Architectural Variants of ETCS-Based Trackside Systems in a Unified Ontology-Driven Validation Framework

Arne Borälv[1][0009-0008-9217-074X] and Daniel Schwencke[2][0000-0002-0592-9551]

[1] Prover Technology AB, Krukmakargatan 21, 118 51 Stockholm, Sweden
[2] German Aerospace Center, Institute of Transportation Systems, Lilienthalplatz 7, 38108 Braunschweig, Germany
arne.boralv@prover.com

Abstract. Modern ETCS-based railway systems are evolving towards increasingly distributed and heterogeneous architectures, driven by moving-block operation, hybrid train detection, and flexible allocation of functionality across subsystems. In parallel, ongoing European initiatives seek to standardise system behaviour and interfaces across implementations. These developments create a need for modelling approaches that support systematic reasoning about architectural alternatives. Conventional modelling approaches, however, often encode architectural assumptions directly in the model structure, limiting reuse and complicating systematic comparison of design alternatives.

This paper builds on an ontology-driven modelling framework in which object-model interfaces are generated from a layered ontology, to provide a uniform basis for requirements, behavioural models, and validation artefacts. At the core of this ontology, a general train-centric abstraction serves as an architecture-independent semantic layer, characterised by lifecycle and consistency invariants and realised as a stable bundle of types capturing train representation and state, train location, authority, and interaction with infrastructure.

Architectural variability is expressed through constraint-based refinement of this abstraction. Rather than restructuring the model, variants are obtained by introducing or modifying constraints on the admissible construction and evolution of core concepts. We illustrate this using representative ETCS-based architectures, including traditional route, hybrid, and dynamic route concepts.
A key result is that the modelling structure, lifecycle semantics, and validation mechanisms remain unchanged across variants. This enables reuse of executable test generation and formal verification artefacts and supports consistent analysis of architectural alternatives within a single framework.

Keywords: Validation Framework, Architectural Variants, ETCS, Train-Centric Modelling, Constraint-Based Refinement, Ontology-Driven Modelling.

1 Introduction

ETCS-based railway signalling systems are evolving towards increasingly distributed and heterogeneous architectures. Concepts such as moving-block operation [1], hybrid train detection [2] and flexible allocation of functionality across subsystems [3] challenge established assumptions about how movement authority, train separation, and infrastructure constraints – such as routes, block sections, or restricted path segments – are represented. In parallel, ongoing European initiatives seek to standardise system behaviour and interfaces across implementations. These developments create a need for modelling approaches that support systematic reasoning about architectural alternatives.

Another axis of architectural variation concerns how train position is determined. Modern systems such as ETCS Level 2 and CBTC increasingly rely on train-based position reporting relative to known reference points in the infrastructure (e.g. balises), communicated via radio to trackside control components such as an RBC or Zone Controller. At the same time, traditional trackside detection systems such as track circuits or axle counters are often retained to support fallback operation or additional supervision. These choices influence how movement authority is issued and how infrastructure constraints are enforced.

This architectural variability creates challenges for modelling. Many modelling approaches embed architectural assumptions directly in the model structure, so that exploring alternative architectures typically requires restructuring the model itself, limiting reuse of behavioural descriptions, test cases, and verification artefacts.

In addition, ETCS Level 2 system architectures typically exhibit tight coupling between infrastructure-level route management (in an interlocking) and train-specific authority control (in an RBC). For example, route establishment and release depend on ongoing train movements, and movement authority in turn depends on the availability of routes. While such coupling is necessary for safe operation, it is often realised through fixed architectural decompositions and interfaces, which can make it more difficult to explore and compare alternative architectural designs.

Efforts to evolve railway system architectures have been pursued in European initiatives such as the Reference CCS Architecture (RCA), which aimed to promote modularity, standardised interfaces, and clearer allocation of responsibilities across system components. These principles are continued and expanded in the Europe's Rail System Pillar (SP), reflecting a broader trend towards architectural decoupling and increased flexibility in system design. Recent work within SP, particularly within the Traffic Control and Supervision (Traffic CS) domain, reflects a similar shift towards train-centric and implementation-agnostic system design. Notably, concepts such as dynamic train-centric control, movement permissions, and associated reserved paths aim to decouple authority management from fixed infrastructure segmentation.

Together, these developments motivate modelling approaches that separate semantic abstraction from architectural realisation, allowing different allocations of responsibilities and interaction patterns to be expressed within a common framework. This perspective aligns with principles of distributed system design, where functionality is decomposed across interacting components with clearly defined interfaces.

This paper builds on the ontology-driven modelling and validation framework developed in the Europe's Rail Joint Undertaking project FP2-R2DATO [4]. While that work introduced a general train-centric abstraction, the systematic representation of architectural variants within a single modelling basis remained open.

The main contribution of this paper is to show that architectural variability can be expressed as constraint-based refinement over a shared semantic core. Rather than redefining the model for each architecture, variants are obtained by introducing or modifying constraints over a stable abstraction.

2 Background: Ontology-Driven Train-Centric Modelling

2.1 Ontology-Driven Modelling Framework

The framework is based on a layered ontology separating domain-independent modelling concepts from railway-specific abstractions. The upper ontology defines general notions such as objects, types, attributes, relations, and lifecycle concepts, together with structuring mechanisms such as inheritance, specialisation, and composition. It also includes generic path abstractions, where the most general path type allows paths to begin and end at arbitrary positions in the topology, independent of predefined infrastructure segmentation. The lower ontology introduces railway-specific concepts relevant to ETCS-based trackside systems, such as trains, train locations, and authorities, building on these general abstractions.

From the combined ontology, object-model interfaces are generated mechanically. These interfaces expose a simplified operational view consisting of typed objects and relations, while internalising structural complexity, as required for modularity and reuse. The result is a stable interface for modelling and analysis.

This interface forms a common semantic layer across requirements, behavioural models, test cases, and formal properties, reducing reinterpretation between artefacts and supporting consistency across validation tasks.

2.2 Train-Centric Abstraction

At the domain level, the framework adopts a train-centric abstraction. Rather than modelling signalling logic primarily in terms of infrastructure states, it focuses on train-related state and its relation to infrastructure constraints.

The abstraction models authorities to move as path-based objects that are created, extended, reduced, and removed according to defined lifecycle rules. This makes the dynamic evolution of authority to move explicit, rather than treating it as an implicit property of the system state. Furthermore, this paths-based abstraction avoids dependence on fixed, pre-defined block sections characteristic of traditional signalling. This enables modelling at a level that is independent of specific infrastructure segmentation and supports more flexible architectural interpretations.

The abstraction is naturally compatible with the expression of additional, orthogonal concepts within the same modelling framework. These include domain-specific constraints such as forbidden paths, lifecycle transitions related to train progression and

release of resources, as well as more complex operational scenarios such as joining and splitting of trains. Such concepts can be expressed uniformly over the same path-based representation without introducing additional structural elements.

2.3 Validation Approach

Validation combines executable and formal techniques. Infrastructure data, for example in RDF form, are translated into an internal representation and used to instantiate the object model. Executable validation is then performed through automatically generated movement scenarios.

In parallel, formal verification is applied to selected classes of properties, including typing constraints derived from the ontology, lifecycle invariants, and domain-specific safety conditions such as authority consistency and forbidden-path constraints.

The framework also supports explicit representation of non-nominal situations through dedicated non-nominal (NN) objects, enabling analysis of fault scenarios and degraded behaviour within the same modelling framework.

2.4 Scope in this Paper

The framework described above establishes a general abstraction and validation approach but does not prescribe a specific railway system architecture. In particular, it does not enforce route-based constraints, specific train-detection schemes, or particular allocations of functionality across subsystems.

This paper builds on that generality and investigates how architectural variants can be represented as constraints over the same abstraction.

3 Train-Centric Abstraction and Refinement

3.1 Core Abstraction as a Bundle of Types

The abstraction builds on the generic path abstractions introduced in the ontology and extends them with additional domain-specific concepts. It is defined as a stable bundle of the following core types:

- Central Unit Train (CUT)
- Train Location Path (TLP)
- Authority Base (AB)
- Authority Path (AP)
- Swept Path (SWEPT)

These types, together with their lifecycle relations and invariants, form an architecture-independent semantic core intended to remain stable under architectural change.

CUT represents a train managed by the system, including its associated state. TLP represents the train's current location as known to the system (i.e. the available knowledge of its position). AB bounds the maximal extent of authority that may be granted to the train, while AP represents the authority to move that has been granted.

SWEPT represents the path confirmed as traversed by the train, derived from successive updates of train location (e.g., successive train position reports).

A key design principle is the separation of concerns between these elements: the representation of the managed train and its state, location knowledge, authority bounds, granted authority, and historical progression are modelled independently. This allows architectural concepts such as routes, blocks, or subsystem boundaries to be expressed as constraints rather than being embedded in the core model.

Authority Bases (AB) are treated as runtime objects local to the RBC. External components, such as a separate interlocking, do not create ABs directly but may provide constraints (e.g. ROUTE objects) that restrict how ABs are constructed and evolved.

Authority Paths (AP) are typed (e.g. movement authority, staff responsible, reversing) and oriented. Forward and reversing movements are treated uniformly using the same path-based concepts, differing only in orientation and alignment relative to the TLP. This avoids introducing special-case constructs for reversing behaviour and contributes to a compact abstraction.

The relationships between these concepts are illustrated in Fig. 1.

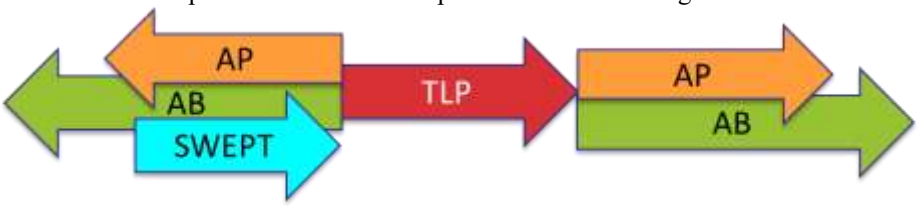

Fig. 1. Conceptual relationship between paths in the train-centric abstraction.

3.2 Fundamental Invariants and Domain Constraints

Building on the core abstraction defined in Section 3.1, additional constraints define admissible behaviour and capture domain-specific requirements.

The abstraction is accompanied by a set of invariants that characterise nominal system behaviour. These define simple normative principles rather than assumptions that must hold in all operational situations.

Structural Invariants. In nominal situations, any Authority Path (AP) is required to be a prefix path of a corresponding Authority Base (AB), i.e. an initial contiguous segment of AB. This ensures that granted authority extends continuously from the train's current position along an admissible path.

Where a Train Location Path (TLP) is available, alignment constraints apply uniformly to both AB and AP. For forward movement, the rear of the path coincides with the front of the TLP. For reversing movement, the rear of the path coincides with the rear of the TLP and the path has opposite orientation. This shared alignment principle ensures that both Authority Bases and Authority Paths are consistently anchored in the train's current position.

Global consistency applies across trains: Authority Bases associated with different CUTs must not overlap in nominal operation.

Lifecycle Constraints. Lifecycle constraints define a dependency ordering between dynamic objects:

CUT → TLP → AB → AP

This ensures that authority is always grounded in a managed train and, where available, a known train location. Creation and deletion of objects must respect this ordering, preserving well-formed system states throughout system evolution.

Domain Constraints. In addition to structural invariants, the abstraction supports domain-specific constraints, expressed uniformly using the same path-based concepts and largely independent of architectural variation.

(a) Exclusion Constraints. A representative example is the notion of an End-of-Authority Exclusion Area (EoA-EA). Such areas represent regions of the topology that must not be occupied by a train once its authority has been fully consumed. Informally, when a train reaches the end of its Authority Path (AP), its Train Location Path (TLP) shall not overlap any region classified as EoA-EA. These areas may correspond to protected infrastructure such as points or crossings; see Fig. 2 for an example.

Another important class of exclusion constraints is given by forbidden paths. A forbidden path represents a section of track that is explicitly disallowed for train movement, either in one direction or in both directions. Forbidden paths provide a declarative way to express safety constraints that would otherwise be embedded in route logic or procedural rules.

Typical use cases include:

- flank protection, preventing conflicting movements adjacent to an authorised train path;
- handover protection, restricting movement toward a boundary during coordination with adjacent control areas;
- operational restrictions, such as temporary or permanent prohibitions.

The path-based formulation allows such constraints to be expressed uniformly within the train-centric abstraction, without introducing additional structural concepts.

(b) Conditional Constraints. In contrast to purely prohibitive constraints, the abstraction also supports cases where overlapping paths are permitted but require a corresponding behavioural adjustment.

A representative example is the interaction between an Authority Path (AP) and a Temporary Speed Restriction (TSR), or an area of unknown state; see Fig. 2. In such cases, the AP may overlap the constrained region, but the authority must be associated with an appropriate operating mode, such as On-sight, to ensure that the train proceeds under restricted conditions.

This illustrates that path interactions are not limited to purely allowed or forbidden cases, but may encode conditional behaviour, where the presence of certain constraints influences how movement authority is interpreted and executed.

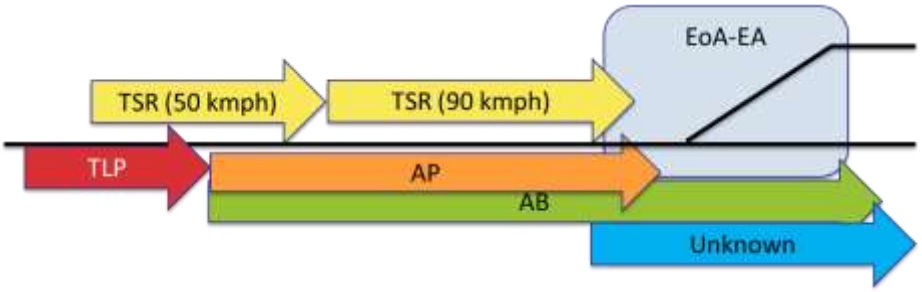

Fig. 2. Path-based authority with example domain constraints: conditional overlap (TSR, unknown path, EoA-EA).

Non-Nominal Situations. Violations of invariants or domain constraints are represented explicitly using non-nominal (NN) objects. Rather than excluding such situations, the framework records and analyses them, enabling systematic reasoning about degraded and exceptional behaviour.

For example, a violation of a forbidden path constraint detected at runtime results in the creation of a corresponding NN object, which persists until the underlying cause has been resolved. This mechanism separates the definition of constraints (what is not allowed) from the handling of their violation.

Intermediate Summary. These examples illustrate that the abstraction supports both prohibitive and conditional constraints on movement authority within a uniform path-based framework, while providing explicit mechanisms for representing and analysing deviations from nominal behaviour.

3.3 Constraint-Based Architectural Refinement

Architectural variability is expressed by introducing or modifying constraints over the abstraction defined above. A variant is obtained by adding or strengthening constraints to restrict admissible behaviour, or removing or relaxing constraints to allow additional behaviour. Typical constraints govern:

- the admissible extent of Authority Bases (e.g. requiring AB to be a subpath of a ROUTE),
- the construction and evolution of authority (e.g. conditions for creating or extending AB and issuing AP),
- exclusivity conditions across trains or infrastructure resources, and
- the allocation of responsibilities between components.

These constraints restrict the set of admissible system states and behaviours defined by the abstraction. Importantly, the bundle of types (CUT, TLP, AB, AP, SWEPT), their lifecycle relations, and handling of non-nominal situations remain unchanged.

The abstraction can therefore be understood as a maximal semantic module, with architectures defined as constrained instantiations of this module. This separation

between semantic abstraction and architectural constraints enables systematic comparison of variants and reuse of validation artefacts across them.

Fig. 3 illustrates how a stable bundle of core types (CUT, TLP, AB, AP, SWEPT), together with lifecycle and nominal invariants, defines a common semantic basis. Architectural variants are obtained by instantiating the core abstraction and introducing or relaxing constraints over this core, rather than by restructuring the model.

For illustration, we refer to representative architectural variants based on ETCS Level 2: M2 denotes a moving-block configuration, F2 a fixed-block (route-based) configuration, and H2 a hybrid train detection configuration. We also refer to a dynamic variant (D2) to denote architectures based on dynamically constructed route concepts.

These variants are discussed in Sections 4.1-4.4.

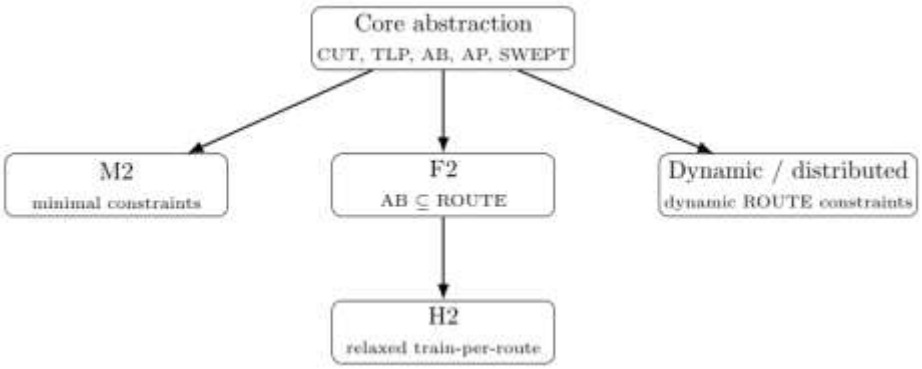

Fig. 3. Constraint-based architectural refinement over the train-centric core abstraction.

4 Architectural Variants as Constraint Refinements

This section illustrates how different architectural configurations of ETCS-based systems can be expressed as constraint refinements over the common abstraction defined in Section 3. Rather than introducing separate models for each architecture, variants are obtained by imposing additional constraints that restrict the admissible construction and evolution of authority and infrastructure usage within the abstraction.

4.1 Baseline: Unconstrained Authority Model (M2)

The baseline model represents the least constrained instantiation of the abstraction. Authority Bases (AB) are arbitrary paths anchored in the Train Location Path (TLP) of a Central Unit Train (CUT), and Authority Paths (AP) are issued as prefix paths of AB in accordance with the core invariants.

No architectural constraints are imposed beyond those defined in Section 3. In particular, AB is not restricted by predefined infrastructure constructs such as routes or blocks, and no additional exclusivity constraints apply beyond non-overlap of AB across CUTs.

This baseline can be interpreted as a generalised moving-block model and serves as the reference point for all subsequent variants.

4.2 Route-Constrained Authority (F2)

This variant introduces infrastructure-level constraints through the notion of routes. A separate interlocking component defines ROUTE objects representing admissible infrastructure paths together with compatibility constraints. The interlocking may ensure that conflicting ROUTEs—such as overlapping or incompatible paths—are not active simultaneously.

The RBC constructs Authority Bases (AB) subject to these constraints, typically requiring that AB is a subpath of an active ROUTE (AB ⊆ ROUTE), see Fig. 4. This may be combined with exclusivity constraints, for example requiring that the ROUTE is not currently associated with another train: i.e., that it does not overlap with paths such as TLP, AB, or AP belonging to another train.

This variant captures the essential characteristics of classical route-based ETCS Level 2 operation.

Importantly, the route concept is introduced as a constraint on the construction and extent of AB paths, rather than as a structural element of the abstraction.

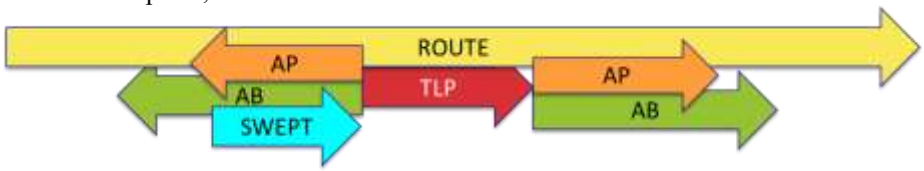

Fig. 4. Route-constrained authority model (F2): interlocking-provided ROUTE constraints bound RBC authority construction.

4.3 Hybrid / Relaxed Constraints (H2)

Starting from the route-constrained variant (F2), this model relaxes RBC -imposed restrictions on the number of trains associated with a given route.

Authority Bases remain constrained by ROUTE objects, but multiple CUTs may be associated with the same route, provided that their respective Authority Paths remain consistent with the core invariants.

This enables more flexible utilisation of infrastructure and captures key aspects of hybrid train detection, where separation is not enforced solely through exclusive route allocation but through a combination of constraints on authority and train state.

4.4 Dynamic Routes and Distributed Authority (D2)

Emerging architectural directions explore the generalisation of routes from static infrastructure elements to dynamically evolving constructs. In such approaches, routes are not predefined objects but are constructed, extended, and reduced during operation based on local infrastructure conditions and control logic.

One such direction, as explored in Task 30.2 of the FP2-R2DATO project, can be interpreted as follows. In this approach, dynamic routes are characterised implicitly by their endpoints together with the requirement that all intermediate drive protection sections are traversable and appropriately locked.

Drive protection sections represent regions of the topology associated with field elements such as points or crossings. Each section may either satisfy the conditions required for train movement or not. In addition, sections that are used by a route are reserved (locked), preventing conflicting use by other routes. Within the present framework, such dynamic route concepts can be interpreted as constraints over the train-centric abstraction. In particular, a dynamic route may be viewed as defining an admissible path interval whose extent is governed by infrastructure-level conditions. This can be represented by a ROUTE object that constrains the admissible extent of Authority Bases (AB), such that AB remains within the dynamically constructed ROUTE.

A particularly relevant interpretation arises when considering interaction between an interlocking component and a train-centric authority management system such as an RBC. In this setting, the interlocking may internally construct dynamic routes based on its own modelling concepts, for example as compositions of adjacent drive protection sections. These internal representations need not be exposed directly. Instead, the interlocking can communicate the resulting route as a path-based constraint through a ROUTE object.

The RBC, in turn, constructs Authority Bases (AB) and Authority Paths (AP) within the train-centric abstraction, subject to the constraint that AB remains within the communicated ROUTE. Updates to the dynamic route, such as extension or reduction, can be reflected through corresponding updates to the ROUTE constraint.

Importantly, this interpretation does not prescribe a fixed allocation of responsibilities between components. Rather, it illustrates one possible realisation in which infrastructure-level route construction and train-centric authority management interact through a shared constraint representation. Alternative decompositions are possible, in which responsibilities and information flows between components differ, while still conforming to the same underlying abstraction. This perspective shifts the role of interfaces such as those between interlocking and RBC from fixed architectural boundaries to design choices that can be varied and analysed within a common modelling framework.

The communicated ROUTE may also reflect infrastructure-level protection measures established by the interlocking, such as flank protection. Rather than exposing the underlying mechanisms, such information can be conveyed as constraints on admissible movement. For example, protection measures may be represented in the RBC as corresponding forbidden paths, while the detailed protection logic and state can remain internal to the interlocking.

More complex operational scenarios, such as splitting and joining of trains, can be interpreted within the same framework. On the train-centric side, such operations involve the creation, transformation, and removal of objects such as TLP, AB, and AP to reflect the evolving set of trains and their associated authority. On the infrastructure side, corresponding adjustments may be required to route-related constraints, for example through splitting or merging of route representations within an interlocking.

These processes can be realised through different sequences of interactions between components, depending on the architectural decomposition and allocation of responsibilities. Coordination between train-centric and infrastructure-level representations may therefore involve the exchange of requests and updates concerning authority, route constraints, and train state, without prescribing a fixed direction of control or specific procedural steps. This illustrates that the abstraction supports dynamic operational scenarios involving multiple interacting components, while allowing flexibility in how such interactions are structured and implemented.

This interpretation enables a modular separation of concerns:

- the interlocking manages infrastructure-level protection and route construction,
- the RBC manages train-centric authority (AB/AP),
- and ROUTE acts as a semantic interface between these layers.

In this way, different modelling paradigms can co-exist within a single framework. The interlocking may employ a detailed infrastructure-oriented representation, while the RBC operates on a train-centric abstraction. The ROUTE concept provides a common representation that enables meaningful exchange of information without requiring alignment of internal modelling structures.

The framework thus accommodates dynamic and distributed control strategies without introducing new core concepts. Instead, such architectures are expressed through constraints over the same bundle of types, extending the applicability of the abstraction while preserving its simplicity.

4.5 Comparative Summary

The variants above differ only in the constraints imposed on a shared semantic core. Table 1 summarises the main cases.

Table 1. Comparative overview of the architectural variants.

Variant	Key constraint	Effect
Baseline (M2)	Minimal constraints beyond core invariants	Authority over arbitrary paths
Route-constrained (F2)	AB ⊆ ROUTE; conflicting ROUTEs excluded; often with exclusivity constraints (e.g. one train per route)	Classical route-based behaviour
Hybrid / relaxed (H2)	AB ⊆ ROUTE; relaxed train allocation policy	Multiple trains per route
Dynamic / distributed (D2)	Dynamic ROUTE constraints; distributed responsibilities	Dynamically constructed routes

Across all variants, the same bundle of types (CUT, TLP, AB, AP, SWEPT), lifecycle constraints, and non-nominal handling mechanisms are preserved. Architectural differences are expressed explicitly through constraints, enabling systematic comparison within a single modelling framework.

These variants differ primarily in where constraints are imposed, for example at the level of infrastructure allocation (e.g. routes), authority assignment, or dynamic coordination between components.

5 Implications for Validation

The constraint-based treatment of architectural variability has direct implications for validation. Because all variants share the same core abstraction—expressed as a stable bundle of types (CUT, TLP, AB, AP, SWEPT)—validation activities can be organised around this common semantic basis.

In particular, the structure of validation aligns naturally with the layers of constraints introduced in Section 3.2: structural invariants, lifecycle constraints, domain constraints, and non-nominal situations.

5.1 Validation of Structural Invariants

Structural invariants defined over the abstraction—such as the prefix relation between Authority Path (AP) and Authority Base (AB), alignment with Train Location Path (TLP), and non-overlap of Authority Bases—are independent of architectural variation.

In particular, safety properties expressed over the core abstraction remain applicable across variants, as they are defined independently of architectural decomposition.

As a result, formal verification of these properties can be defined once and reused across all variants, while additional properties may be required to capture variant-specific constraints. Test case generation based on movement scenarios can also exercise these invariants uniformly, since the underlying object model and path semantics remain unchanged.

This provides a stable foundation for validation, ensuring that core safety properties are preserved regardless of architectural choices.

5.2 Validation of Lifecycle Behaviour

Lifecycle constraints governing the creation and evolution of objects (CUT → TLP → AB → AP) define a consistent model of dynamic behaviour. These constraints are also invariant across architectural variants. They are largely ensured by construction within the framework (e.g. through typing and lifecycle rules), while still being subject to validation and explicitly representable in non-nominal situations.

Executable validation can therefore focus on scenario-based exploration of object lifecycles, including creation, extension, reduction, and removal of authority-related paths. Since the lifecycle structure is unchanged, such scenarios can be reused across variants without modification.

This supports systematic testing of dynamic behaviour while avoiding duplication of modelling effort.

5.3 Validation of Domain Constraints

Domain-specific constraints—such as exclusion areas (EoA-EA), forbidden paths, and conditional constraints (e.g. TSR or unknown areas)—are expressed using the same path-based concepts as the core abstraction, while being conceptually independent of it and largely independent of architectural variation. This allows validation of such constraints to be performed uniformly across different system configurations. For example:

- exclusion constraints can be checked as invariant violations,
- conditional constraints can be validated through scenario-based testing of operating modes,
- interactions between authority paths and constrained regions can be analysed consistently.

Because these constraints are expressed declaratively over paths, they can be integrated into both formal verification and executable validation without introducing architecture-specific logic.

5.4 Validation of Non-Nominal Situations

The explicit representation of non-nominal (NN) situations enables validation beyond nominal correctness. Violations of structural or domain constraints—such as forbidden path infringements or inconsistent authority assignments—are represented as NN objects and analysed within the same framework.

This supports validation of degraded modes of operation, failure scenarios, and operational overrides. Importantly, the mechanism for representing and handling non-nominal situations is independent of architectural variation, allowing such analyses to be reused across all variants.

5.5 Cross-Variant Analysis

Because architectural variants differ only through constraints, they can be instantiated and analysed within a single modelling environment. This enables systematic comparison of alternatives under consistent validation conditions.
In particular, it becomes possible to:

- evaluate how different constraint sets affect admissible system behaviour,
- compare safety properties across architectures,
- and explore trade-offs between flexibility and restriction.

This extends the validation approach developed in D30.3 and D30.4 from single-model validation to structured exploration of architectural design spaces.

5.6 Summary

The key consequence of the proposed approach is that validation artefacts—test scenarios, formal properties, and analysis methods—can be defined once over the core abstraction and reused across architectural variants.

By separating semantic abstraction from architectural constraints, the framework enables scalable, consistent, and reusable validation of evolving railway system designs.

6 Discussion and Future Work

The abstraction presented in this paper provides a foundation for representing evolving railway system architectures within a single semantic framework. By separating train-centric authority concepts from infrastructure-level constraints, it enables different architectural decompositions to be expressed and compared without modifying the underlying model structure.

This separation also highlights architectural design choices that are often implicit in current systems. In particular, the interaction between interlocking and RBC components can be interpreted as an exchange of constraints and authority information, rather than as fixed functional boundaries. This suggests a design space in which responsibilities and information flows between components can be varied while preserving a common semantic basis.

The interpretation of infrastructure-level constructs—such as routes, dynamic routes, or other constraint models—as path-based constraints within the abstraction provides a unifying perspective across different signalling paradigms. This supports integration of heterogeneous modelling approaches while maintaining a clear separation of concerns.

Several directions for further work arise from this perspective. These include a more explicit formalisation of constraint-based refinement, particularly with respect to preservation of safety properties, as well as a more detailed treatment of interfaces between components, including how constraints, authority, and state information are exchanged and kept consistent across interlocking and RBC boundaries. Another area concerns the systematic modelling of non-nominal situations and degraded modes, building on the explicit representation of such situations in the framework.

Overall, the proposed abstraction offers a flexible basis for reasoning about architectural variation in distributed railway systems, while leaving open a range of possibilities for further refinement and integration with more detailed models.

7 Conclusion

This paper has presented an ontology-driven modelling approach in which a train-centric abstraction, formulated as a stable bundle of core types, serves as a common semantic core across architectural variants of ETCS-based systems.

Architectural variability is expressed through constraint-based refinement over this abstraction, rather than through structural changes to the model. This allows route-based, hybrid, and dynamic/distributed architectures to be represented within a single framework while preserving modelling structure, lifecycle semantics, and validation mechanisms.

The approach also provides a basis for integrating infrastructure-level constraint models—such as route-based, dynamic route, or zone-based approaches—with train-

centric authority management through well-defined path-based constraint interfaces, without requiring alignment of their internal modelling concepts.

These results suggest that separating semantic abstraction from architectural constraints provides a scalable foundation for modelling and validating evolving distributed railway systems.

Acknowledgment and Disclaimer. This work has been conducted within the FP2-R2DATO project. The project is supported by the Europe's Rail Joint Undertaking (JU) and its members. It has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101102001. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or JU. Neither the European Union nor the JU can be held responsible for them.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. D4.1 – Moving Block Specification. X2Rail-5 project deliverable, version 12 (21 Dec 2022)
2. ERTMS Users Group: ERTMS/ETCS Hybrid Train Detection – Principles. Ref: 16E042, version 2 (11 Feb 2026)
3. D48.1 Autonomous Route Setting (AnRS) conceptual studies: use case list and concept definition. FP2-R2DATO project deliverable, version 8 (10 Feb 2025)
4. D30.4 – ERTMS L3 trackside with moving block formalisation. FP2-R2DATO project deliverable, version 11 (20 Apr 2026)