tested on simulated images.

For some blocks, outputs were asserted against expected states and visualized annotated frames were inspected for correctness.

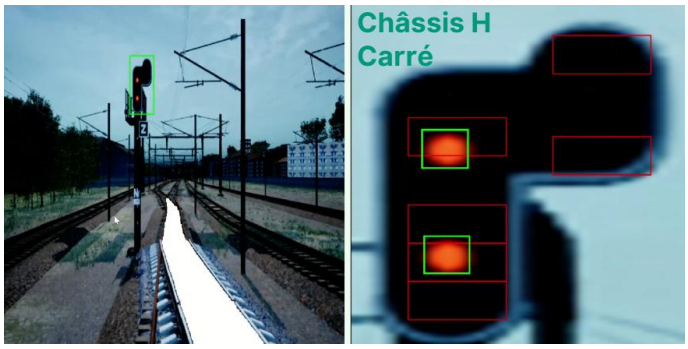


Figure 60: Annotated signal

Conclusion: all blocks were validated individually, and the integrated behavior matches design expectations, with annotated visual checks confirming both model predictions and pipeline operation.

End-to-end functional validation of PER

The objective is to validate the complete perception block: evaluation of PER behavior on target scenarios, quantitative analysis of key indicators, and assessment versus the objectives defined in Table 6.

Test scenarios and procedure

The PER block was validated on three nominal scenarios, presented in section 4.2.2.2.

Each scenario was executed first in optimal conditions (daytime, good visibility) and then in degraded conditions (night, sunrise, adverse weather) to challenge robustness.

Two signals were monitored in Simulink scopes to objectively validate PER outputs:

- FRAME_ASPECT: indicates the detected signal state transmitted to PER.
- M_STABLE: binary flag indicating stability of the detection over the sliding window.

Example observations for the nominal scenario 1:

- M_STABLE scope: four distinct peaks at 1 corresponding to stable detection for each of the four signals.

Figure 61: M_STABLE scope for scenario 1

It can be observed that some peaks are wider than others. This variation is explained by the speed at which the train passes the signal: the faster the train moves in front of the signal, the shorter and narrower the corresponding detection peak becomes.

- FRAME_ASPECT scope: begins Empty (no detection requested), transitions to SIG_DOWN while the sliding window fills and then stabilizes to “voie libre” once enough frames are aggregated.

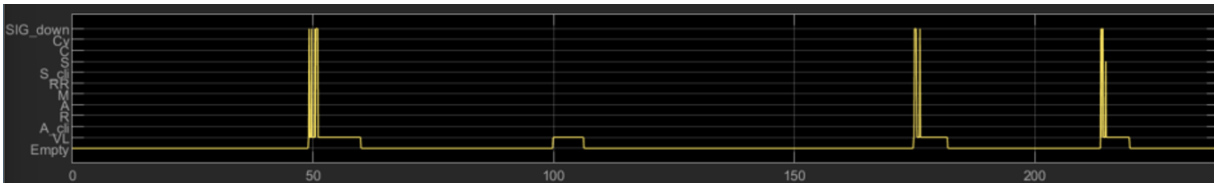


Figure 62: Signal state sent to SCV for scenario 1

These traces validate the nominal PER behavior for reference scenarios.

Quantitative analysis and key indicators

ID	Weather	Track	Signal state n°1	Signal state n°2	Signal state n°3	Signal state n°4	Chassis type	Lamp positions	Rail segmentation	Signal state sent by SD
S01	Sun	4	VL	VL	VL	VL	OK (4/4)	OK	OK	OK (4/4)
S02	Sun	52	A.	C	R	RR	OK (4/4)	OK	OK	OK (4/4)
S03	Sun	4	C	A.	A.	C	OK (4/4)	OK	OK	OK (4/4)
S04	Night	52	VL	VL	VL	VL	ERROR (3/4)	OK	OK	ERROR (3/4)
S05	Night	4	A.	C	R	RR	ERROR (3/4)	OK	OK	ERROR (2/4)
S06	Sun Rise	52	VL	VL	VL	VL	ERROR (3/4)	OK	OK	ERROR (3/4)
S07	Sun Rise	4	A.	C	R	RR	ERROR (2/4)	OK	OK	ERROR (2/4)

Table 9: PER test results for each scenario

All three models (chassis detection, lamp detection, rail segmentation) are correctly validated in the three nominal scenarios. Most errors originate from the chassis detection model. Lamp detection and segmentation remain stable.

In optimal conditions, PER transmits correct and stable outputs for all nominal scenarios. In degraded conditions, the PER output is not stable, mainly due to chassis detection dropouts.

Figure 63: Example of a frame (from degraded-condition scenario) for which the chassis detection failed

Detection quality on single real frame

On real-world data, the models achieve overall very satisfactory performance. Based on the metrics defined in the previous section, representative results indicate the probability of obtaining a correct detection in a single image is approximately:

- P(chassis) ≈ 96%

- P(lamp) ≈ 97%
- P(segmentation) ≈ 99%

Here, we make the assumptions that, thanks to rail segmentation, the correct chassis is selected in 99% of cases and at most two chassis are present in an image and that each chassis has at most two lamps lit, and detections are treated as independent events

This is an approximation because the independence assumption may not hold in practice, errors are often correlated, due to lighting or occlusion.

These assumptions are made to obtain a relative performance score. This score may be challenged, as detection probabilities vary significantly with the environment. Under optimal conditions, performance can approach 100%, whereas in degraded conditions such as rain, fog, and nighttime our detection algorithms could become largely ineffective at this stage of development.

For a single image, the probability that both chassis are correctly detected is:

- P(chassis_total) = $0{,}96^2 = 0{,}9216$ ↔ Probability both chassis detections are correct
- P(lampe_total) = $0{,}97^2 = 0{,}9409$ ↔ Probability both lamp detections are correct
- P(final) = $0{,}99 \times 0{,}9216 \times 0{,}9409 \approx 0{,}8585 \approx 86\ \%$ ↔ Combined probability that segmentation, chassis and lamps are all correct

Finaly, there is approximately **86 %** chance per single image to correctly detect the full signal state. This value is close to the **90%** detection-quality objective and improves when sequential filtering (70-frame sliding window) is applied because temporal aggregation reduces sporadic errors.

Robustness

Under optimal conditions, **100% of the defined scenarios are validated** at the SD module level. Thus, the robustness objective is achieved.

Under degraded conditions, failures are concentrated on the chassis detection model.

Performance (FPS)

The simulation currently runs at **6.5 FPS**.

Implementing GPU acceleration would, in theory, allow it to easily exceed 10 FPS, thus meeting the speed objective. There is more argument in Section IV: 1. Inference time via CPU profiling results.

Memory efficiency

The goal was to keep the model sizes **under 50 MB**.

This objective has been met for both the segmentation and detection models, for example the segmentation model sizes are just 11 MB and it's approximately the same for the detection models.

Conclusion:

Overall, the 3D Environment → PER → SCV perception chain has been validated end-to-end. The three vision models: rail segmentation, chassis detection, and lamp detection, show strong performance on real data (rail segmentation IoU ≈ 99%, chassis detection ≈ 96%, lamp detection ≈ 97%) while meeting the model size constraint (< 50 MB). The main residual weakness is the degradation of chassis detection on simulated data under challenging conditions (nighttime and adverse weather). The current CPU-based execution achieves approximately 6.5 FPS, migration to a GPU-accelerated runtime is required to reach the 10 FPS target.

Finally, 3D simulation proved essential for generating controlled and repeatable test scenarios, and for establishing a structured validation framework that offers a promising basis for future certification activities.

Table 10 presents the KPI summary:

KPI	Target	Current measurement	Comment
Detection score per image (on real data)	≥ 90%	≈ 86%	**Almost OK** (More data on degraded conditions are required)
Robustness under optimal conditions (on simulation)	100%	100% (SD module level)	**OK**
End-to-end FPS (on simulation)	≥ 10	≈ 6.5 (CPU)	**Almost OK** (GPU acceleration is required)
Memory efficiency	< 50 MB	≈ 11 MB	**OK**
Degraded scenarios (on simulation)	No performance deterioration	Errors concentrated on chassis detection	**Failed** (Requires new training with targeted data / augmentation)

Table 10 : KPI compliance

4.2.3.4 Limits and opportunities

This subsection summarizes the key limitations identified during validation and the main opportunities enabled by the 3D modeling approach.

Inference time via CPU profiling results: need for GPU acceleration

During our tests, we observed that the simulation stuttered when entering a detection zone, even after optimizing the vision models to reduce inference time. While this latency does not affect the functioning of the model itself, it is disruptive to the user experience of the ATO demonstrator. To address this, we explored ways to accelerate the simulation, notably by leveraging GPU computation for elements that slow down the simulation. In order to identify computational bottlenecks, a full profiling campaign using the Simulink profiler on the ATO simulation produced the following measurements:

- **Total simulation duration:** 214 s (~3 min 34 s).
- **Most time-consuming blocks:**
  - 3D Environment (video stream and signal-state management).
  - PER image processing and inference.

Figure 64: Simulink profiler for the simulation (entire GoA4 model)

It is important to note that although the diagram shows that PER is not the most time consuming module, it is in fact the most computationally expensive when it is executed. Indeed, the PER block is only called 25% of the time, whereas the 3D Environment block runs continuously and operates 100% of the time.

A detailed profiling of the PER blocks:

- **Total iterations:** 328 (4 detection requests ≈ 82 frames per request).
- **Total PER compute time:** 52 s for all frames.
- **Average time per image:** 0.1585 s → **≈ 6.5 FPS**.

Two major observations emerge from the analysis,

- **Effective optimization:** Recent improvements to the models and detection strategy have roughly tripled the inference speed, representing an increase of about 200%.
- **Hardware limitation:** To surpass the 10 FPS threshold without compromising detection quality, implementing hardware acceleration via GPU appears essential.

Mode	Speed CPU (ms)	Speed GPU (ms)	Approximate gain
Yolov11-n	56 ms	1.5 ms	*37
Yolov11-s	90 ms	2.5 ms	*36

Table 11: Estimated GPU acceleration, based on Ultralytics benchmarks (640×640 images, NVIDIA T4 GPU)

The expected acceleration with a specific GPU is given on Table 11. With a more modest GPU and within a Simulink execution environment, a conservative and realistic speed-up of around ×10 can be expected for the models developed in this work.

Video stream quality

As previously noted, the video stream has been stabilized and corrected using an automatic brightness adjustment integrated in Simulink.While this solution makes the stream usable, it does not fully compensate for the original degradation, particularly under nighttime conditions.The aim moving forward is to obtain a native, unaltered stream directly from Unreal Engine through optimization.

Synthetic dataset: simulated annotated database

The objective is to create a fully annotated simulated dataset to perform quantitative evaluations in virtual conditions comparable to real-data tests.

A synthetic database provides:

- annotated images for chassis detection and rail segmentation,
- annotated images for lamp detection,
- controlled variations across lighting, weather, occlusion and camera viewpoints.

This dataset will enable systematic, repeatable experiments and facilitate domain-comparison between simulated and real data.

Insights on Simulation Performance and Trade-offs

Generalization

Models trained on real data generalize well to the simulated environment. In nominal conditions, performance equals or exceeds real-data results because synthetic images are less noisy and more uniform.

For challenging conditions, such as backlighting, heavy rain or night, the realism of synthetic data must be improved to preserve equivalent performance.

The use of a 3D model to test a perception algorithm intended for railway applications is therefore a necessary step that could eventually be integrated into an AI algorithm certification process. This step is crucial, as it offers significant reductions in human effort and cost, and mitigates the risks associated with real-world testing. However, it also raises several questions, most notably the representativeness of the 3D model. How closely should the simulation environment match reality?

To what level of detail is it justified to refine it, and at what point does additional complexity become unnecessary over-quality? Ultimately, how do we determine the right balance?

Operational advantages of 3D modelling

3D simulation provides several strategic and operational advantages. 3D simulation is a powerful, controlled platform for developing and validating computer-vision systems. Below is a compact, slightly more detailed summary of its main advantages and practical opportunities.

- Full control & reproducibility: Every scene parameter (lighting, material properties, camera pose, object trajectories, sensor noise) can be set precisely and replayed identically.
- Rare-case and corner-case coverage: You can generate low-probability but high-impact events on demand, extreme weather, sensor failures, or unusual object arrangements, that are costly or dangerous to capture in real life.
- Privacy-friendly data generation: Synthetic scenes contain no real people or sensitive information, simplifying compliance with privacy laws and removing consent/annotation complications.
- Systematic parameter analysis: By varying a single factor at a time (illumination, occlusion, motion blur), you can quantify its direct effect on model performance, something nearly impossible to do reliably with uncontrolled real-world data. This controlled environment also allows the identification of potential failure modes (FMEA) and supports a roadmap for understanding system limits and prioritizing future safety or robustness improvements.
- Cost reduction and improved safety: Virtual testing reduces field campaigns, human supervision, and associated risks. Early failures can be detected and fixed in simulation before any real-world deployment.

Additional opportunities for computer vision had been identified beyond this project

- Automatic annotation: Simulation provides exact labels (bounding boxes, segmentation masks, depth, semantic IDs) for every frame at no manual cost accelerating dataset creation and enabling pixel-perfect benchmarks.
- Controlled data augmentation: Easily create targeted synthetic samples that fill gaps in real datasets (e.g., rare sign types, edge lighting, reflective surfaces) to improve generalization.
- Reinforcement learning (RL): Agents can learn behaviors through trial and error safely in simulation, including long training episodes that would be impractical or unsafe in the real world.
- Safety and extreme-case testing: Repeatable, auditable scenarios allow validation under hazardous conditions (night glare, near misses, sensor blackout) and help establish failure modes.

- Certification support: Standardized, versioned simulation protocols produce auditable test suites that can support verification and regulatory certification of AI components in future.

Limitations and risks

- Simulation to real environment gap: high performance in simulation does not guarantee equivalent real-world performance, due to the discrepancies between the 3D environment and the real world as discussed in Section II. “Development and unit validation of computer-vision models”, fine-tuning on real data remains necessary.
- Approximate physics: some material and optical effects, such as subsurface scattering, micro-roughness, complex translucency, are hard to reproduce in simulation.
- Rendering cost: photorealistic rendering requires significant GPU/CPU resources.
- Diminishing returns: improving visual fidelity becomes increasingly costly, while the performance gains for the algorithm become progressively smaller.
- Mandatory real validation: calibration and validation on field data are essential before operational deployment.
- Representiveness: Simulation is not a silver bullet, its value depends on the fidelity and representativeness of the virtual environment. High fidelity improves transfer to real data but raises cost and complexity, low fidelity is cheaper but risks unrealistic results.

4.2.4 Signal Converter (SCV) module validation

4.2.4.1 SCV module architecture

Signal Conversion software shares data with two other software components: PER and ADM. PER provides signaling data acquired from the ground through computer vision. ADM expects speed and access information about upcoming sectors. The function of the Signal Converter in between is to process signaling data perceived from the ground and convert it to a Movement Authority. This authorization will be used by ADM to compute train control commands.

Initially, many legacy functions have been implemented in SCV software component. The main all-in-one functions imported are Safe_Deceleration and Supervision_Limit_Monitoring.

- The Safe_Deceleration function features the calculation of A_brake_emergency (model conversion), calculation of integrated factor Kv and Kr, EBD curves (emergency brake decelerations) and A_Gradient.
- Supervision_Limit_Monitoring features the calculation of Deceleration Grid, the MRSP (most restrictive speed profile) and supervision limits (emergency/service brake deceleration and intervention).

However, among the values returned by the SCV software block, only a few are used by the next block in the logical chain, ADM. Therefore, only useful data are being reviewed in the validation procedure. This validation process focuses on internal blocks that are indeed used by ADM.

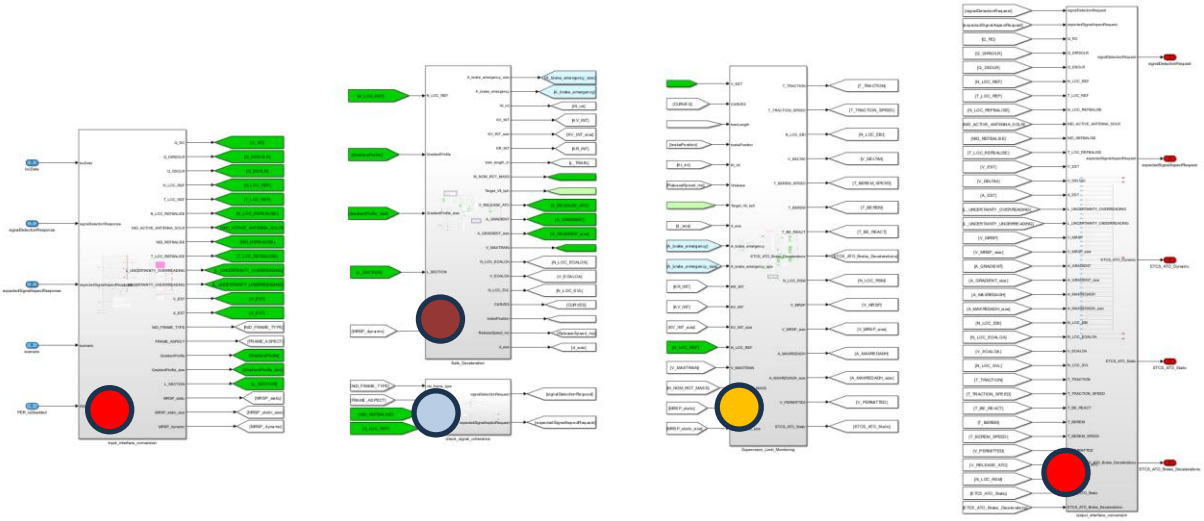


Figure 65: Simulink model of SCV subsystem

Legend:
Interface function blocks for input and output conversion and datatyping
Check_signal_coherence function block
Safe_Deceleration function block
Supervision_limit_monitoring function block

The validation of interface PER-SCV will focus on the input interface subfunction, while validation of SCV-ADM will focus on the Safe_Deceleration subfunction.

4.2.4.2 Validation of interface PER-SCV

The aim is to confirm that information transmitted through light signals is correctly converted to a Movement Authority or MA. A MA indicates the permission for a train to move from one point to another according to the characteristics of the infrastructure and the occupation on the sector: line peculiarities, signaling, slopes, speed limitations, position of the downstream trains, etc. However, light signals work in a sequential way and general operation can not be tested separately. Therefore, several scenarios involving various light signals have been created to reproduce typical situations. The tests involve all light signals that the train may interact with in its environment.

Each time a light signal is acquired by PER and light pattern is detected, various information are provided to SCV. Depending on the signal pattern acquired, the location of the End of Authority is set. End Of Authority Location up to which a train or a shunting composition is authorized to proceed. More occasionally, alterations of the speed limit might also happen, such as sequences “30 announcement > 30 reminder” symbolized as R-RR, or the “60 announcement > 60 reminder”, symbolized as the blinking variant (R)-(RR).

In the end, the two main information returned are the updated End of Authority location, and MRSP speed profile affected by event-related changes (switch crossing etc).

Validation of this process will focus on how MA evolves, based on interactions with signaling along the way.

Validation of SCV also allows to test Repository (REP) software component. This component theoretically contains all the various End of Authority locations and MRSP profiles for each sector, depending on the signaling encountered. REP was initially intended as an individual component. But software developments led to the integration of some REP data into SCV. The validation of SCV processed below allows the analysis of REP stored data at the same time.

- Scenario A-C-A

The first test scenario simulates encounter with a full stop, known as C signal.

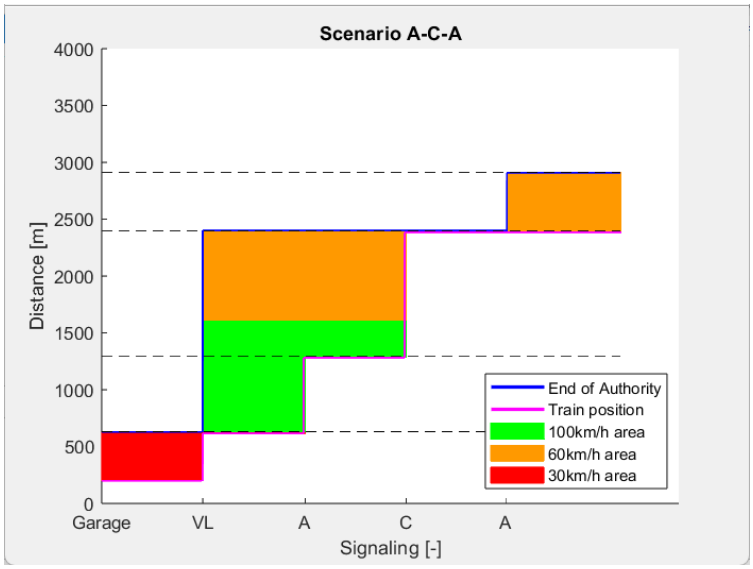


Figure 66: information related to Movement Authority, in scenario A-C-A

1. Reading of VL signal provides 2 information
    a. Train can access track over 2 blocks ahead safely
    b. Extension of Movement Authority reveals that a speed limitation is going to occur after 1600m, from 100km/h to 60km/h
2. Reading of A signal provides 2 information
    a. Movement Authority is only extended of one length, meaning that the next signal could be a track free at best, a warning A or a full stop C. This imminent end of Authority means that the driver must anticipate its deceleration in order to stop before this milestone
    b. MRSP profile knowledge is not extended so no change in speed limitations knowledge
3. Reading of C signal provides 2 information
    a. No access is authorized ahead of the full stop C signal. If the train does cross the signal despite the C signal, onboard train protection would activate and trigger an emergency braking
    b. Although current C signal forces the train to stop, current speed limitation is still 60km/h. The train will be allowed to run at that speed as soon as Movement Authority is extended, based on a different signal pattern.
4. Reading of A signal provides 2 information
    a. While stopped, ATO reads a warning signal, which extends access to one block ahead of the current signal. Movement Authority now tops at 2900m.

b. ATO can now reach 60km/h again on the next sector. Extension of Movement Authority provides no speed change occurring in the upcoming sector.

- Scenario R-RR

The second scenario simulates occurrence of a local speed limitation, down to 30km/h, related to the protection of an upcoming switch. In this scenario, all MA are limited to 3500m, regardless of the signal pattern. This can be explained by the existence of an end-of-track buffer stop.

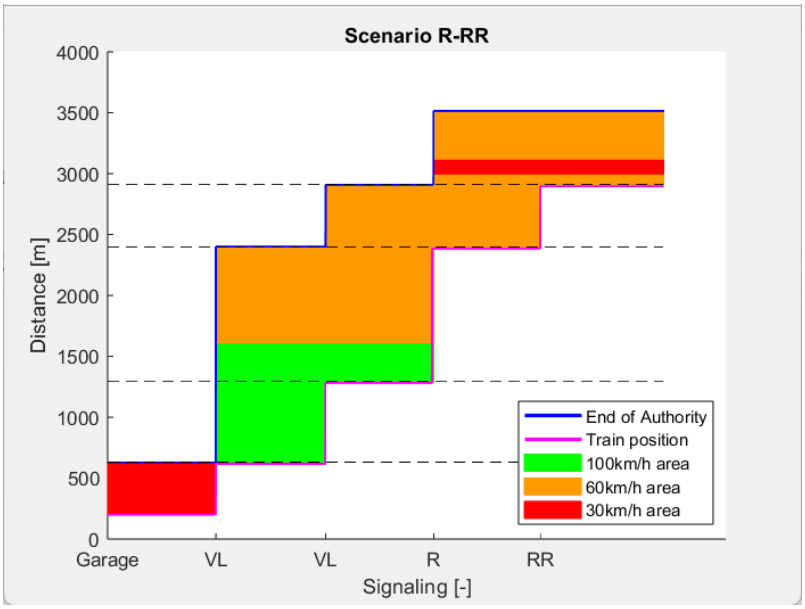


Figure 67: Information on Movement Authority in scenario R-RR

1. Reading of VL signal provides 2 information
   a. Train can access track over 2 blocks ahead safely
   b. Extension of Movement Authority reveals that a speed limitation is going to occur after 1600m, from 100km/h to 60km/h
2. Reading of VL signal provides 2 information
   a. Train can access track over 2 blocks ahead safely, up to 2800m
   b. Extension of Movement Authority reveals no upcoming change in the speed profile
3. Reading of R signal provides 2 information
   a. Train can access track over 2 blocks ahead safely
   b. The direct meaning of this signal is “upcoming 30km/h speed limitation”, meaning that a strong speed limitation is about to occur in the next sector. Indeed, speed profile of the new sector revealed does include a short-term 30km/h portion in the middle of a 60 speed zone. This 30 slow zone is related to a switch being encountered on the way
4. Reading of RR signal provides 2 information
   a. The meaning of RR signal is “30km/h reminder”, it confirms the previous signal “30km/h slow zone notification”. A slow zone is indeed present 100 meters ahead. This slow zone is limited to the switch length, before going back to 60km/h limitation. The slow zone is only applied if the train is taking the secondary branch of the switch.

b. Movement Authority is still limited to the imminent sector only, up to 3500m from the start. This is related to the presence of the buffer stop and the end of the track, regardless of the signal acquired.

Conclusion:

Through the two scenarios described above, various signal patterns have been acquired and submitted to SCV for the creation of Movement Authority. After review, returned values for End of Authority locations and speed profiles behave as expected.

4.2.4.3 Validation of interface SCV-ADM

The aim is to validate the conversion model implemented in the Simulink model of SCV.

In order to validate the current Conversion Model, a theoretical model has to be used. The European Union Agency for Railways (ERA) has developed a tool, called braking curves, which allows predetermining all the braking distances. Namely, it does provide a tab returning brake parameters and related deceleration as a function of speed:

	Brake percentage λ (%)
Brake percentage for emergency brake λ_{eb} (%)	90
Brake percentage for service brake λ_{sb} (%)	90
V_lim for emergency brake (km/h)	115,62
V_lim for service brake (km/h)	115,62
kto	1,20
T_brake_emergency_cm0 (seconds)	5,0200
T_brake_emergency_cmt (seconds)	6,0240
T_brake_service_cm0 (seconds)	14,0800
T_brake_service_cmt (seconds)	16,8960
T_brake_emergency (seconds)	6,024
T_brake_service (seconds) ◉ Conversion Model ○ User's shorter value	16,896
T_brake_emergency_react (seconds)	2,99
T_brake_service_react (seconds)	2,07

Figure 68: Extract from ERA braking tool, displaying user interface for brake percentage and the resulting parameters

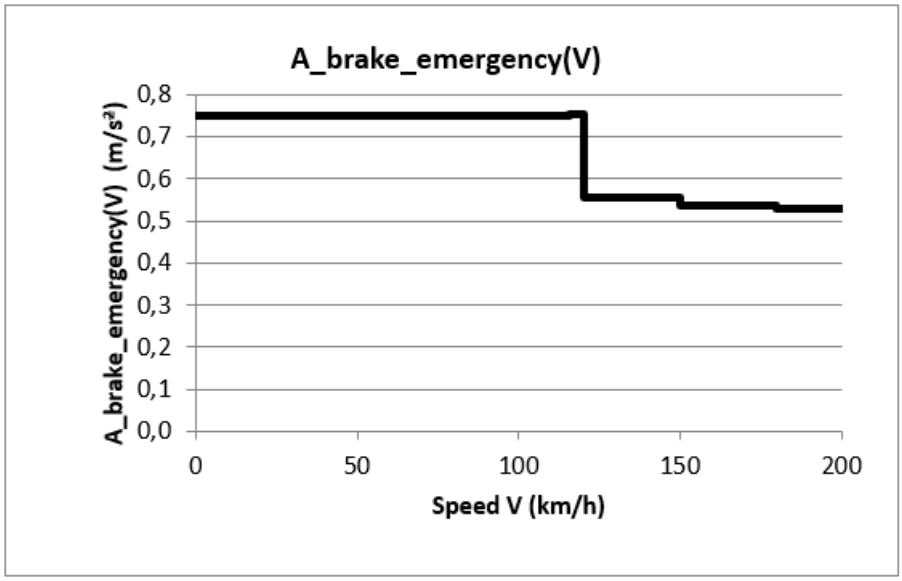


Figure 69: Results from ERA braking tool, presenting A_brake_emergency(V)

These computed values will be set as reference values for the conversion model. Comparison will be established between reference values and model values, in order to validate simulation results.

Model Conversion, implemented in simulation model, is described in document Subset 026 §A.3.7 “Calculation of the basic deceleration”. It is built using a set of speed thresholds and polynomials gathering specific coefficients:

A.3.7.5 The following steps of the basic deceleration shall be calculated by means of a set of polynomials of the third order with the following format:

$AD\_n = a3\_n * \lambda_o^3 + a2\_n * \lambda_o^2 + a1\_n * \lambda_o + a0\_n$

Figure 70: Extract from subset 026 §A.3.7, presenting basic deceleration calculation

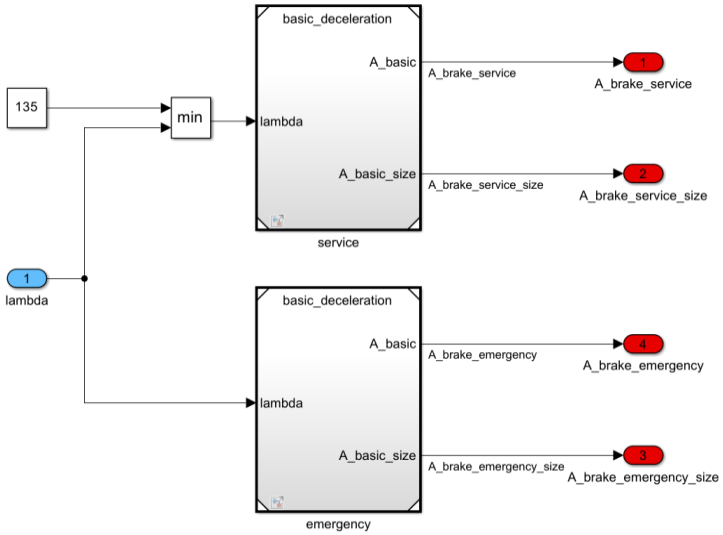


Figure 71: Model conversion from subset 026 §A.3.7 implemented in SCV Simulink model, for both service and emergency braking

Figure 71 illustrates the implementation of Model Conversion in Simulink, for both service and emergency braking.

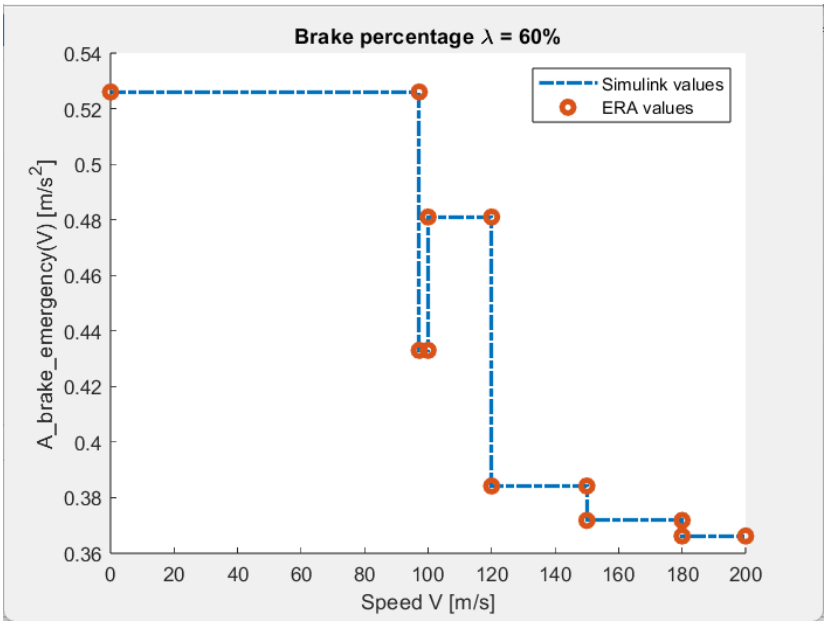


Figure 72: A_brake_emergency step values for λ = 60%, from Simulink and ERA sources

On Figure 72 presenting A_brake_emergency(V), established for a brake percentage of 60%, values obtained through simulation are very close to reference values from ERA tool. The step changes are very similar between the two sources.

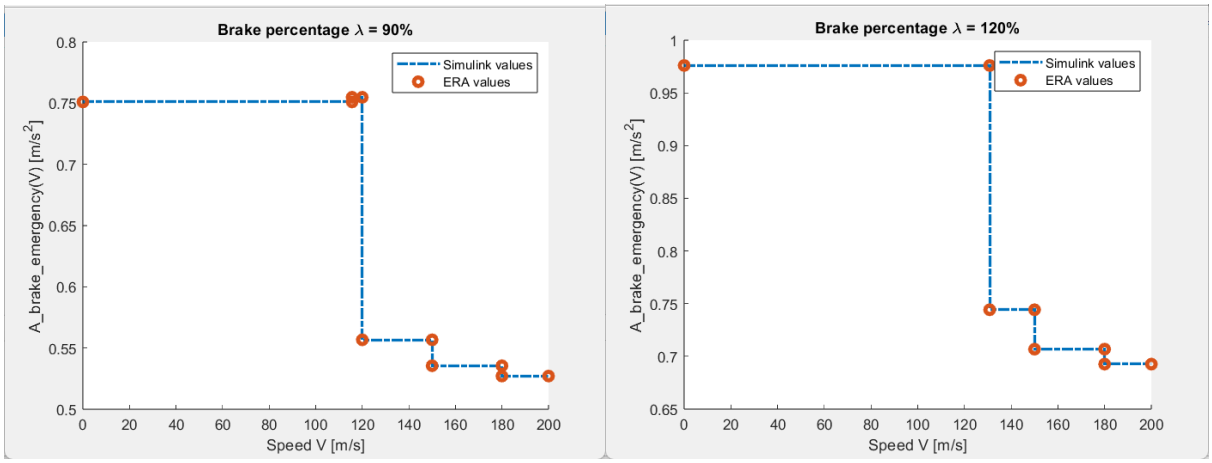


Figure 73: A_brake_emergency step values for λ = 90% and λ = 120%, from Simulink and ERA sources

The same observation can be made for lambda of 90% and 120%. Reference values from ERA do match step values from simulation model.

As a conclusion, conversion model implemented in simulation model returns values which are consistent with reference values from ERA.

4.2.5 Automatic Driving Module (ADM) validation

4.2.5.1 ADM module architecture

The ADM (Automatic Drive Model) follows the architecture described in Section 2.2.3. Its interfaces are structured as follows:

Inputs:

- SCV (Signaling Control System): Provides movement authority data in Subset-130 format.
- REP (Route Planning System): Supplies Journey Plans (JP) and Speed Profiles (SP). *Note: REP communications are not modeled; JP/SP selection is handled internally by the ADM module.*

Outputs:

- Train Model: Generates a control command value, applied exclusively when ATO GoA4 is active.

The ADM is decomposed into submodules, detailed in Table 12. The table summarizes their functions and validation status (based on the GoA2 model):

Submodule	Description	Validation already done for the GoA2 model
Profile recalculation logic	Verifies recalculation conditions for TTSM, ATSM, SSEM, and OSP profiles.	NO
TTSM profile calculation	Computes TTSM profile	YES
ATSM profile calculation	Computes ATSM profile	YES
SSEM profile calculation	Computes SSEM profile	NO
OSP profile calculation	Merges TTSM, ATSM and SSEM prfiles to construct the OSP.	NO
MPC controller	Realizes OSP profile tracking through predictive control	YES

Table 12: ADM submodules

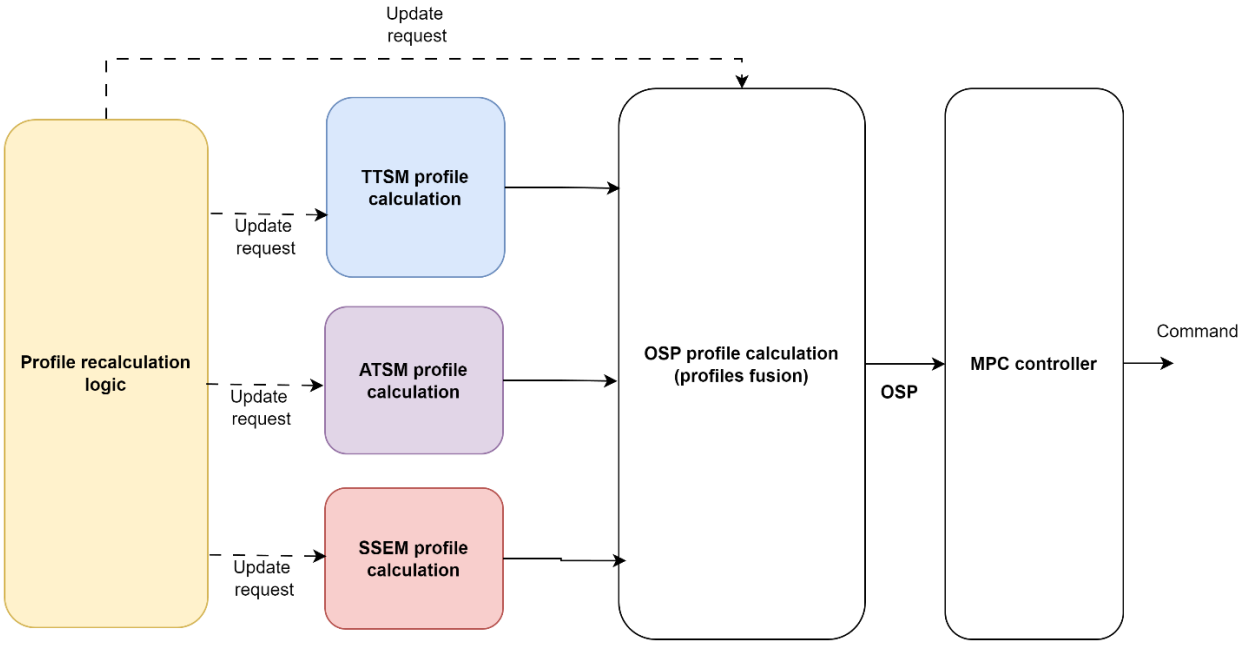


Figure 74: ADM architecture

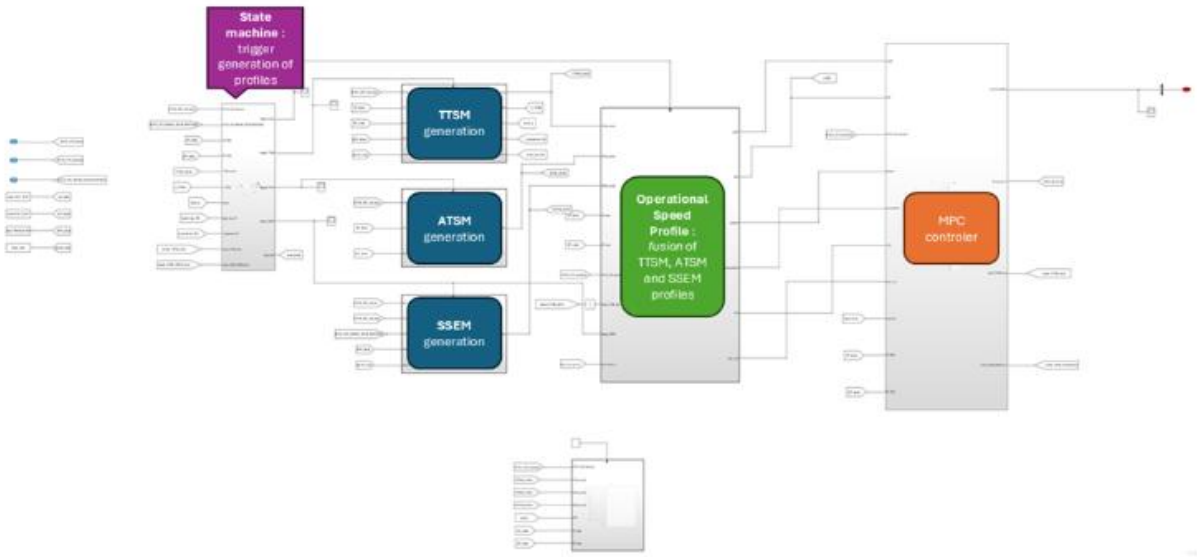


Figure 75: ADM subsystem on Simulink

Since certain components were already validated in the GoA2 model, the current validation focuses on the following modules:

- SSEM Profile Construction
- Profile Fusion (construction of the OSP)
- Profile Recalculation Logic

4.2.5.2 Validation of SSEM Profile Construction

SSEM Calculation Principle

According to Subset-125, the SSEM is defined as *"the maximum speed the train can run avoiding ETCS intervention."* For this project, the SSEM has been implemented as the ETCS intervention curve.

However, Subset-126 specifies that ETCS intervention occurs at a single point called EBI (Emergency Brake Intervention), which represents the intervention point *at the current speed*. To construct a continuous EBI curve, the EBI calculation methodology has been extended to compute intervention points *at each speed value*.

At any given position, the SSEM is defined as the minimum of the following components:

1. **EBI related to speed limits:**

   Corresponds to the MRSP plus the dV_ebi margin (overspeed tolerance before intervention).

2. **EBI related to braking targets:**

   Derived from EBD (Emergency Brake Deceleration) curves calculated for braking targets (EOA, LOA, MRSP). For each target:

   - Compute the EBD curve using Subset-130 data.
   - For a sampled set of speeds, determine the corresponding EBI point (as per Subset-026 methodology).

3. **Traction Curves:**

   Generated at each speed limit increase to ensure SSEM continuity when speed limits is rising.

Note: the release speed has not be considered in the SSEM computation, resulting in a deviation from the Subset 125. The reason for this assumption is that adding the release speed would result in a discontinuity of the SSEM profile at the RSM (Release Speed Monitoring) start location.

Validation Scope

Two critical steps of the SSEM calculation will be validated:

1. **EBD and EBI Curve Calculation** for braking targets.
2. **SSEM Profile Construction** from EBI and traction curves.

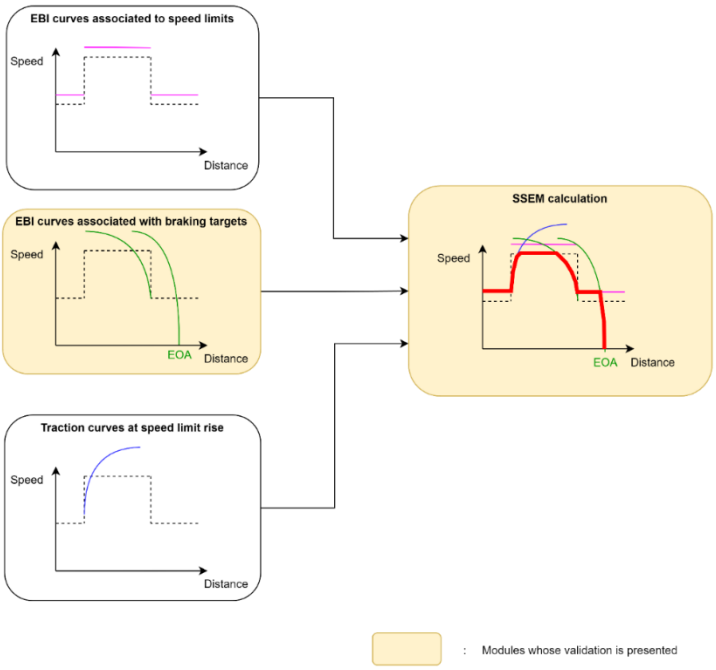


Figure 76: SSEM calculation

Validation of EBD and EBI Calculations

Since real ETCS data for EBD and EBI calculations is unavailable, the model is validated by comparison with a trusted reference model: the ERA Braking Curves Tool.

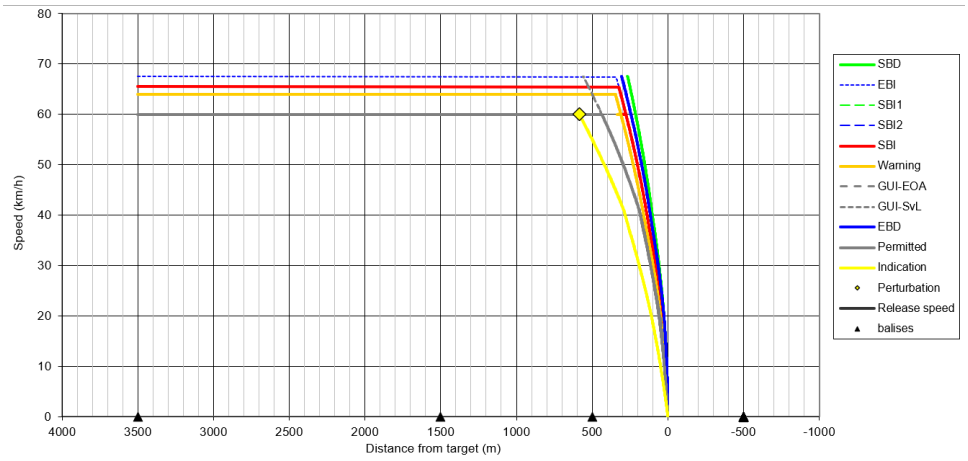


Figure 77: Visualization of deceleration curves on ERA Braking Curves Tool

Ensuring iso-parameter conditions with the ERA tool presents a key challenge due to differing input data sources: SSEM Model uses ATO-OB data transmitted by ETCS via Subset-130, while ERA Braking Curves Tool uses EVC (European Vital Computer) data mentioned in Subset-026. As a consequence, special attention was paid to data consistency between these sources. For example: the ATO-OB directly receives the A_GRADIENT table (acceleration/deceleration values due to track gradient), while the ERA tool requires raw gradient values as input.

The comparison was conducted for two target types:EOA (End of Authority) and MRSP (Most Restrictive Speed Profile)

Figure 78 shows the EBD and EBI curves generated by both the ERA Braking Curves Tool and the ADM model, along with the absolute and relative errors between them.

The error remains below 1%, confirming the validity of the ADM model.

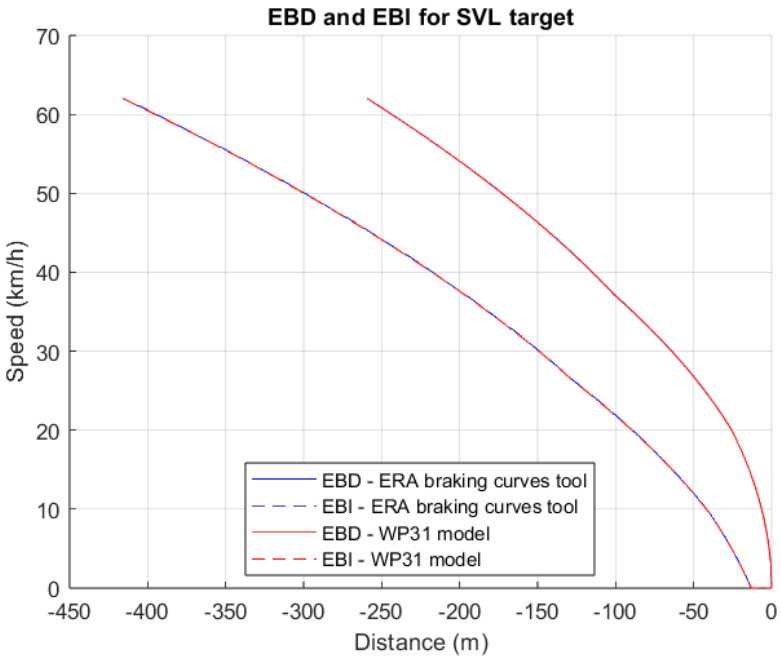


Figure 78: Comparison of EBD and EBI curves computed by WP31 model and ERA Braking Curves Tool for a SvL target

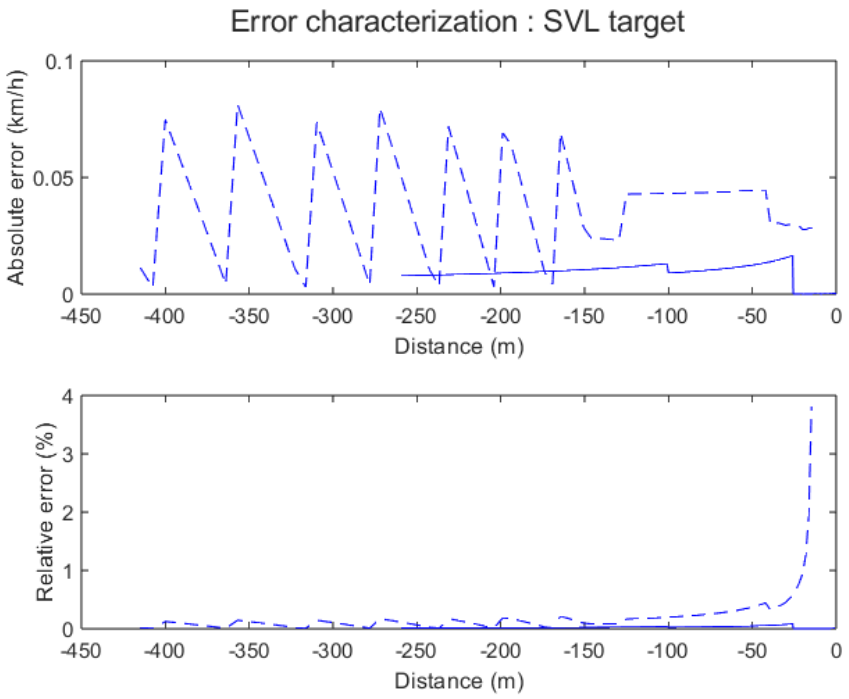


Figure 79: Error between EBD (solid line) and EBI (dashed line) computed by WP31 model and ERA Braking Curves tool for a SvL target

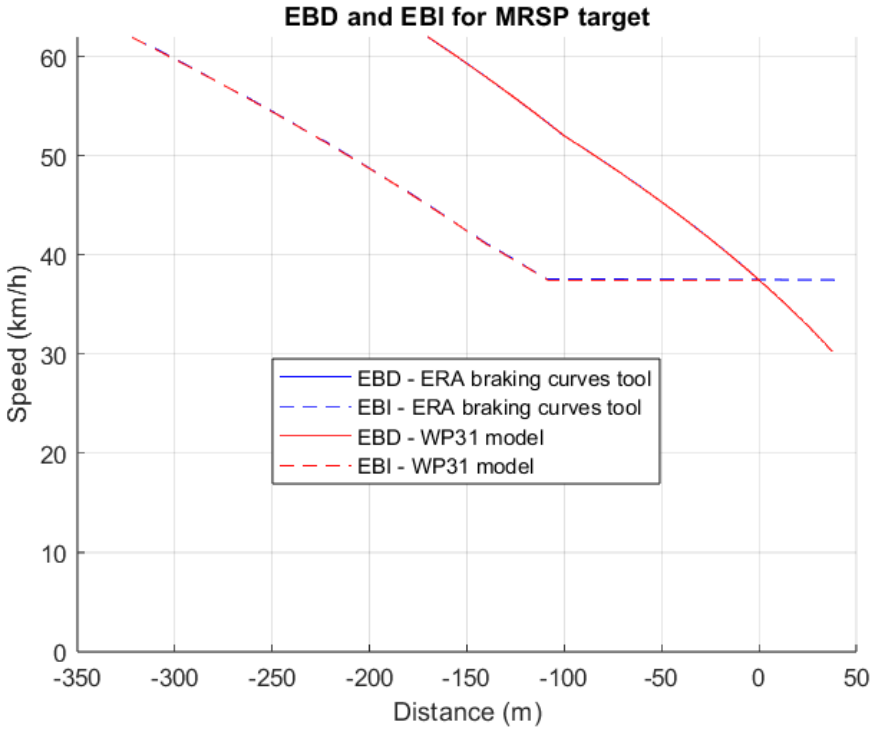


Figure 80: Comparison of EBD and EBI curves computed by WP31 model and ERA Braking Curves Tool for a MRSP target

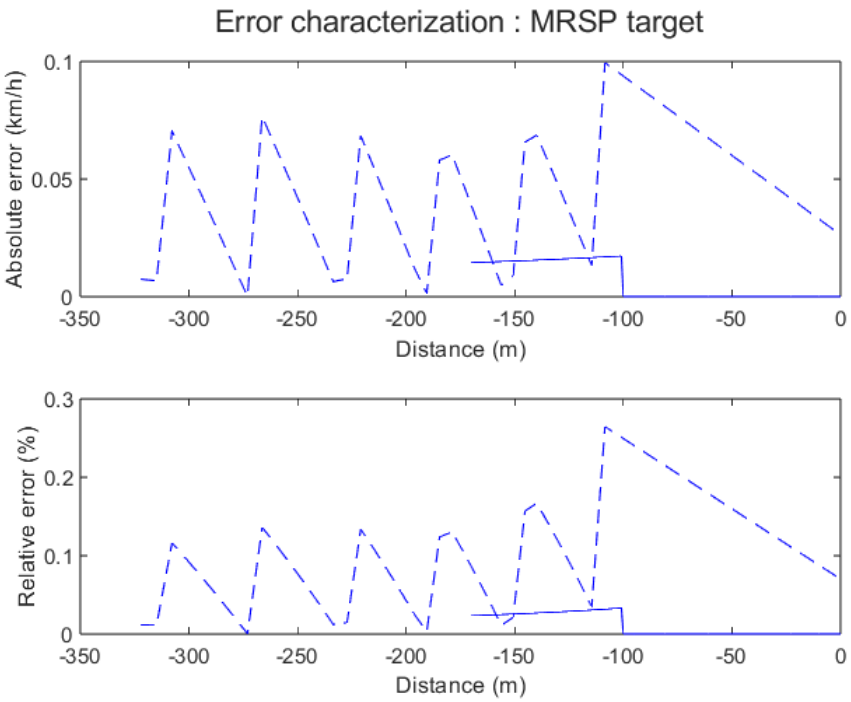


Figure 81: Error between EBD (solid line) and EBI (dashed line) computed by WP31 model and ERA Braking Curves tool for a MRSP target

Validation of SSEM Construction

As illustrated in Figure 76, the SSEM profile is constructed by sampling positions and selecting the minimum value among all EBI curves and traction curves defined at each point.

The implementation is validated graphically by overlaying on a single plot input curves (EBI and traction) and resulting SSEM profile

Figure 82 demonstrates that the SSEM correctly represents the pointwise minimum of the plotted curves across all positions.

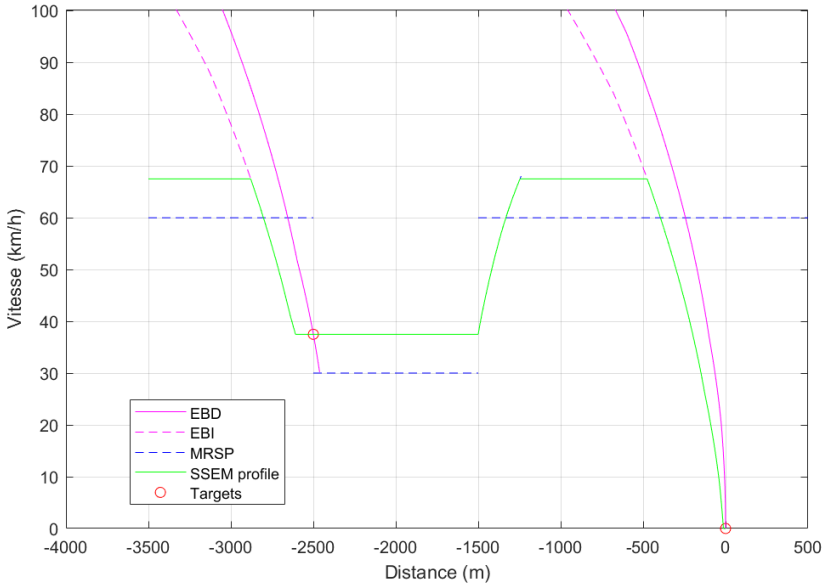


Figure 82: SSEM profile construction

4.2.5.3 Validation of profile fusion (construction of the OSP)

The TTSM and ATSM profiles are computed using the same methodology as in the GoA2 model.

Once the three profiles (TTSM, ATSM, and SSEM) are generated, they must be merged into a single profile, the OSP (Operational Speed Profile), as defined in Subset-125.

The OSP is constructed by selecting, at each position, the minimum speed value among the three profiles: TTSM, ATSM and SSEM

Note : Prior to fusion, the three profiles are resampled to a common positional grid to ensure consistency in OSP calculation.

7.1.1.2 **Note:** The following figure shows an overview of the functional features. This diagram does not describe the architecture.

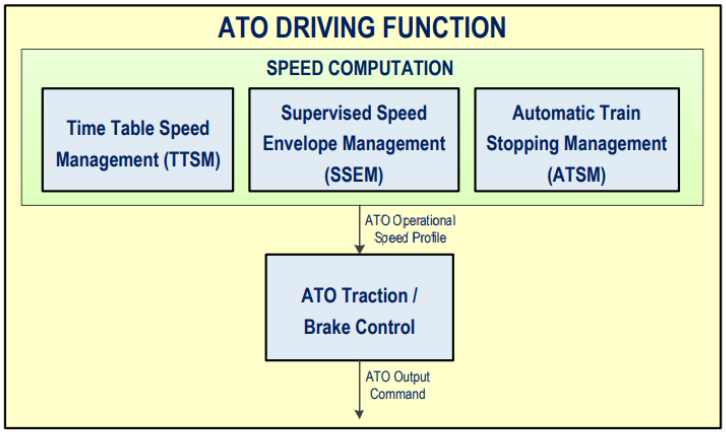


Figure 2 ATO driving function

Figure 83: ATO driving function (extract from Subset-125 7.1.1.2)

The implementation generates the following plot, which confirms that the OSP correctly represents the pointwise minimum of the three precomputed profiles (TTSM, ATSM, and SSEM).

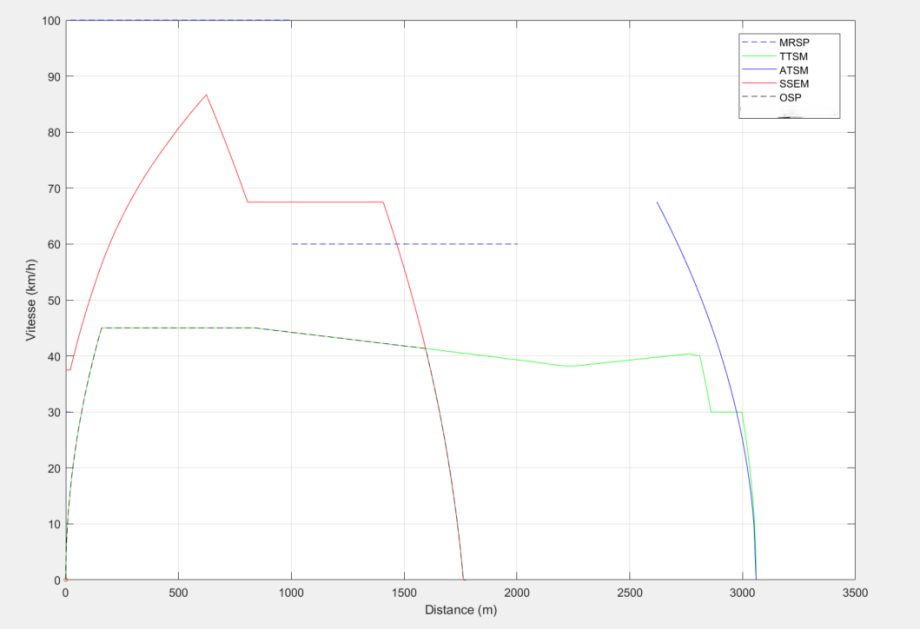


Figure 84: Operational Speed Profile (OSP) construction

The OSP is initially defined as a list of position-speed pairs *(position, speed)*. To enable tracking by the Model Predictive Controller (MPC) (described in Section 4.1.3), a time component is added. This is achieved by integrating the inverse of speed over the traveled distance, converting the profile into a position-speed-time trajectory.

4.2.5.4 Validation of profiles recalculation logic

It would not be efficient to recompute the TTSM, ATSM, SSEM, and OSP profiles at every simulation time step. Instead, these profiles are only recalculated when specific conditions are met, as listed in Table 13.

This list of conditions was built incrementally: validation tests progressively revealed additional situations that required a profile update.
Due to the scope defined for the modelling work, only a subset of these conditions has been implemented in the current model.

Submodule triggered	Conditoin id.	Condition for triggering the module	Rationale	Implemented in the model ?
TTSM calculation	C_TTSM_01	Departure from a stopping point	The TTSM profile is defined from current position to the next stopping point.	YES
	C_TTSM_02	Modification of the Journey Profile during the mission	Change in timetable constraints for traffic regulation.	NO
	C_TTSM_03	Change in train characteristics	For example, an isolated bogie.	NO
	C_TTSM_04	Excessive delay with respect to the TTSM reference trajectory (x(t)) (threshold TBD) and the currently followed profile is the TTSM	If the train is too late compared with the TTSM forecast, MPC tracking alone is not sufficient. The TTSM must be recomputed to obtain a new, more energy-efficient catching-up strategy. The second condition prevents frequent TTSM recomputations during ATSM and especially SSEM braking phases.	YES
	C_TTSM_05	Transition from ATSM or SSEM tracking back to TTSM tracking	When switching to SSEM or ATSM, any accumulated delay is "reset" (see C_OSP_02). The MPC therefore no longer accounts for this delay, and a new TTSM must be computed.	YES
ATSM calculation	C_ATSM_01	Departure from a stopping point	The ATSM profile is computed to manage the approach to the next stopping point.	YES
	C_ATSM_02	Engagement of ATO GoA4	The ATSM profile is computed to manage the approach to the next stopping point.	YES

	C_ATSM_03	Modification of the Journey Profile during the mission	Change in timetable constraints for traffic regulation.	**NO**
	C_ATSM_04	Change in train characteristics	For example, an isolated bogie.	**NO**
SSEM calculation	C_SSEM_01	Significant change in the data sent by ETCS to the ATO-OB	For example, an extension of the Movement Authority (MA).	**YES**
Profile fusion (OSP calculation)	C_OSP_01	Recalculation of any of the TTSM, ATSM or SSEM profiles	A change in any of the three profiles may impact the resulting OSP.	**YES**
	C_OSP_02	Switching to ATSM or SSEM mode	Allows recalculation of the time axis of the profile using the current train state as reference. This effectively resets any previously accumulated delay so that only stopping accuracy remains as the objective.	**YES**

Table 13: Conditions for profiles calculation

To validate the implementation of these recalculation conditions, specific scenarios are constructed in which the corresponding transitions occur. This requires coupling the ADM with a train dynamics model, so that the evolution of the train state can trigger the recalculation events.

As an example, consider the extension of the Movement Authority (MA). When the train passes a signal displaying a clear aspect, the MA is extended by one section. As a consequence, the input data sent by ETCS-OB to ATO-OB are updated. According to the recalculation conditions table, this should trigger:

- a recalculation of the SSEM (condition C_SSEM_01), followed by
- a recalculation of the OSP (condition C_OSP_01).

This behavior is confirmed in Figure 85, which all profiles (TTSM, ATSM, SSEM, and OSP) before and after the MA extension.

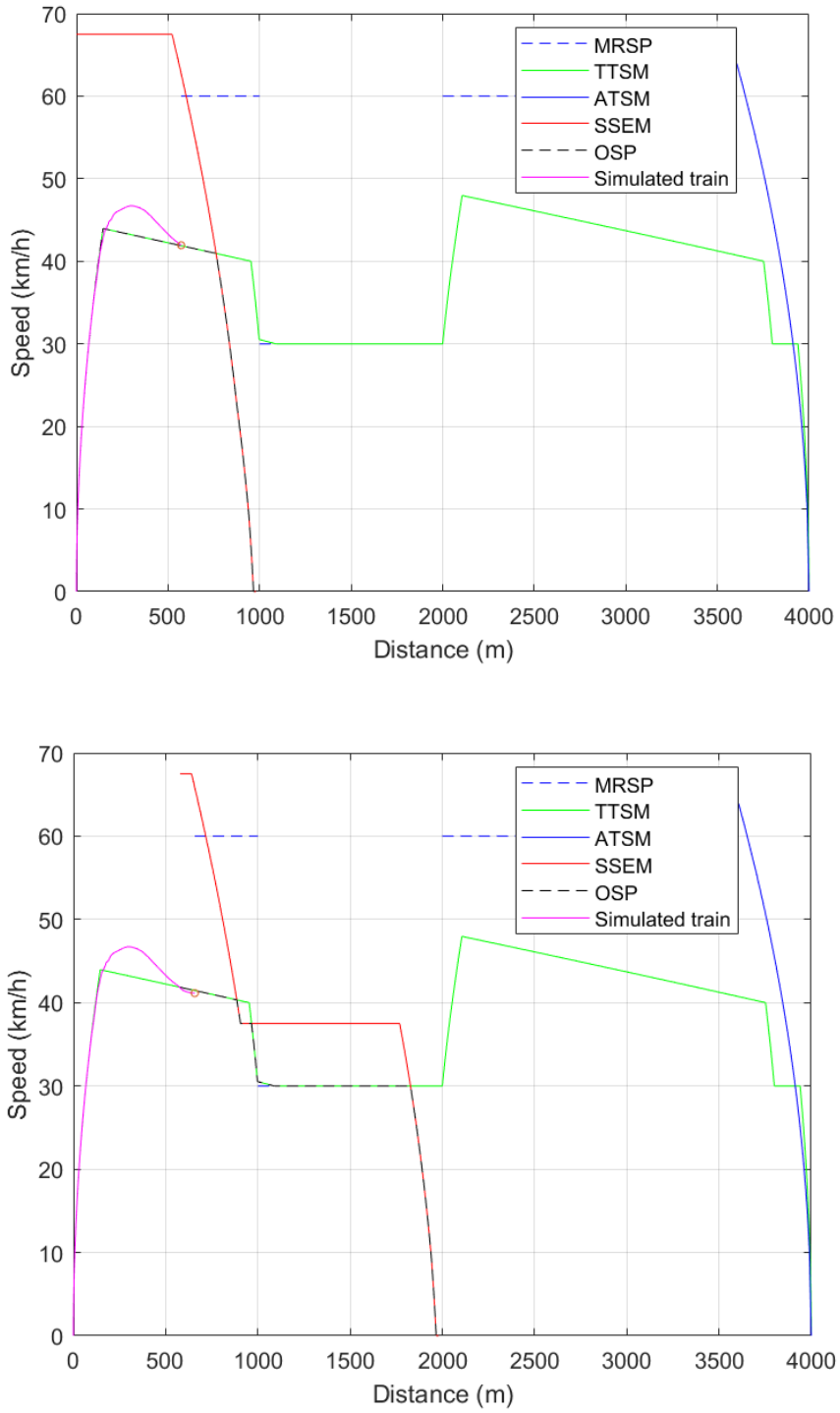


Figure 85: Extension of the Movement Authority when passing a "Voie Libre" signal. The first figure is just before passing the signal. The second figure is just after passing the signal.

When the Movement Authority (MA) is extended *while the train is actively following the SSEM profile* (i.e., the EOA-related braking curve), this event triggers a following recalculations:

- a recalculation of the SSEM profile (Condition C_SSEM_01)
- a recalculation of the TTSM profile (Condition C_TTSM_05), followed by
- OSP Recalculation (Condition C_OSP_01)

This behavior is confirmed in Figure 86, which all profiles (TTSM, ATSM, SSEM, and OSP) before and after the MA extension.

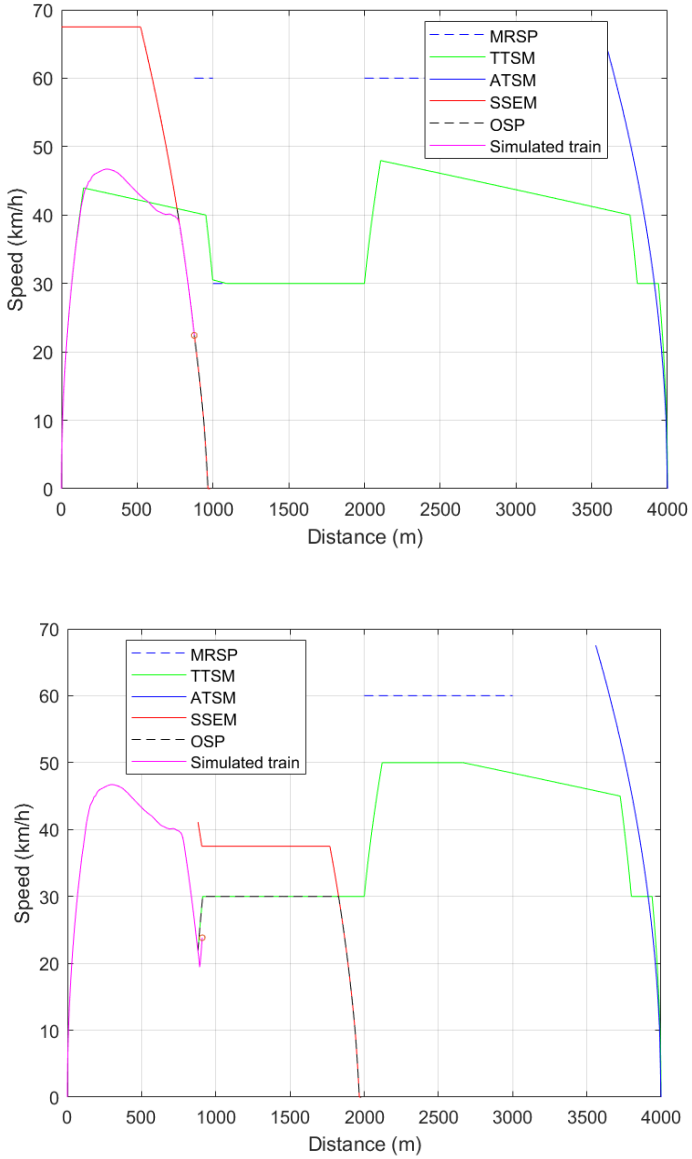
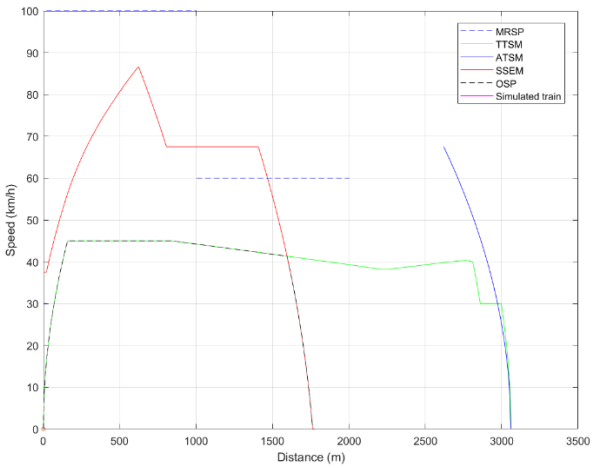
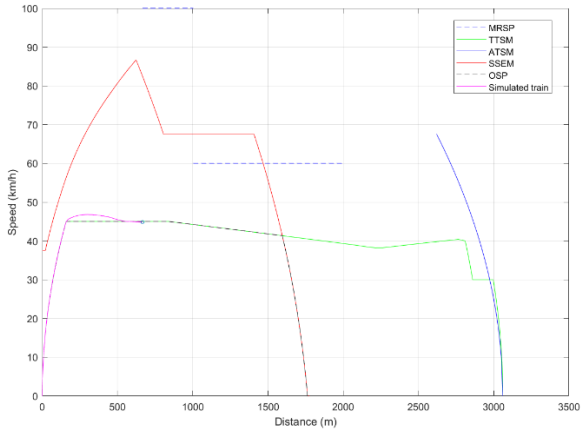
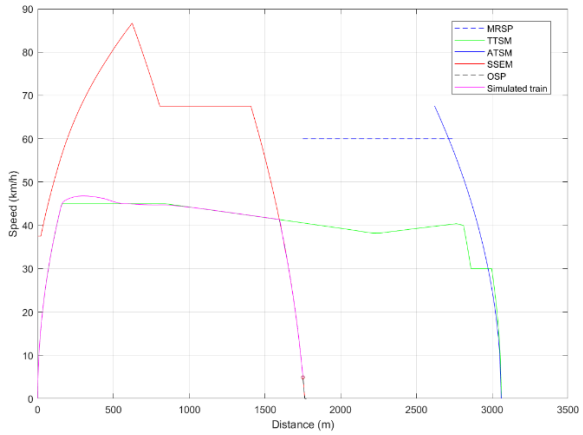


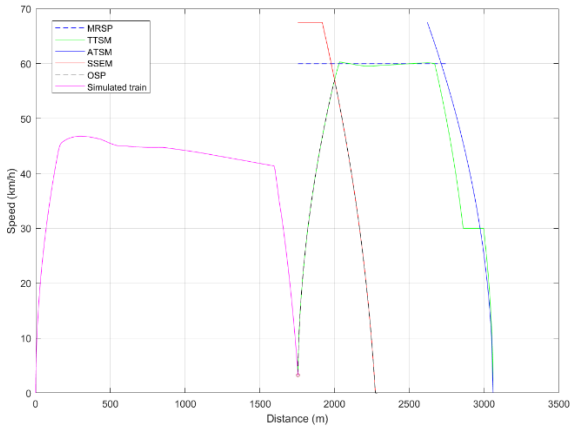
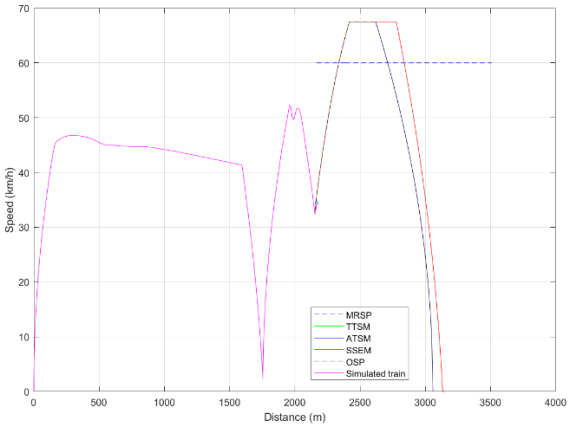
Figure 86: Extension of the Movement Authority when passing a "Voie Libre" signal while the train is following the SSEM profile. The first figure is just before passing the signal. The second figure is just after passing the signal.

4.2.5.5 End-to-end validation of ADM module

To validate the overall behavior of the ADM, Test Scenarios 1 to 3 are executed by stimulating the ADM module with the expected inputs for each scenario.

Table 14 presents the key stages of Scenario 2 simulation (precise stopping at target because of a signal with state “Carré”).

Simulation stage	Profiles diagram	Description
Departure from depot (in front of signal C01)		The train begins its technical movement. The TTSM, ATSM, SSEM, and OSP profiles are computed.
Passing signal C03 at Warning state: MA unchanged		The MA is not extended when passing the signal. The ADM inputs remain unchanged, and no recalculation is triggered.
Arrival at signal C05 with Non-crossable stop state: Train stops		Upon reaching the EOA (End of Authority), the train brakes until it comes to a complete stop.

Signal C05 switches to Warning state: MA extension		The MA is extended by 1 section (510m). The train recalculates the TTSM, ATSM, SSEM, and OSP profiles. The TTSM profile now reaches higher speeds: the stop has caused a delay in the mission schedule, requiring stronger acceleration to catch up.
Arrival at signal C07 at Track Free state: MA extension		The MA is extended again by 860m. The train recalculates the TTSM, ATSM, SSEM, and OSP profiles. However, the TTSM calculation fails because it is no longer possible to find a profile that satisfies the timetable constraints at the destination station. Indeed, too much delay has accumulated due to successive stops. The train therefore follows a maximum performance driving strategy until the stopping point.

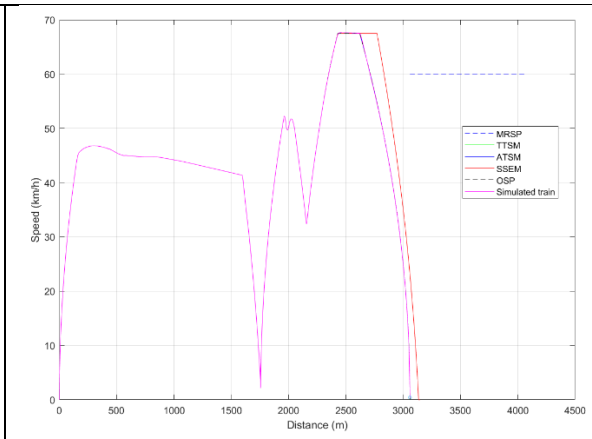
Arrival at station		The train brakes to stop at the target stopping point. The stopping accuracy is 70 cm.

Table 14: ADM validation on scenario 2

The command, shown in Figure 87, is consistent with the speed profile: the typical driving phases are clearly identifiable—traction, cruising (speed holding), coasting, and braking.

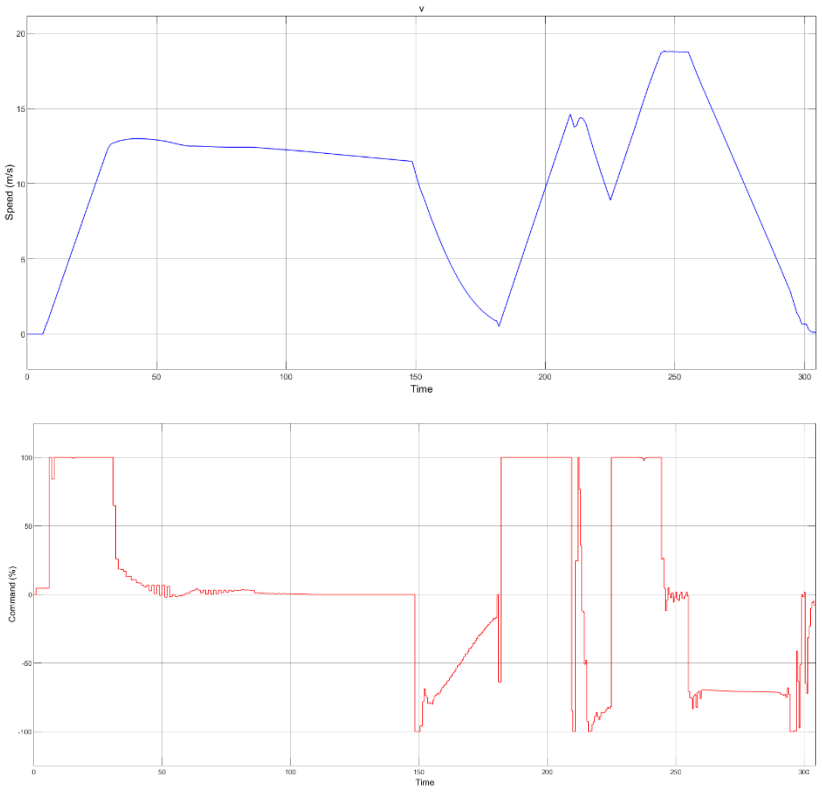


Figure 87: Evolution of speed and command through simulation (scenario 2)

The ADM's behavior fully aligns with expected performance in Scenario 2 (precise stopping at target). This consistency is equally validated across the two additional test scenarios under investigation.

4.2.6 Global validation of the ATO GoA4 model

Following the unit validation of each module composing the ATO GoA4 model, this section focuses on the global validation of the integrated system. The objective is to assess the model's performance and robustness across nominal operating scenarios as well as perturbed conditions, ensuring its reliability in real-world railway environments.

4.2.6.1 Nominal scenarios

The GoA4 model successfully implements the selected architecture in scenarios that mirror real-world nominal operational conditions. It offers full parameterization capabilities, including the static speed profile, gradient profile, rolling stock dynamics, timetable constraints (with Timing Point positions and schedules), and signal states.

Performance evaluation of the MPC controller demonstrates its ability to accurately track the speed profile while effectively recovering delays—without exceeding the static speed limits. The controller's tuning flexibility allows for optimization of the trade-off between delay recovery responsiveness and energy efficiency.

A key challenge arises in real-time execution when the simulation stutters due to computational load of the perception module, during SCV perception requests (further details are provided in the PER validation section).

The model's behavior aligns with expectations across all three nominal scenarios (detailed in Section 4.2.2.2), with validation results summarized in the Table 15.

Scenario	General behaviour	Validation criteria	Status	Remarks
ID 01 : Track free	The train adheres to the TTSM speed profile throughout the journey and achieves a high-precision stop at the stopping point.	Set in motion	OK	
		Compliance with maximum static speed constraint	OK	
		Detection of track free and no brake	OK	MA automatically extended by one block upon encountering each "Track Free" signal.
		Stop at platform mark at station	OK	Stopping precision: 6 cm
ID 02 : Stop	The train undergoes successive decelerations (SSEM tracking) due to restrictive signalling, while maintaining compliance with all speed limits and signal aspects.	Set in motion	OK	
		Compliance with maximum static speed constraint	OK	Speed limits respected even during delay recovery attempts following successive decelerations.
		Warning detection (C03) and speed decrease	OK	MA not extended when passing signal C03.

		Non crossable stop detection and stop before signal	OK	Train stops safely before the signal.
		Detect change: stop to warning	OK	MA extended by one section when crossing the signal.
		Detect track free (C07)	OK	Detection range significantly impacts performance. Late detection triggers emergency braking due to MA expiration (C05 was in warning state).
		Stop at platform mark at station	OK	Precision stop: 6 cm Arrival delay accumulated due to successive slowdowns at signals C05 and C07.
ID 03 : Switch crossing	The train detects and complies with signalling announcing a 30 km/h speed restriction at the switch, adjusting its speed supervision accordingly.	Set in motion	OK	
		Compliance with maximum static speed constraint	OK	Compliance maintained **including at the switch** (30 kph limit enforced).
		Detect track free (C03) and no brake	OK	
		Detect signal announcement for speed reduction to 30kph: reduce speed	OK	Signal properly treated with the 30 kph speed limit applied at the switch.
		Detect signal speed execution: maintain speed at 30kph on switch	OK	
		Stop at platform mark at station	OK	Stopping precision: 10 cm

Table 15: ATO GoA4 validation in the three test nominal scenarios

Note on Perception Robustness: While the perception system occasionally experiences interference from adjacent track signals, the implemented filtering mechanisms effectively mitigate these disturbances. In all evaluated nominal scenarios, such interference was successfully excluded

from the algorithmic chain, ensuring that the ATO GoA4 model's decision-making process remains unaffected. This demonstrates the system's resilience to environmental noise in standard operating conditions.

4.2.6.2 Robustness under perturbed conditions

To evaluate the ATO GoA4 model's robustness, this section introduces controlled perturbations designed to test not only the perception algorithm's ability to handle environmental fluctuations but also its cascading effects on the model's overall behaviour, in the context of the signalling scenarios defined in Section 4.2.2.2.

Controlled perturbations (rain, snow, fog) were injected into Scenario ID01 (Track Free), thanks to the simulation control panel (see section), to evaluate their effects on the train's signal perception, speed control, and stopping accuracy. The tables below summarize the observed impacts on system performance.

Weather conditions

Perturbation	Environmental effects in the model	Observed performance impact on scenario 01 (Track free)
Rain 	Falling water drops Sky appearance change (rainy) Water drops on camera lens Modified object surface (wet texture)	No performance degradation observed
Snow 	Falling snowflakes Sky appearance change (snowy) Object surfaces covered with a white layer Light fog bank reducing the visibility distance	No performance degradation observed
Fog 	Fog bank reducing significantly the visibility distance	Correct but delayed detection due to the reduced visibility.

Table 16: Robustness of the model under altered weather conditions

Lightning conditions

Perturbation	Environmental effects in the model	Observed performance impact on scenario 01 (Track free)
Sunset 	Sky appearance change Sun glare effect	No performance degradation observed
Night 	Sky appearance change Low light environment Train headlights	Signal is visible later (compared to the nominal scenario), when it enters the headlight coverage zone. Slight degradation in lamp perception performance caused by headlight reflections on signal glass surfaces. No impact on global performance.

Table 17: Robustness of the GoA4 model under altered light conditions

Train tail light

Figure 88: Train tail light perturbation

Tests were conducted to assess whether the rear lights of a crossing train (visible on an adjacent track) could interfere with signal detection. Observations confirmed that the system's object recognition prioritizes chassis geometry over individual light sources. Consequently, no performance degradation was observed.

Signalling failure

In the event of a signalling failure, or when the signal detection module (PER) is unable to reliably read the signal aspect, the SCV module applies a fail-safe strategy by assuming the most restrictive plausible aspect for the next signal, based on the last valid one. For example, if the previous signal was Track free, the following signal is conservatively treated as Warning; if the previous signal was already at Warning, the next one is assumed to be at non-crossable stop.

Due to time constraints, the subsequent operational procedure is not yet modelled: in practice, the train would request authorisation from the ground supervision system to be allowed to pass a failed or unreadable signal.

The simulation's flexible parameterization enabled rapid testing of diverse scenarios with injected perturbations, providing an efficient framework to evaluate the system's robustness under varying environmental conditions. The results demonstrate the model's strong robustness to the environmental perturbations tested, with no critical performance degradation observed in Scenario 01 (Track Free). Based on these findings, practical guidelines could be proposed to enhance safety in edge cases. For instance, reduced approach speed could be recommended in dense fog conditions to allow the perception algorithm sufficient time for valid signal detection.

To further enhance the model's resilience, future work should prioritize multi-perturbation testing (e.g., combining night, rain, and fog) to assess cumulative effects on perception performance, alongside expanded validation across all signalling scenarios.

4.2.7 Parameters

To ensure the ATO GoA4 model accurately represents real-world operations, a comprehensive set of parameters must be defined. These parameters span infrastructure characteristics, train dynamics, and operational constraints, each playing a critical role in shaping the model's behaviour.

Parameter	Usage	Source
Timing point (position and schedule)	TTSM ATSM and SSEM profile generation	Operational document: timetable (infrasctructure manager)
Signal position and possible states	MA computation	Line diagram : diagram with line infrastructure (position of track points, stations and signalling equipment (infrasctructure manager)

Limit speed profile	Elaboration de la MA. Tracé TTSM.	Line diagram : diagram with line infrastructure (position of track points, stations and signalling equipment (infrasctructure manager)
Gradient profile	EVC deceleration, TTSM, train model	Line elevation data (infrasctructure manager)
Line geometry and environment	3D model : rail placement and environment	Mapbox cartographic data
ERTMS safe braking characteristics	SSEM	EVC parametrization on reference train.
Braking characteristics	TTSM. Train model	Train manufacturer's braking test report
Traction characteristics	TTSM. Train model	Train manufacturer's traction test report
Train motion resistance coefficients	TTSM. Train model	Train manufacturer's aerodynamic test report (simulation and laboratory)
Train mass	TTSM. Train model	Standard load from mass evaluation analysis from train manufacturer

Table 18: Model parameters

Parameters must remain consistent across all subsystems to ensure coherent system behaviour and valid simulation results. For example, ERTMS braking deceleration used by the signalling and safety modules must match the braking characteristics implemented in the train dynamics model.

Shared parameters should be centralized when appropriate—stored in the global data dictionary and inherited by subsystem dictionaries—so they are defined once and propagated everywhere. Centralization also simplifies scenario variations and sensitivity studies: changing a single, well-documented parameter entry updates every dependent model, reduces the risk of hidden inconsistencies, and accelerates verification. More generally, parameters must be easy to modify and traceable (clear naming, versioning and provenance) to support iterative development, testing of degraded conditions, and reproducible validation results.

4.2.8 ATO GoA4 simulator

An ATO GoA4 simulator was built using the ATO GoA4 model and an interface developed with App Designer, a dedicated tool to develop applications in Simulink. This interface provides real-time control over scenario parameters:

- Environment: Users can simulate adverse weather conditions (e.g., rain, snow, fog) and lighting variations (day, night, or sunset). These adjustments directly challenge the perception algorithm, exposing potential vulnerabilities in detecting rolling stock or interpreting signal states.

- Signal states: A dropdown menu enables quick modifications to signal states, allowing for rapid testing of failure modes or emergency scenarios.
- Mission: To cover the test scenarios, the simulator supports multiple track missions

This tool not only enhances communication and demonstration of ATO GoA4 capabilities but also serves as a powerful validation platform for testing diverse operational conditions. The ability to modify easily the simulation parameters through the interface allows to reduce the time taken to perform multiple test cases.

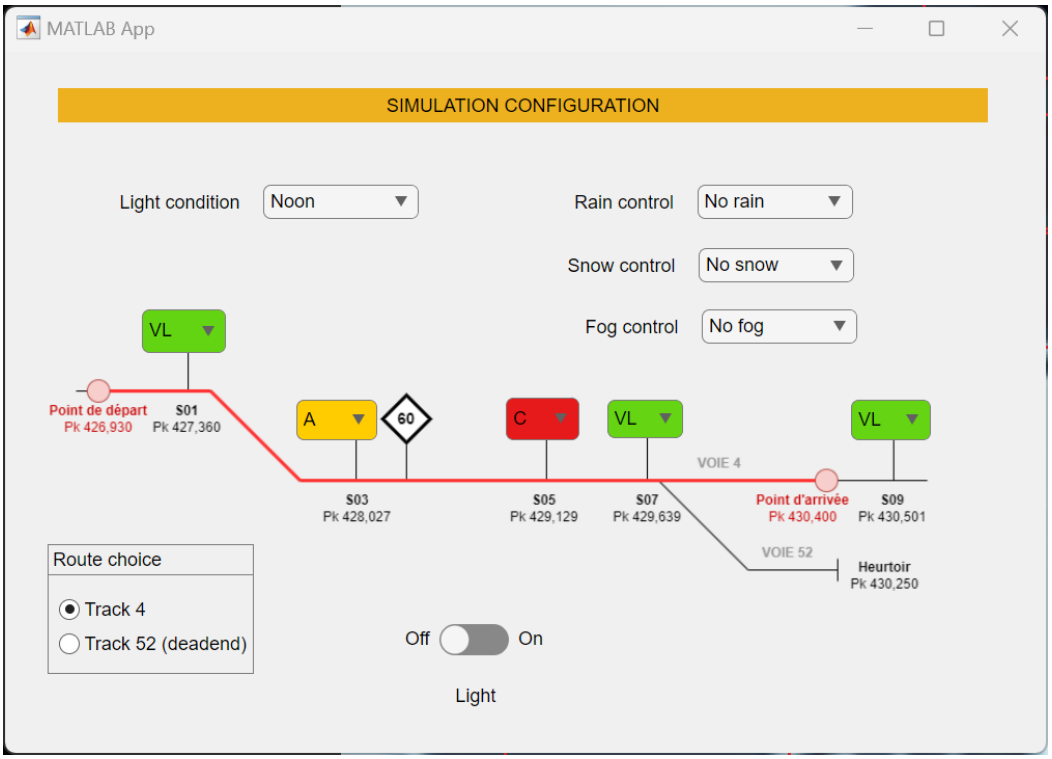


Figure 89: User interface of the ATO GoA4 simulator

4.2.9 Modelling feedback report

The GoA4 ATO modelling work demonstrated the effectiveness of a modular approach in achieving representative behaviour rapidly. Modular design enabled both algorithm reuse (e.g., the GoA2 ATO-OB was reused adapted for ADM in the GoA4 model) and parallel teamwork, significantly accelerating development.

The modelling activity also raised several specification issues and implementation questions for GoA4 ATO systems, summarized in Table 19.

Issue	Description	Modelling decision
Stopping distance in front of signals	When stopping before a signal, the signal must remain visible to the perception camera to detect state changes (e.g., red to green). Stopping exactly at	Target stopping position was set 10 meters upstream of the signal to ensure continuous visibility.

	the signal may place it outside the camera field of view.	
Release Speed Monitoring (RSM) handling	Release speeds were not modelled, because there were a source of discontinuities in the SSEM profile at RSM start points. RSM start point is located on a SBD curve but Subset-130 does not provide sufficient data to compute this SBD curve.	RSM was not included in the model.
Distance to EBI provided by ETCS	According to Subset-130, ETCS supplies the ATO-OB with the location of the EBI point (N_LOC_EBI). What is the intended use of this parameter in ATO trajectory planning?	This parameter was not used in the model. Its role requires clarification in specifications.
Replacement of "On Sight" mode	In manual driving, situations such as approaching a buffer stop (e.g., test scenario 3) are handled via "On Sight" driving mode. In driverless operation, this mode is unavailable. How should such situations be managed?	Assumption that buffer stop position is transmitted to the onboard system by the balises and treated as an SvL.

Table 19: Specification issues for ATO GoA4

Several directions can extend the GoA4 modelling work:

- **Enhanced Automatic Protection Module (APM):** Model interactions with remote control for departure/arrival procedures and degraded mode handling.
- **Hardware integration:** Progress toward Processor-in-the-Loop (PIL) simulation to validate real-time algorithm performance on target hardware. It should be noted that perception algorithms face a critical trade-off between detection quality (accuracy, robustness) and computational load, especially for real-time onboard implementation.
- **Obstacle perception chain:** Integrate obstacle detection and the associated emergency reaction chain.

4.3 MULTIPHYSICS SIMULATION RESULS

4.3.1 Use case : freight trains modelling

The complex dynamic behaviours of freight trains (such as in-train forces, coupler impacts, and brake propagation effects) make their modelling particularly challenging. These phenomena not only impact safety and asset integrity but also complicate the design and implementation of Automated Train Operation (ATO) strategies. For ATO systems to be reliable and efficient in freight applications, they must account for these highly nonlinear and coupled dynamic interactions, which introduce significant technical hurdles.

This section presents multiphysics simulation results, based on the freight train use case, following two complementary approaches: simulation with Simscape, the Multiphysics simulation tool integrated in Simulink, and cosimulation between Simulink and SIMPACK. Both Simscape and SIMPACK were presented in section 3.3.2.3. The following subsections describe the modelling assumptions, the co-simulation architecture, and the main results obtained on the freight train scenario, highlighting the specific insights each approach brings to the development and validation of ATO functions.

The multiphysics models developed in this project do not aim at exhaustively capture all aspects of freight train dynamics. Instead, their purpose is to highlight the value of multiphysics tools in modelling the complex dynamic behaviors of freight trains, including coupling interactions, in-train forces, and brake propagation. These models also serve to illustrate the key challenges associated with such simulations (representativity trade-offs and computational performance constraints) which are critical considerations in both research and real-world applications of ATO.

This section is entirely independent from Sections 4.1 and 4.2, which focus on GoA2 and GoA4 automation models designed for passenger train use cases. The train models presented here are not intended for integration with those earlier models, as they address a distinct use case: freight trains, which introduce unique dynamic challenges not covered in the previous sections (e.g., in-train forces, coupler interactions, and brake propagation in long, heavy convoys).

4.3.2 Simscape

4.3.2.1 Wagon coupler

A coupler is a mechanical device designed to connect two rail vehicles, enabling the formation of a train. In this project, the modelled coupler is a buffers-and-chain coupler. This coupler consists of two key subsystems:

- **Buffers**: These elements incorporate internal damping mechanisms to absorb longitudinal shocks during traction and braking. By dissipating energy, buffers mitigate the impact forces transmitted between vehicles, thereby enhancing safety and reducing wear on the rolling stock.
- **Coupling chain**: A flexible chain composed of multiple links (three in this specific model) that transmits pulling forces between vehicles. The chain is connected to the vehicle's draw gear via a hook.

Figure 90: Freight train coupler

The Simscape model of the coupler is shown in Figure 91. It consists of two blocks associated with the wagons (in green in Figure 92), each connected to the ground (in grey) through a prismatic joint representing longitudinal motion.

The buffers are represented by cylindrical elements (in cyan), also connected to the wagon via a prismatic joint, with the contact force between buffer elements explicitly modelled. In this model, only a single buffer contact is represented in Simscape, although two physical contacts exist between adjacent wagons. The longitudinal forces induced by both contacts can be accurately modelled using a single equivalent contact by adjusting its characteristics (e.g., stiffness, damping). This simplification reduces the computational load for the simulator, as fewer physical elements need to be processed while maintaining the fidelity of the dynamic behaviour.

The coupling chain is represented by a series of three parallelepiped bodies connected by revolute joints along the chain, allowing relative rotation between consecutive links.

This arrangement captures both the compressive behaviour of the buffers and the tensile behaviour of the chain, which together govern the longitudinal force transmission between wagons.

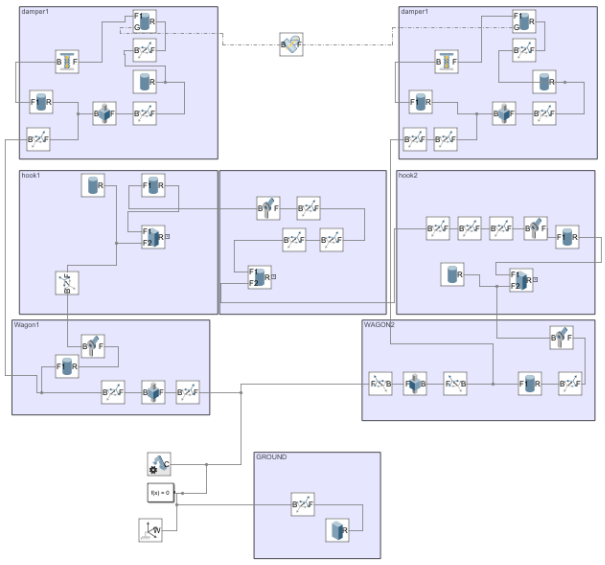


Figure 91: Simscape coupler model

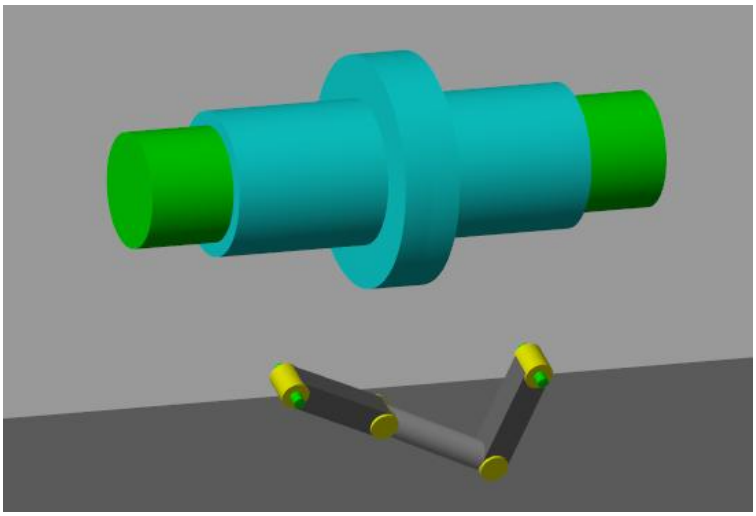


Figure 92: Coupler model 3D representation on Simscape

Since the purpose of this work is to demonstrate the capabilities of the modelling tool, the coupler model was not parameterized with real-world values (e.g., component masses, buffer damping, or joint and sliding friction). Figure 93 provides an example of the simulation results, illustrating the displacement of one of the buffers relative to the rail. The large displacement at the beginning of the simulation corresponds to the transient phase, during which the coupler settles into an equilibrium position. The persistent oscillations observed in the results stem from the absence of energy dissipation modelling in the system. During the simulation, a 3D animation of the coupler's movement is available, based on the same model as the snapshot provided in Figure 92. This visualization allows for real-time observation of the dynamic behaviour, including buffer compression, chain tension, and relative displacements between components.

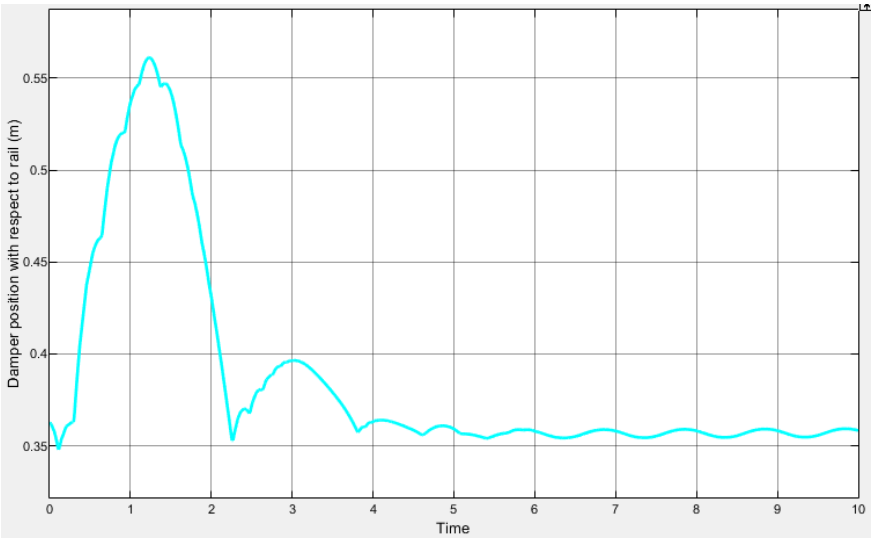


Figure 93: Buffer displacement during simulation along longitudinal axis

4.3.2.2 Longitudinal dynamics

To model the longitudinal dynamics in a simple way, one locomotive and four wagons are represented as rigid parallelepiped bodies constrained to move along the track through prismatic joints, without explicitly modelling wheel–rail contact. The previously developed coupler model is reused between adjacent vehicles, which illustrates again the efficiency of the modular modelling approach.

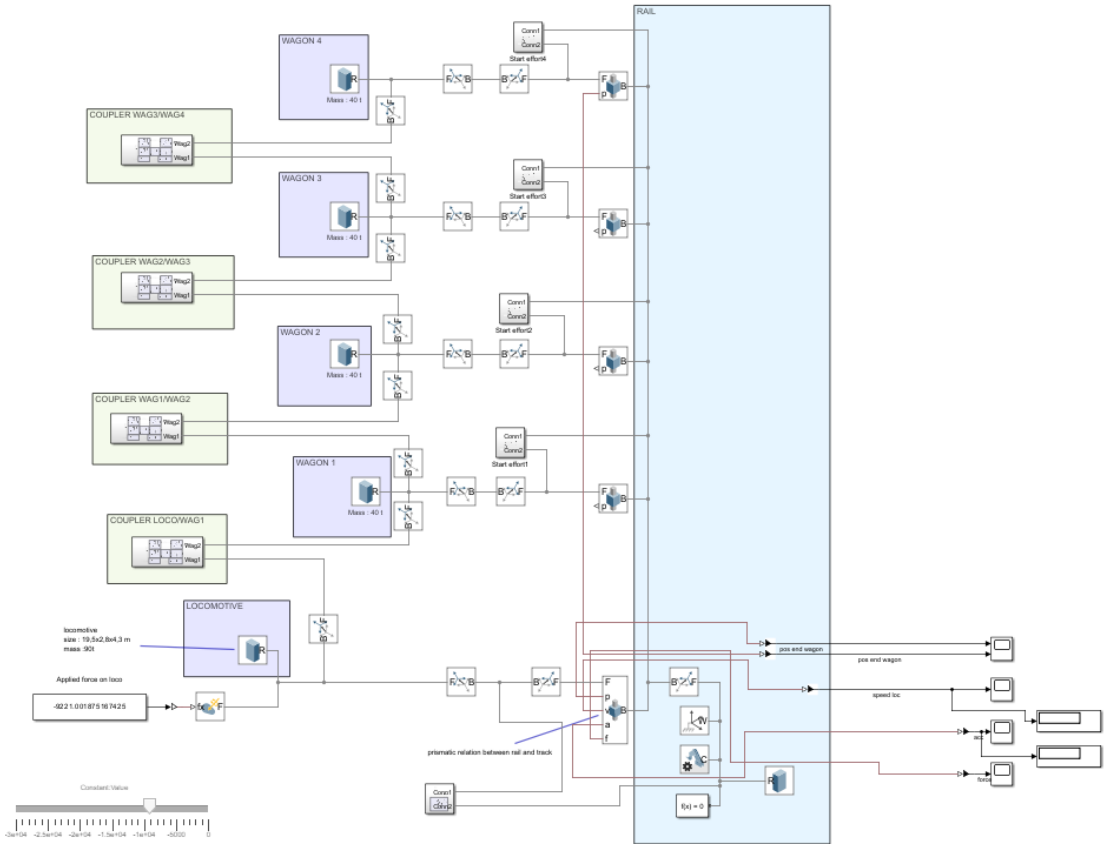


Figure 94: Simscape full train model

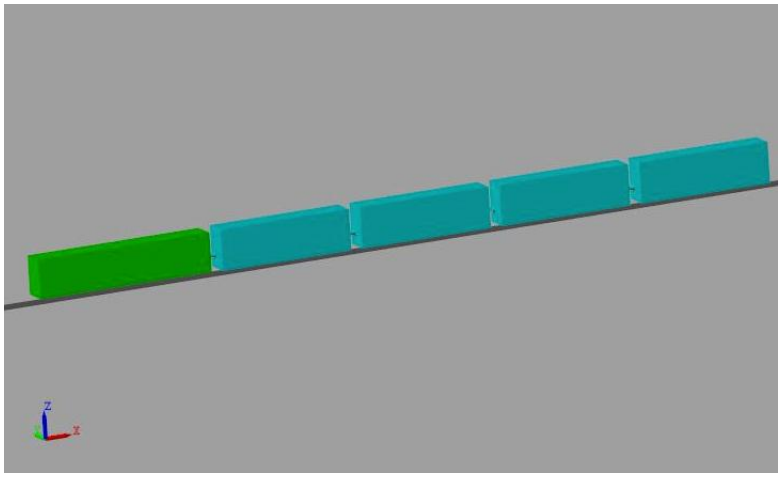


Figure 95: Full train model 3D representation on Simscape

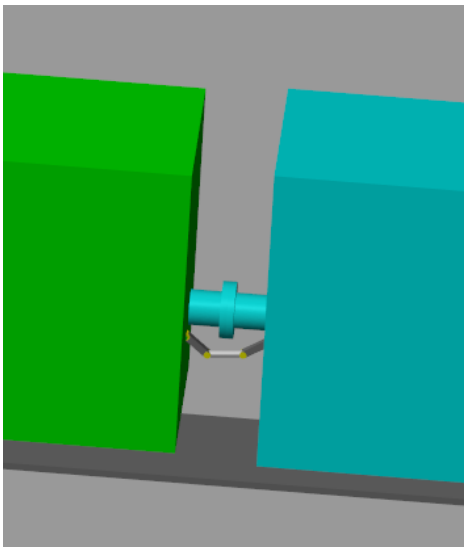


Figure 96: Zoom on coupler in Simscape full train model

Figure 97 presents an example of the simulation results, illustrating the variation in distance between the locomotive and the last wagon. The significant initial fluctuation in this distance is due to a transient phenomenon as the couplers settle into their equilibrium positions. The simulation effectively captures the complexity of the system's dynamics, which would have been challenging to replicate using a traditional analytical representation of the system.

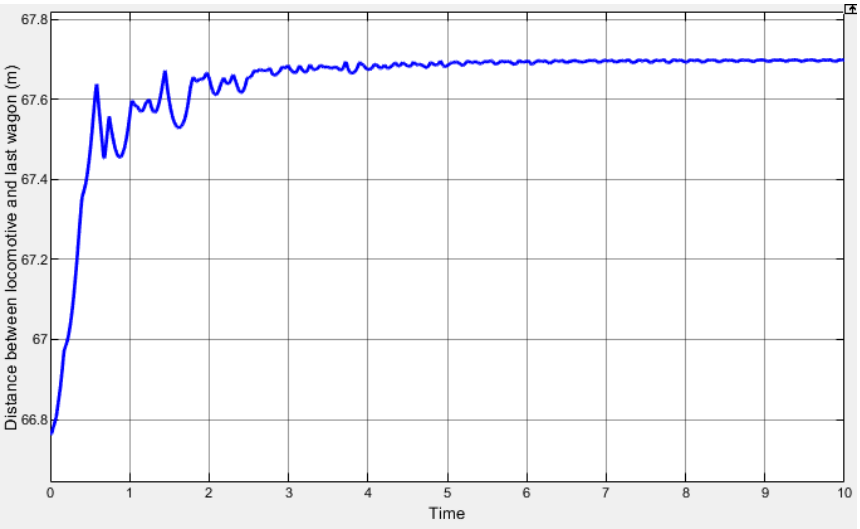


Figure 97: Evolution of the distance between locomotive and rear wagon

4.3.3 Cosimulation with SIMPACK

4.3.3.1 Longitudinal dynamics on straight track

Following a similar methodology to the Simscape-based longitudinal dynamics studies, a freight train model was developed in SIMPACK and exported to Simulink as S-Function for co-simulation. An incremental development approach was adopted to progressively validate the model's behaviour and integration. In the first stage, the multibody model used in this study is based on a train formation model provided by Dassault Systèmes. This model was selected due to its simplified representation, which minimizes nonlinearities and thus reduces computational time. The model consists of a

locomotive with a full mass of 31,300 kg on rail, supported by two bogies. The model takes as inputs the torque applied at each axle, and outputs the longitudinal acceleration measured at the centre of the carbody.

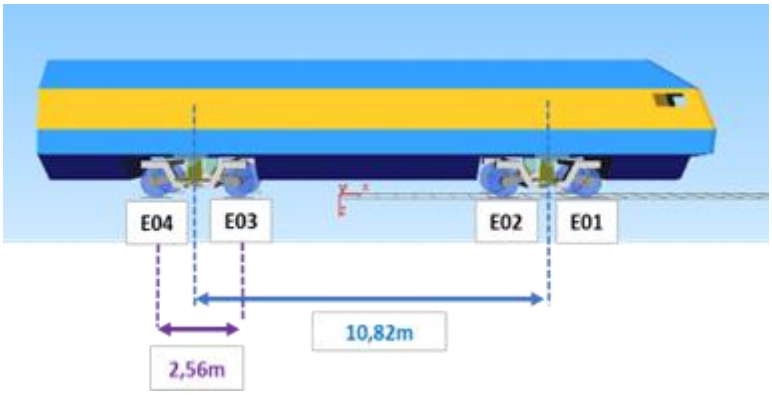


Figure 98: Locomotive model in SIMPACK

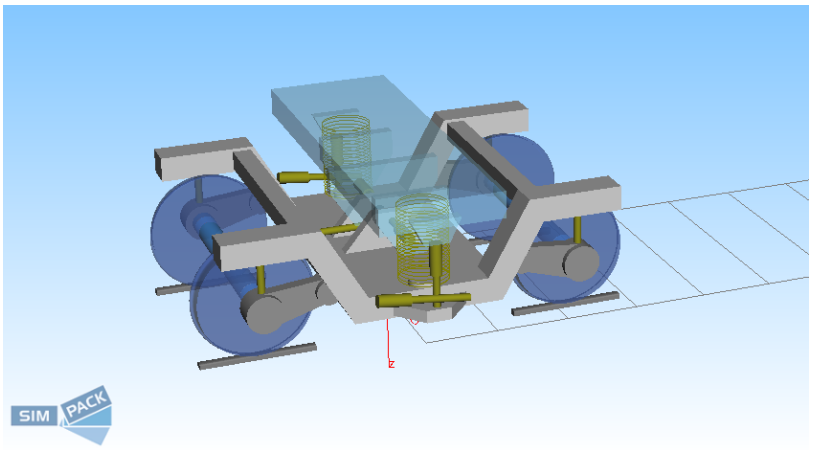


Figure 99: Bogie modelling on SIMPACK

Figure 100 presents the acceleration output along with the corresponding velocity and position obtained through successive integrations. In this example, the input command consists of an identical traction torque applied to all four axles during the first 10 seconds of simulation, followed by a coasting phase. The oscillations observed during the transitions between command phasis are explained by the dynamics of the traction rod between the bogies and the carbody. It should be noted that these dampers had not been modelled in the Simscape approach.

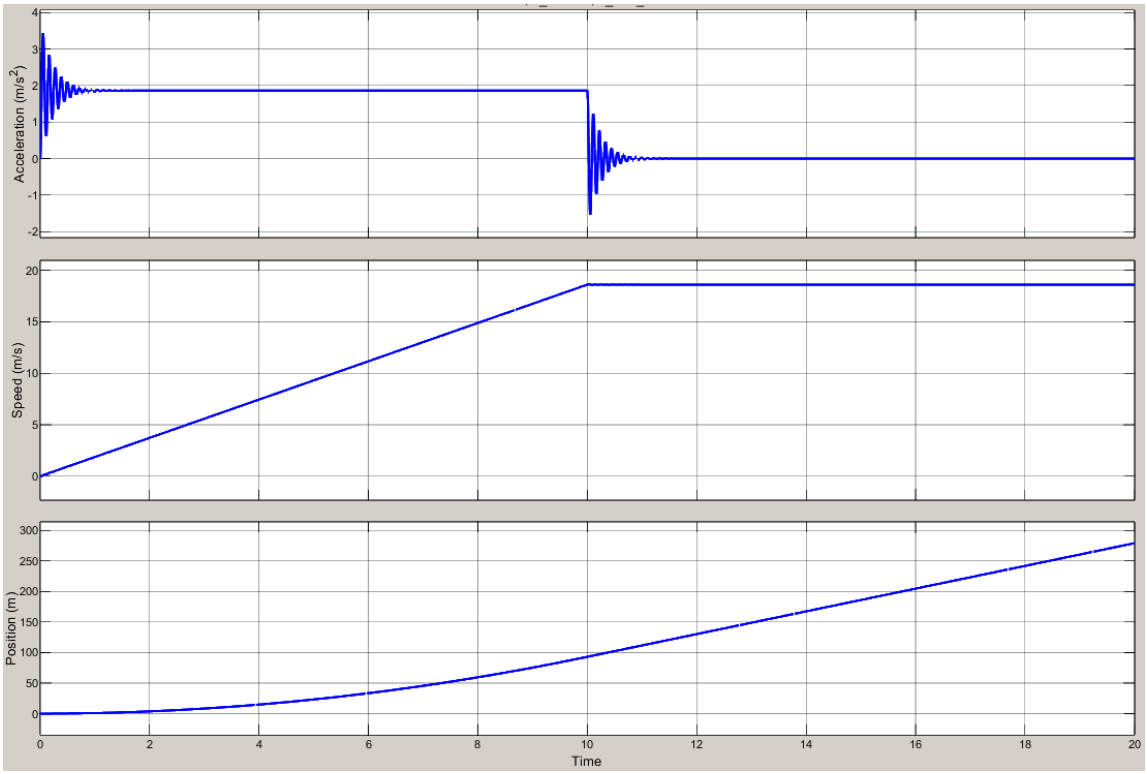


Figure 100: Cosimulation result for the locomotive on a straight track

In a second stage, the model was extended to include four wagons, whose individual masses are configurable as parameters directly from the Simulink environment. The results presented hereafter correspond to the following wagon mass distribution (counted from the locomotive to the rear of the train): $m_{wagon1} = 20\,000\ kg$, $m_{wagon2} = 30\,000\ kg$, $m_{wagon3} = 40\,000\ kg$, $m_{wagon4} = 70\,000\ kg$.

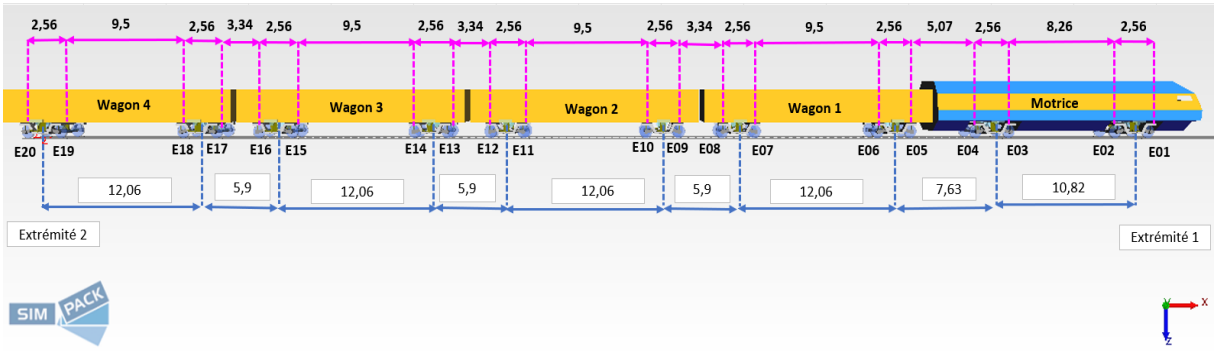


Figure 101: Full train model in SIMPACK

A simulation was carried out with a constant traction torque applied to the locomotive axles during the first 15 seconds, followed by a coasting phase, and then a braking phase initiated at 20 seconds (the same braking torque is applied to all axles of the train).

Figure 102 shows the inter-vehicle forces projected onto the longitudinal axis (in the direction of motion). Traction_Force_i denotes the force exerted by wagon i−1 (or the locomotive if i = 1) on wagon i. As expected, the forces are positive during traction and negative during braking.

Furthermore, the steady-state force ratios during the traction phase are consistent with the mass distribution.

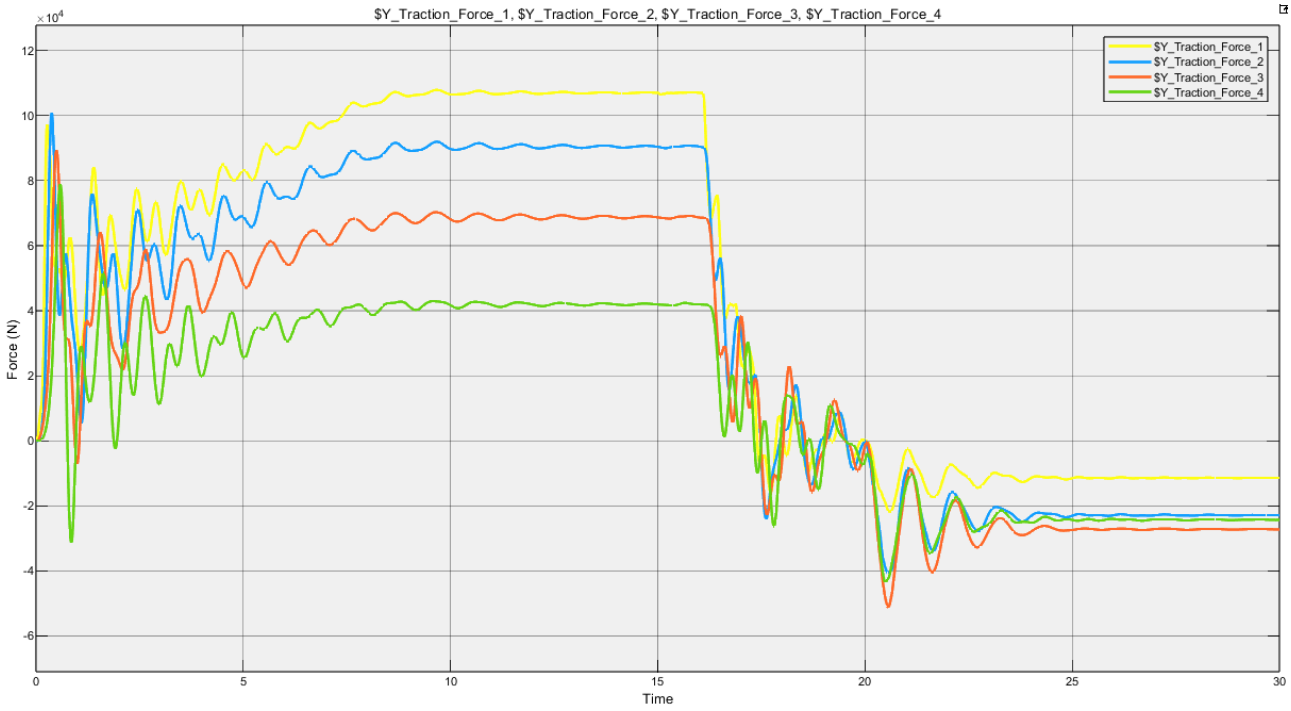


Figure 102: Forces between vehicles during simulation

This co-simulation approach provides a more representative model of the train's dynamic behaviour compared to the Simscape models, particularly in capturing the coupling dynamics and load distribution within the convoy. By varying the mass of individual wagons, significantly different dynamic behaviours emerge, highlighting the model's sensitivity to load configuration—a critical factor for freight train simulations.

This increased fidelity comes at the cost of longer simulation times, which must be taken into account when selecting the appropriate modelling approach for a given study objective. This computational overhead also poses a significant challenge for real-time implementation in onboard ATO systems, which demand optimized simulation performance.

4.3.3.2 Train dynamics on a non-straight track

An important challenge in the development of automated train operation systems is the accurate simulation of onboard sensors that feed the train localization function. In Hardware-in-the-Loop (HIL) testing environments, Inertial Measurement Units (IMUs) present a particular limitation: it is not feasible to physically stimulate the sensor in the same way it would be excited on an actual moving train. Current practice relies on either data replay from recorded runs or virtual sensor models applied to test scenarios, which limits the representativeness of the HIL bench compared to real operating conditions. Simulating IMU behaviour is made particularly challenging by the complexity of the lateral and vertical dynamics of the carbody, which result from the intricate wheel–rail contact mechanics. This contact phenomenon is explicitly modelled in SIMPACK, making it a valuable tool for generating realistic sensor signals in non-straight track scenarios.

Track Model

The test track consists of a sequence of representative geometric elements:

- Right-hand curve (radius of curvature 1,200 m) with a cant of 120 mm
- Left-hand curve (radius of curvature 1,200 m) with a cant of 120 mm
- Uphill gradient of 8 mm/m
- Downhill gradient of 8 mm/m

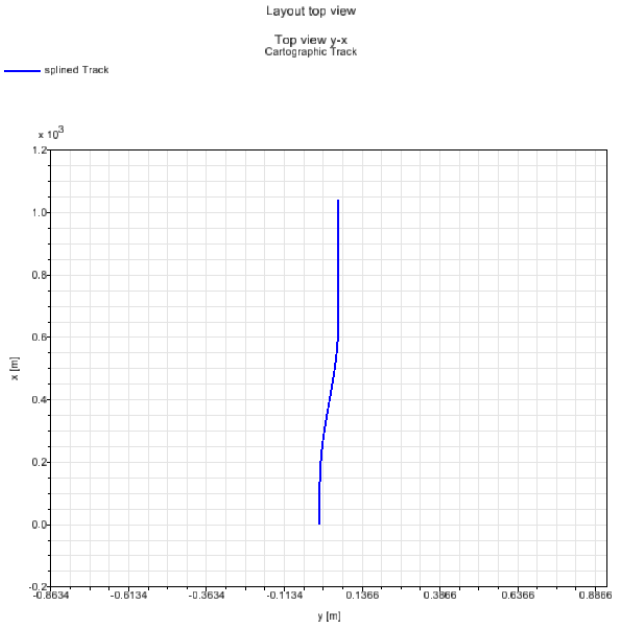


Figure 103: Top view of the track

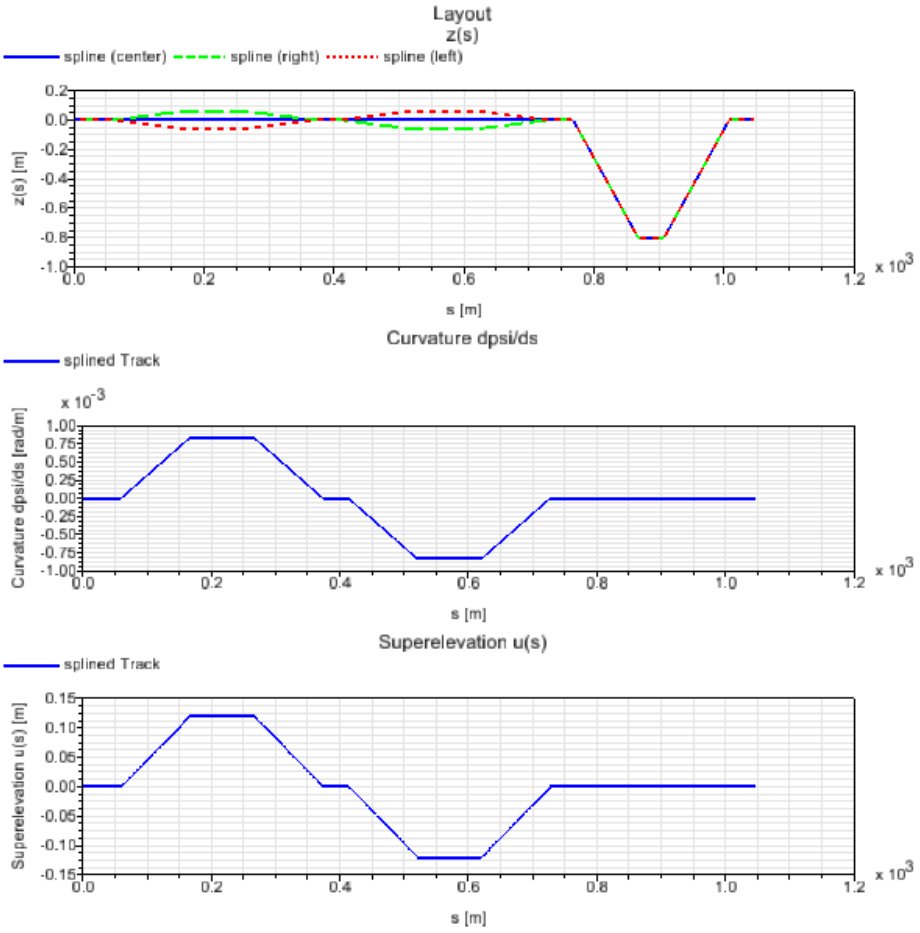


Figure 104: Curvature, elevation and superelevation along the track

SIMPACK Model

The SIMPACK model represents a four-axle locomotive with a total mass of 38 tonnes. A top view of the model is shown in Figure 105.

The model inputs consist of the torques applied at each of the four axles; for this simulation, identical traction torques are progressively ramped up on all axles and then released at t = 10 s, after which the train coasts through the remainder of the track.

The model outputs include four types of sensor measurements:

- Three-axis accelerations (along 3 axes as defined in Figure 105) measured at the centre of the carbody and at the centre of each of the two bogies (note: the z-axis is oriented downward toward the ground)
- Angular angles (roll, pitch, yaw) at the centre of the carbody, following the notation convention shown in Figure 105.

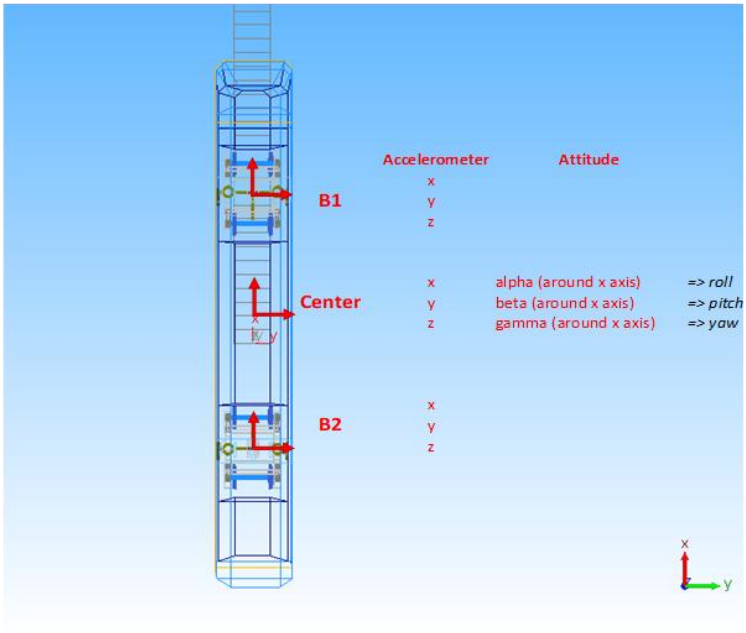


Figure 105: SIMPACK model outputs for non-straight track

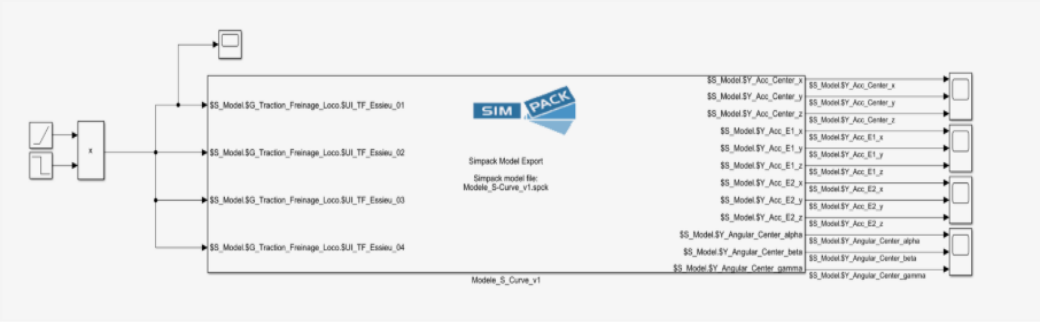


Figure 106: Simulink model for non-straight track

Simulation Results

Figure 107 and Figure 108 present the simulated IMU data recorded at the centre of the carbody. The following observations can be made:

- **Curved sections:** Lateral acceleration (y-axis) exhibits clear variations as the train negotiates the curves. The yaw angle varies continuously to follow the track curvature. Roll and pitch angles are induced by the track cant (superelevation).
- **Gradient sections:** Vertical acceleration (z-axis) changes as the train climbs or descends, and the pitch angle varies accordingly to reflect the track slope.
- **Dynamic effects:** Oscillations superimposed on the steady-state values are due to the suspension dampers explicitly modelled in SIMPACK.

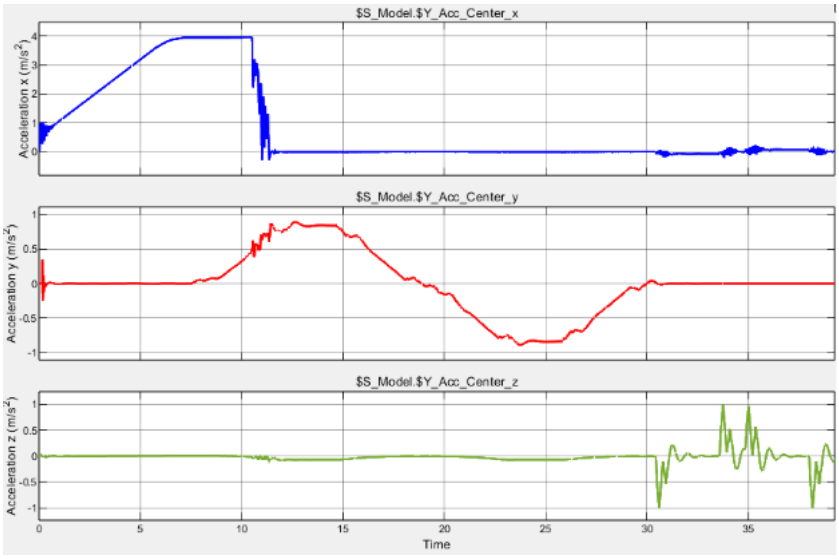


Figure 107: Accelerometer data at the centre of the carbody for the non-straight track simulation

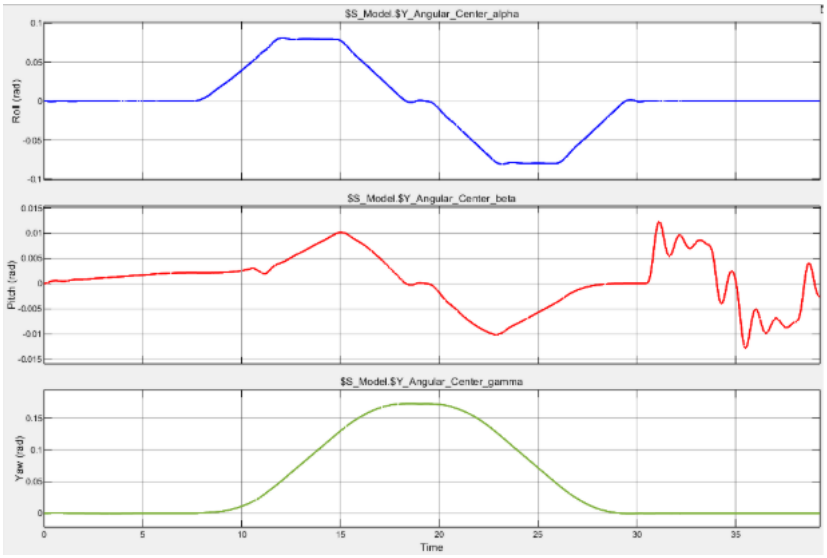


Figure 108: Angulars positions at the centre of the carbody for the non-straight track simulation

This co-simulation approach demonstrates the value of using a multibody dynamics tool to generate high-fidelity sensor data for testing and validating localization and control algorithms under realistic operating conditions, including track geometry effects that cannot be adequately reproduced on a HIL bench.

4.4 TESTS AND PROOF

This section aims at demonstrating how MBD tools can bridge the gap between theoretical safety requirements and practical implementation. By modelling a simplified use case—train separation enforced via signal logic and dynamic speed control—we illustrate the capacity of these tools to conduct exhaustive tests and to formalize safety proofs.

4.4.1 Use case

The use case focuses on proving a safety property related to train separation on a simplified railway line. The infrastructure is modelled as a line composed of four consecutive track sections, named areas A to D, and supervised by three lineside signals, denoted X, Y, and Z. Each signal can be in one of three aspects: track free, warning, or non-crossable stop. In the model, the state of each signal is not set manually but is derived from the occupancy of the corresponding areas. The model also accounts for the fact that, while a train is passing a signal, its physical length leads it to occupy two consecutive areas at the same time.

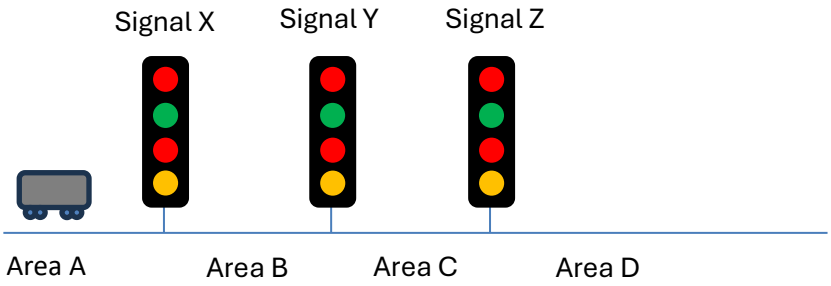


Figure 109: Line modelling: areas and signals

Two trains, labelled A and B, are considered in the scenario. The position of train A is controlled by the model and can evolve freely according to signalling logic and according to the constant duration to be spent in each zone (the train dynamics inside an area is out of modelling scope), whereas the position of train B is treated as an external input to the model. Each train can operate in two driving modes: full speed or signal-targeting mode. In full speed mode, the train runs at the line's nominal speed and will cross the next signal without stopping. In signal-targeting mode, the train adjusts its speed with the objective of stopping before the next signal. The driving mode is set to signal targeting mode if the train crosses a warning signal, which informs the driver that the next signal is in the state "non-crossable stop".

An example of scenario for the model is given in Figure 110. In this scenario, Train B is stationary in area C, which sets Signal X to warning and Signal Y to non-crossable stop due to the occupancy logic. Train A starts in area A, operating in full-speed mode. As it approaches Signal X, the model enforces a mode transition to signal-targeting: Train A decelerates and comes to a halt before Signal Y, respecting the non-crossable stop.

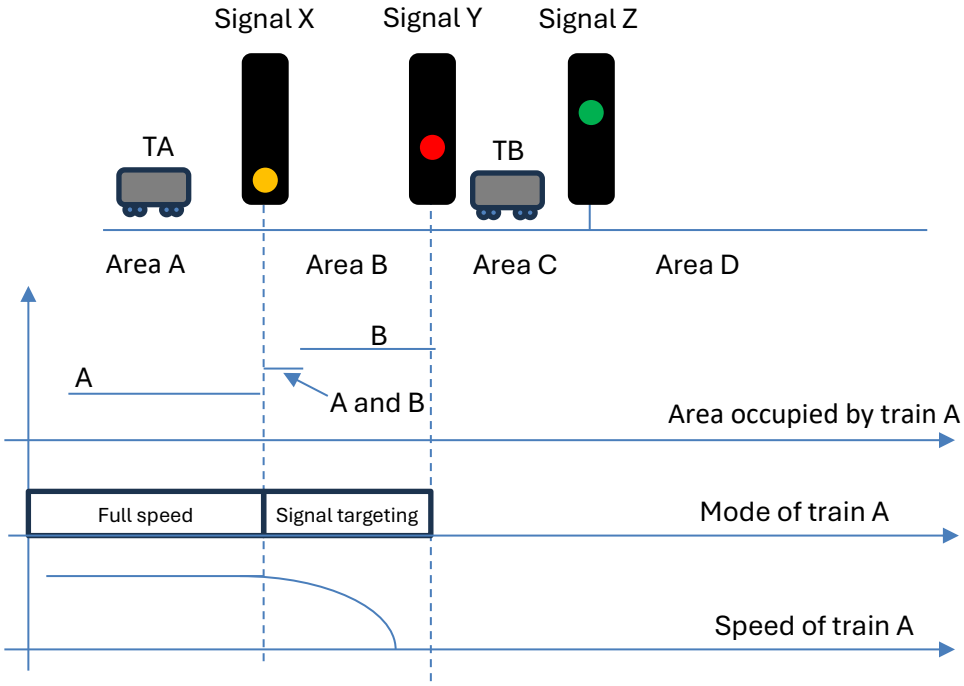


Figure 110: Typical scenario for the model

This model has been implemented in Simulink using Stateflow, a tool that allows to create finite state machine. As an example, the state machine to determine the signal state according to track occupancy is given in Figure 111.

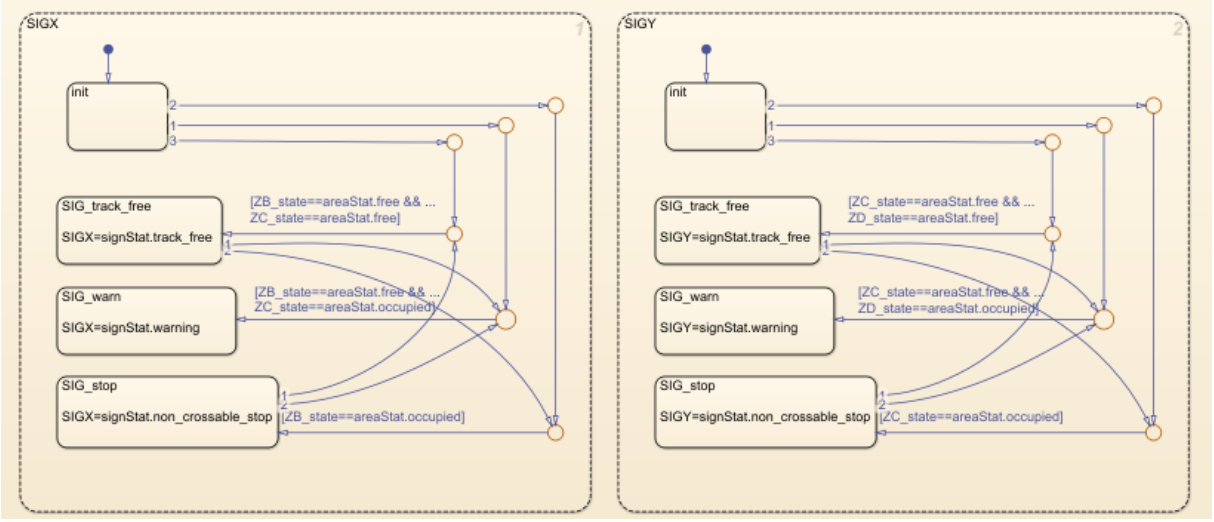


Figure 111: Stateflow state machine for signal state transitions

4.4.2 Test harness

The first modelling tool explored in this validation framework is the test harness. A test harness is a model that isolates the component under test, with inputs, outputs, and verification blocks configured for testing scenarios. This architectural approach enables focused validation by decoupling the system under evaluation from its operational environment, thereby facilitating controlled experimentation.

In the implementation for this use case, a Simulink referenced model was created containing the component under test—here, the signalling and track occupation system. This referenced model is then instantiated within the test harness, allowing its behaviour to be exercised under various conditions. Input signals can be generated in two ways:

- Using predefined blocks from Simulink's *Sources* library (e.g., step, ramp, constant, sine), which provide standard test patterns.
- Creating custom signals with the Signal Builder tool, enabling the design of complex, scenario-specific input profiles (e.g., train B's position sequence).

Outputs from the referenced model are monitored using Scope blocks, which visualize key system states in real time. In this test harness, the following signals are displayed:

- Signal states (track free, warning or non-crossable stop)
- Area occupancy
- Train A's position and driving mode
- Dwell time counter (elapsed time spent by Train A in its current area).

Figure 112 and Figure 113 give examples of output for the scenario of Figure 110: when Train A crosses the signal X, which is in the state "Warning", it switches to signal-targeting mode (after 14 seconds of simulation). The signal Y, which is in the state "non crossable stop" is not crossed by Train A.

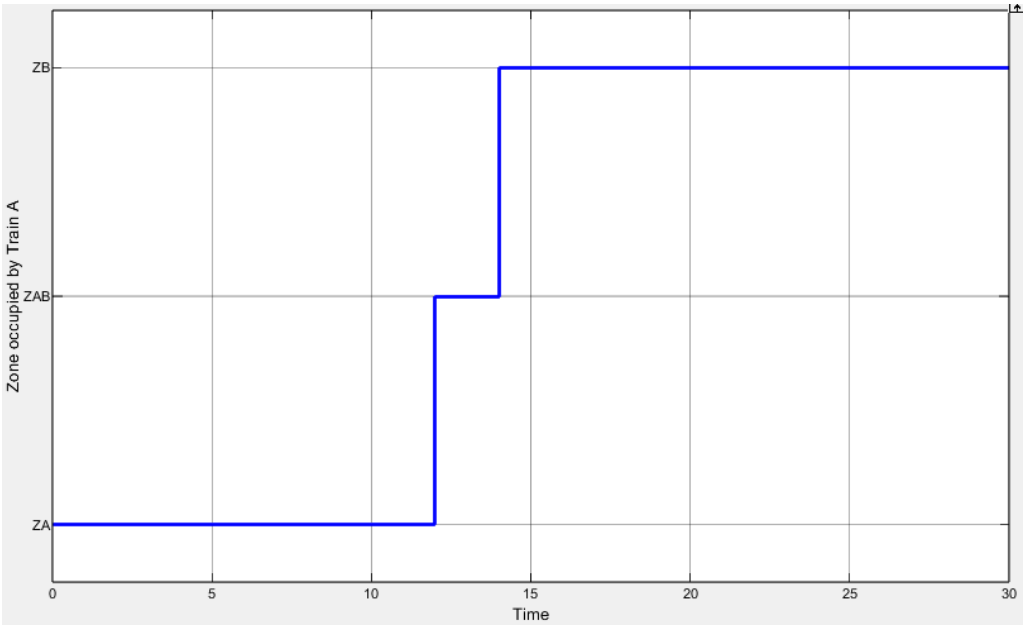


Figure 112: Zone occupied by Train A

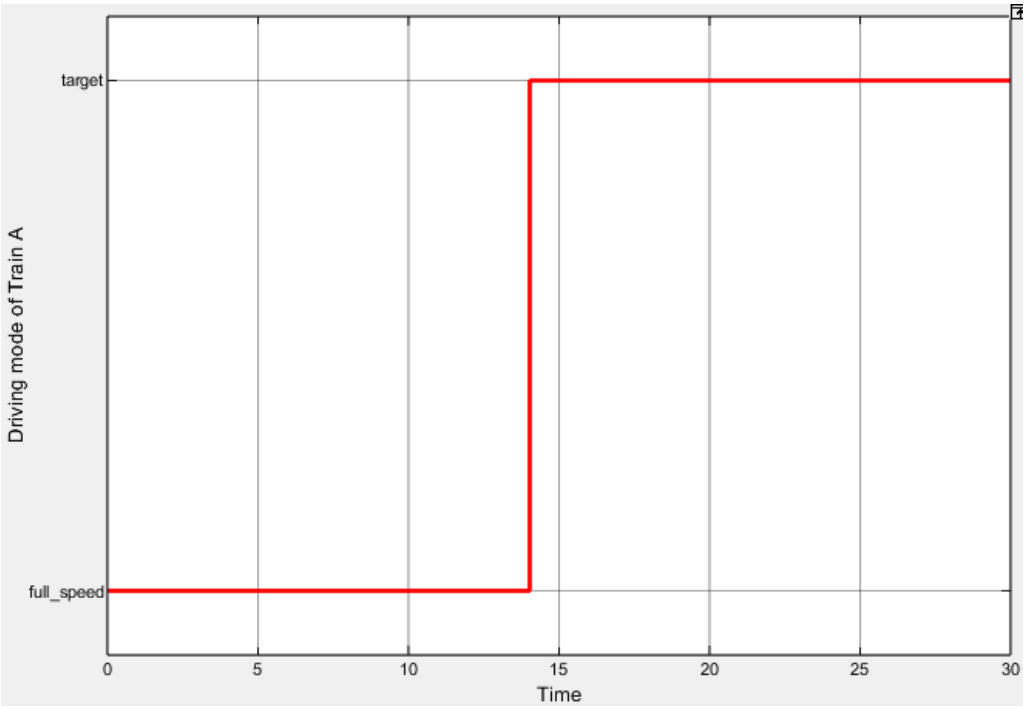


Figure 113: Driving mode of Train A

This straightforward test harness structure was used for unit testing during the development of the GoA4 model presented in section 4.2, where iterative testing and incremental validation were essential.

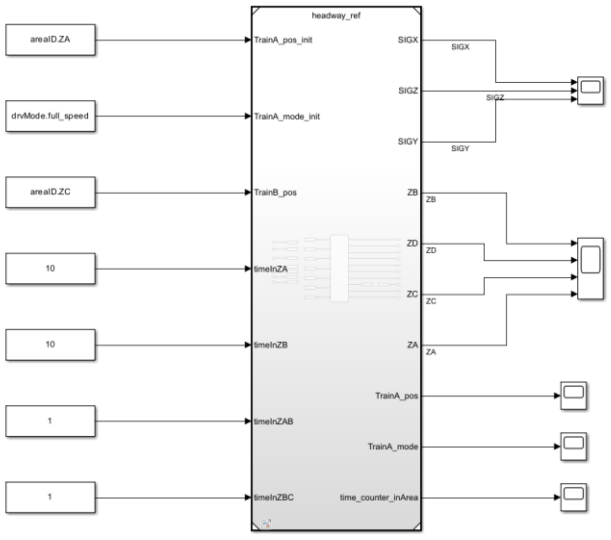


Figure 114: Test harness of the model

4.4.3 Variant model

A variant model enables dynamic activation of different submodels based on a control variable, providing flexibility in system behaviour without restructuring the entire architecture. This approach is particularly useful in testing environments, where specific models can be activated or deactivated

depending on the test case—all controlled programmatically via a script. For example, in the implementation presented on Fig X, the variant model is governed by the scenario variable, which determines the input values of the referenced model. By switching the scenario value, users can easily test different configurations, input sets, or system responses, streamlining validation and reducing manual intervention.

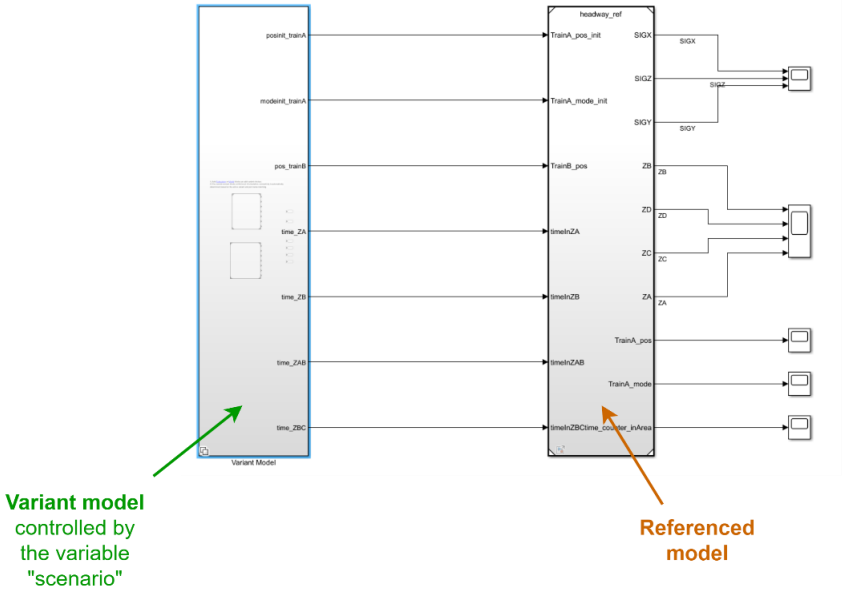


Figure 115: Use of a variant model to provide inputs to the referenced model

4.4.4 Test coverage

Test coverage measures the proportion of model components and decision paths “activated” by existing tests. If test coverage is incomplete, it may indicate the need to:

- Adjust existing tests or create new ones to address untested areas.
- Refine the model, as some features may exceed the intended scope of the requirements.
- Update the requirements if they are incomplete or misaligned with the system under design.

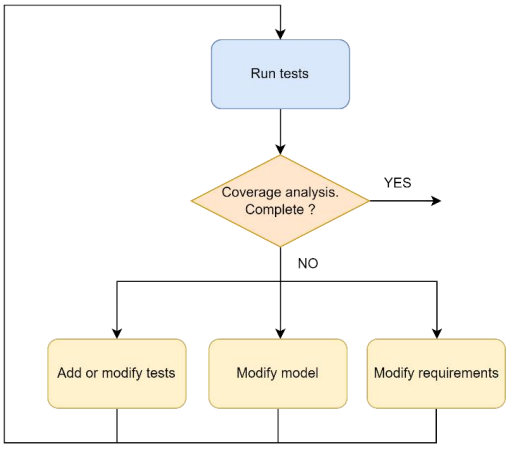


Figure 116: Test coverage analysis process

Simulink provides a dedicated tool for test coverage analysis. This tool computes a coverage percentage for each software component and shows which algorithmic paths are exercised by the tests.

As an example, we first analysed the coverage obtained with the single test scenario shown in Figure 110; a summary of this analysis is provided in the left-hand side of Figure 118. The paths activated by this scenario are coloured in green, for the Stateflow chart controlling area occupancy, in the first image of Figure 117. A second test was then added, in which train B is no longer stationary in area C but, after a certain time, moves to area D, allowing train A to pass signal Y and then enter area C. As shown in Figure 117, this additional scenario activates previously untested transitions and significantly increases the global coverage of the Stateflow model.

Simulink also enables automatic generation of test cases to improve test coverage.

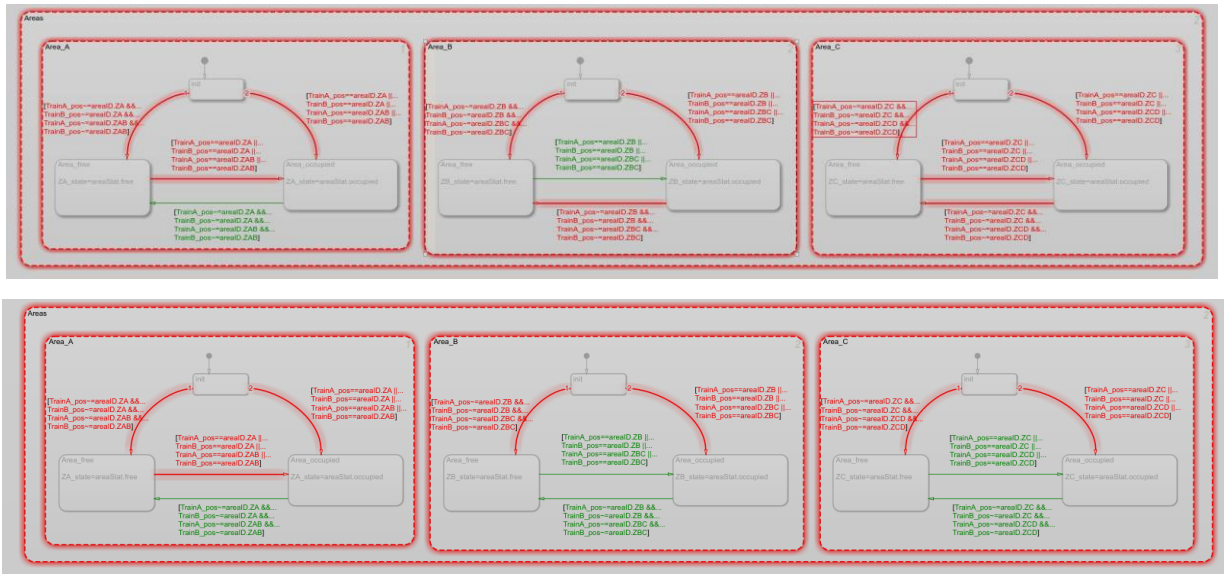


Figure 117: Test coverage visualization of area occupancy diagram with a single test scenario (first picture) and with two test scenarios (second picture)

Summary

Model Hierarchy:	Total Decision	Execution
1. headway_ref	60%	NA
2. . . . Chart	60%	NA
3. . . . . SF: Chart	60%	NA
4. . . . . SF: Train_movement	59%	NA
5. . . . . SF: Areas	61%	NA
6. . . . . SF: Area_A	73%	NA
7. . . . . SF: Area_B	64%	NA
8. . . . . SF: Area_C	45%	NA
9. . . . . SF: SIG	40%	NA
10. . . . . SF: SIGX	30%	NA
11. . . . . SF: SIGY	50%	NA
12. . . . . SF: TrainA	74%	NA
13. . . . . SF: TA_mode	100%	NA
14. . . . . SF: TA_pos	63%	NA

Summary

Model Hierarchy:	Total Decision	Execution
1. headway_ref	77%	NA
2. . . . Chart	77%	NA
3. . . . . SF: Chart	77%	NA
4. . . . . SF: Train_movement	76%	NA
5. . . . . SF: Areas	76%	NA
6. . . . . SF: Area_A	73%	NA
7. . . . . SF: Area_B	73%	NA
8. . . . . SF: Area_C	82%	NA
9. . . . . SF: SIG	55%	NA
10. . . . . SF: SIGX	30%	NA
11. . . . . SF: SIGY	80%	NA
12. . . . . SF: TrainA	96%	NA
13. . . . . SF: TA_mode	100%	NA
14. . . . . SF: TA_pos	94%	NA

Figure 118 : Test coverage summary with a single test scenario (first picture) and with two test scenarios (second picture)

4.4.5 Proof

The Design Verifier toolbox makes it possible to define properties that the model is expected to satisfy. In Simulink, we specify assumptions on the input data, such as the finite or infinite sets in which the inputs may take their values, and the conditions they must meet. For example, in our case study, we can take 2 assumptions :

- **Assumption 1**: Train A is initially in Area A
- **Assumption 2**: Train B can only move forward from one area to the next (from area B to area BC, then to area C, and so on).

We then define the assertions that our design must satisfy. In the example of the simplified signalling system, the assertion to be verified is:

- **Assertion 1**: Train A and Train B must not be in the same area.

Once Design Verifier has completed its analysis, if an assertion is violated, it provides an input dataset that satisfies all the specified assumptions but contradicts the property. This counterexample allows us to analyse in detail how the undesired behaviour occurs, and then decide whether to modify the model design, the input assumptions, or the assertions themselves.

For instance, the previously stated assumptions on train A and B position were insufficient to ensure that Train A and Train B are in different areas. The counterexample given in Figure 119 demonstrates that the two trains can both start in area A, which falsifies the assertion. This led us to add a third assumption:

- **Assumption 3**: Train B must not be in Area A.

When running the Design Verifier analysis again, the assertion is no longer falsified.

Counterexample

Time	**0**
Step	**1**
TrainA_pos_init	areaID.ZA
TrainA_mode_init	drvMode.full_speed
TrainB_pos	areaID.ZA
timeInZA	30
timeInZB	30
timeInZAB	5
timeInZBC	5

Figure 119: Counterexample provided by Design Verifier analysis

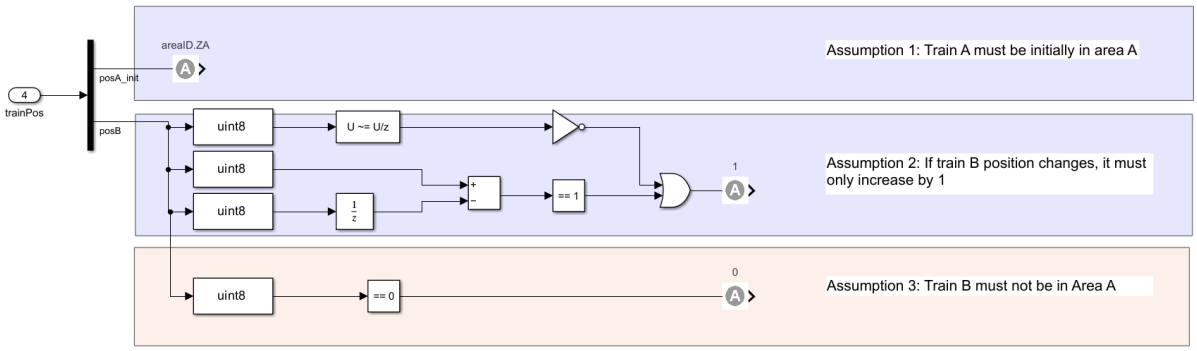


Figure 120: Definition of assumptions in Simulink

By systematically identifying edge cases, formal assertion verification strengthens the robustness of system development—ensuring alignment between specifications and implementation.

5 CONCLUSION

The work conducted in Task 31.1, through the implementation of various aspects of the Model-Based Design (MBD) approach, has successfully demonstrated and illustrated its benefits across the entire development cycle, particularly in the context of ATO use cases.

By applying MBD principles, the study highlighted how early-stage error detection in system specifications reduces risks and avoids costly revisions in later phases. The integration of validation tests further streamlined subsequent integration and testing activities, significantly improving project efficiency. The study conducted underscores that the full potential of modelling is realized when complemented by structured best practices. These include a thorough upfront analysis of system objectives to define the required level of model representativity, carefully balancing trade-offs between development time, computational performance, and representativity of the model. In this context, different modelling strategies can be considered, ranging from simplified analytical representations to more detailed multiphysics models, and the choice of model is closely linked to the targeted level of physical integration, from purely virtual simulations up to Processor-in-the-Loop (PIL) and Hardware-in-the-Loop (HIL) configurations. Furthermore, embracing modular design principles is crucial to promote reusability across projects, reducing redundant efforts. Effective team organization aligned with modelling workflows ensures collaboration and consistency in model-based activities. This holistic approach ensures that modelling not only accelerates development but also improves system reliability and maintainability over its lifecycle.

A model for ATO GoA2 was developed, focusing on a commercial mission use case. A key feature of this model is a speed profile optimization algorithm, designed to minimize the train's energy consumption while strictly adhering to the timetable and ensuring high stopping accuracy. This work illustrates a dual purpose of modelling in early development phases. On the one hand, it provides a preliminary techno-economic analysis tool for the implementation of ATO GoA2, by estimating the energetic gain. The validity of the optimization model was confirmed by the close alignment between its energy-optimal speed profile and the profile recorded by the most energy-efficient human driver on the same mission, demonstrating its practical effectiveness. On the other hand, the modelling task acts as a validation work for the specifications outlined in the subsets. Specifically, the model was used to scrutinize the functional requirements of the ATO-OB (Subset-125) and its interfaces with other systems (Subset-126, Subset-130, and Subset-139).

A second model was developed for ATO GoA4, targeting a technical movement use case between a yard and a passenger station, on which a lateral signalling perception algorithm was implemented. A modular validation process was applied to this model, enabling progressive verification and integration of its components and leading ultimately to the validation of the test scenarios defined at the beginning of the project. The presence of AI-based algorithms for the perception of trackside signals required the use of specific validation processes, tailored to address issues such as robustness and generalisation, and to ensure that the perception chain could be trusted under a wide range of operating conditions.

Multiphysics modelling tools were illustrated using the example of freight trains. Freight trains embody the challenges of complex systems: their dynamics involve coupled interactions across physical domains and highly nonlinear behaviours that make analytical modelling very challenging. Through this application, we illustrated how multiphysics modelling provide a more realistic and actionable representation of system dynamics, critical for performance evaluation and design optimization. While the developed models are exemplary rather than exhaustive, they reveal the

complexity of the dynamic behaviour of freight trains and the computational performance trade-offs that arise when striving for high fidelity. Further dedicated studies are needed to fully address freight train modelling.

Finally, dedicated testing and validation tools were presented. These tools help ensure and maintain the reliability of both specifications and developments throughout the entire development cycle, by enabling systematic verification against requirements and by supporting the continuous alignment of models, implementations, and tests. They also contribute to preserving a unified and consistent view of the system among all stakeholders, thereby facilitating communication and traceability.

Looking ahead, several perspectives emerge from this work. First, the representativity of the models could be selectively increased where required by specific use cases, for example by enriching the physical detail of certain subsystems and by aligning the models more closely with the architectures and interfaces currently under development. Second, on the ATO topic, a change of scope from single-train use cases to a more global view of ATO as a system-of-systems appears particularly promising. Multi-agent models involving several trains, and traffic management entities, would make it possible to analyse capacity impacts and network-wide energy performance, and to support the design and validation of future ATO concepts at line or network scale.

Beyond the modelling examples presented and the demonstrated gains, the work carried out paves the way for a new collaborative approach between railway stakeholders. Traditional specifications could, in some cases, be supplemented or even replaced by models provided by the operator, offering a more precise and dynamic representation of the expected system behaviour. This shift would enhance mutual understanding and reduce ambiguities in requirements. Additionally, operators could provide a comprehensive set of validation tests, enabling manufacturers to systematically verify their development. This model-centric collaboration fosters greater alignment, efficiency, and trust throughout the development process.

REFERENCES

[1] J. T. Haahr, D. Pisinger et M. Sabbaghian. “A dynamic programming approach for optimizing train speed profiles with speed restrictions and passage points”. In: Transportation Research Part B: Methodological 99 (2017), p. 167-182. doi: https://doi.org/10.1016/j.trb.2016.12.016.

[2] L.S. Pontryagin et al. The Mathematical Theory of Optimal Processes. Wiley Interscience, 1962.

[3] S. Irnich et G. Desaulniers. “Shortest Path Problems with Resource Constraints”. In: Boston, MA: Springer US, 2006, p. 33-65. doi: http://dx.doi.org/10.1007/0-387-25486-2_2.

[4] S. Su, Z. Tian et R.M.P. Goverde. Energy-Efficient Train Operation. Springer Cham, 2023.

[5] Braulio Aguiar, Denis Berdjag, Bernard Demaya, Thierry-Marie Guerra, A robust and fault tolerant approach for automatic train stop control system design, IFAC-PapersOnLine, Volume 50, Issue 1, 2017

[6] Valerio De Martinis, Mariano Gallo, Models and Methods to Optimise Train Speed Profiles with and without Energy Recovery Systems: A Suburban Test Case, Procedia - Social and Behavioural Sciences, Volume 87, 2013

–[7] Miyatake, Masafumi & Member, Hideyoshi. (2010). Optimization of Train Speed Profile for Minimum Energy Consumption. IEEJ Transactions on Electrical and Electronic Engineering. 5. 263 - 269. 10.1002/tee.20528.

–[8] Domínguez, María & Cucala, Asunción & Fernández, Antonio & Pecharromán, Ramón & Blanquer, J. (2011). Energy efficiency on train control: Design of metro ATO driving and impact of energy accumulation devices

–[9] Zhu, Xiaomin, Qian Pu, Qian Zhang and Runtong Zhang. “Automatic Train Operation Speed Profile Optimization and Tracking with Multi-Objective in Urban Railway.” Periodica Polytechnica Transportation Engineering (2019): n. pag.

–[10] Achilleos, Achilleas & Anastasopoulos, Markos & Tzanakaki, Anna & Iordache, Marius & Langlois, Olivier & Pheulpin, Jean-Francois & Simeonidou, Dimitra. (2019). Optimal Driving Profiles in Railway Systems based on Data Envelopment Analysis. 254-259. 10.5220/0007878000002179.

APPENDIX

APPENDIX 1 : PER BLOCK DIAGRAM

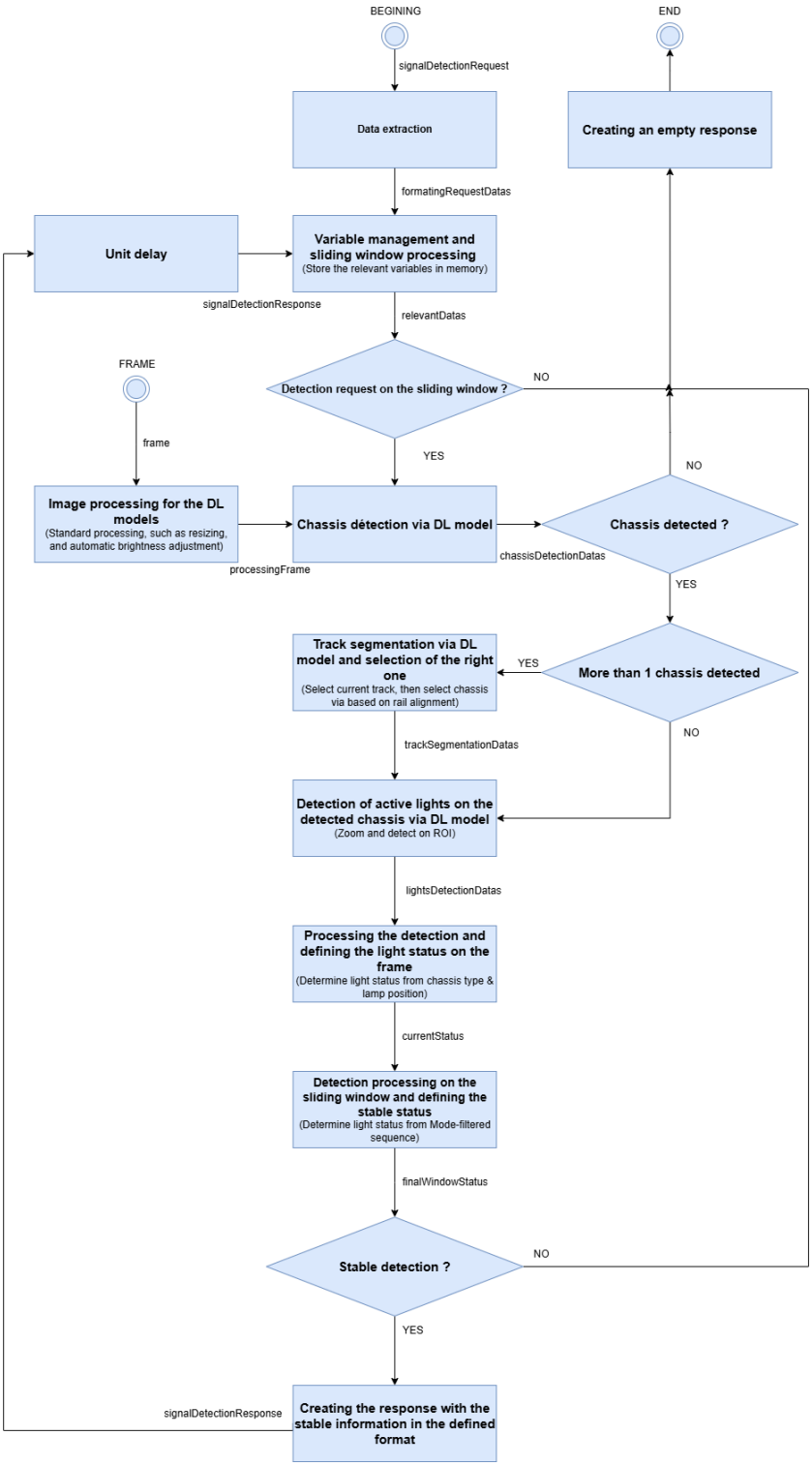


Figure 121 : Bloc diagram of PER module