暨南大學
JINAN UNIVERSITY

本科生课程论文

Title: **Scenario-Based Modeling and Formal Verification in European Train Control System (ETCS) Level 2 and Level 3 A Case Study Approach**

Student Name: VAN SONGHIENG

Student ID Number: 2020059118

Major: International School (CST)

Course: Formal Methods in Software Engineering

Instructor: Qingliang Chen

Date: 2023-06-12

Formal Methods in Software Engineering: A Case Study

Abstract: The European Train Control System (ETCS) is a crucial component of modern railway operations. It is a part of the European Rail Traffic Management System (ERTMS), aimed at harmonizing national train control and command systems to facilitate cross-border interoperability. In an age of rising complexity and advancing automation, this study delves into the crucial role of formal verification methods in ETCS, emphasizing their importance in ensuring the safety, reliability, and efficiency of railway operations. The paper extensively investigates the use of these methods in ETCS, more specifically, in Automatic Train Operation Control System ETCS Level 2 and towards the design and verification of Level 3. The research reveals challenges and opportunities and provides key insights for the future evolution of ETCS.

Key word: ETCS, Formal Method, ATO, Railway system

Content

I. Introduction

The European Train Control System (ETCS) is an essential aspect of the European Rail Traffic Management System (ERTMS), which seeks to standardize train control systems across Europe to ensure smoother and more efficient cross-border rail transport. As a critical control system, ETCS requires high levels of safety and reliability, making formal verification methods essential.

Formal verification is a process used to prove or disprove the correctness of intended algorithms or systems as part of a hardware or software design. It uses formal methods of mathematics and logic to ensure that a system is behaving as expected under all possible conditions, contributing significantly to the system's overall safety and reliability.

In the context of ETCS, formal verification methods play a key role in assessing and validating system components and operations, primarily concerning automated train control systems. With the increasing complexity and levels of automation in modern train control systems, the importance of such verification methods becomes even more pronounced. This paper provides a comprehensive review of formal verification methods used in ETCS, particularly focusing on their application in the Automatic Train Operation Control System of ETCS Level 2 and their evolving role in the design and verification of Level 3.

II. The European Train Control System (ETCS)

A. Overview of ETCS

The European Train Control System (ETCS) is a crucial component of the larger European Rail Traffic Management System (ERTMS). The need for ETCS was born out of the desire for a unified, standardized train control system to replace the multitude of different national systems that currently exist across Europe. Prior to the ETCS, each nation in the European Union had its own train control system, making cross-border rail travel inefficient due to the need for multiple equipment sets and different train operational rules. ETCS standardizes these systems, enabling trains to cross national borders without the need for significant changes in equipment or operations, thereby drastically enhancing the efficiency of international rail traffic.

B. Core features and functionalities

At its core, ETCS provides a sophisticated and comprehensive control system for train operations. It aims to boost interoperability between different nations' railway systems, reduce the costs associated with operating different systems, and enhance railway safety across Europe.

To achieve this, ETCS is divided into different operational levels. Each level provides a different degree of automation and control over the train operations.

- Level 1 is a basic system where data from trackside equipment is transferred to the train using spot transmission principles.
- Level 2, on the other hand, employs a more advanced continuous data exchange between the train and the trackside equipment using a radio-based system. This allows for Movement Authority and other signal information to be relayed directly to the driver's machine interface, removing the need for traditional lineside signals.
- Level 3 introduces the concept of moving blocks, further optimizing track capacity by allowing trains to dynamically adjust their own safety distances based on their speed, location, and the position of other trains.

Furthermore, features like Automatic Train Protection (ATP), which prevents train accidents due to overspeeding or passing signals at danger, and Automatic Train Operation (ATO), which can control the train's movement, are integrated into the system.

C. Benefits and limitations of ETCS

The key benefit of ETCS is its ability to provide seamless interoperability across national boundaries. This functionality not only reduces the time taken for cross-border travel but also cuts down the costs associated with maintaining and operating different systems. By enhancing railway safety through its numerous automated features and strict protocols, it minimizes the risk of human errors leading to accidents. Furthermore, with its ability to manage train traffic more efficiently, ETCS increases track capacity and reduces delays, which leads to more punctual services and improved customer satisfaction.

Despite its many advantages, transitioning to ETCS presents several challenges. The most apparent of these is the significant initial investment required to overhaul existing national control systems. This involves both the physical replacement of equipment and the retraining of staff to operate under the new system. Moreover, while ETCS is designed to bring standardization, variations in implementation continue to exist due to the complexities of replacing national systems, each with their historical and technical idiosyncrasies. This often leads to compatibility issues, requiring further time and resources to resolve.

III. Automatic Train Operation Control System ETCS Level 2

A. Explanation and features of the Automatic Train Operation Control System in ETCS 2

ETCS Level 2 introduces the Automatic Train Operation (ATO) Control System, which significantly enhances automation. In this level, the train continuously communicates with the trackside system. The Radio Block Centre (RBC), ***Fig 1***, provides movement authorities directly to the train's onboard computer, eliminating the need for trackside signals.

The ATO feature manages train operations, including acceleration, cruising, and braking, aiming for optimal speed profiles and precise station stopping. It ensures increased punctuality and energy efficiency, while enabling higher capacity by allowing trains to run closer together safely.

The Automatic Train Operation (ATO) ***Fig2***, system comprises two interconnected subsystems: ATO Track Side (TS) and ATO On Board (OB). ATO-TS gathers and forwards data about trains, tracks, and timetables pertinent to the train journey, whereas ATO-OB receives this data and utilizes it to control and drive the train. The system can be remotely operated by a driver who initiates the autonomous driving. The ATO architecture is such that the Interface Manager allows it to interact with different modules like ATO-TS, ETCS, SMO, SCS, and TIU. The Controller implements the main finite state machine for the various ATO operational modes. The Track Database Manager utilizes odometry data to localize the train on the track and authenticate the journey received from the trackside before the mission starts.

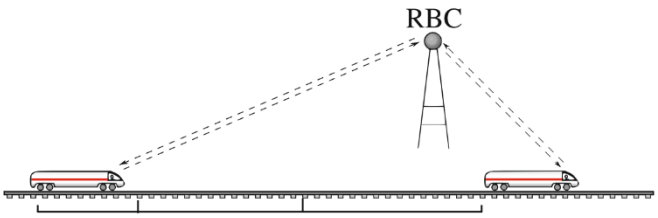


Fig 1. Train system using RBC

The Autonomous Driving Functions module receives track and journey profile data from the Track Database Manager, uses it to generate an optimal speed profile, and issues brake and traction commands to the TIU. The Energy Manager monitors energy levels and assesses the energy needed to complete the current mission using the battery, fuel data, and the status of the traction system.

In a typical scenario where a train is stationed at a charging point, when the train is chosen by an ATO-TS remote driver, ATO-OB checks for the database version alignment with the trackside one and performs internal system tests. The remote driver plots the journey profile to be sent on board and waits for confirmation. When the ETCS mode moves to full supervision and other conditions are met (for example, ETCS and ATO are not applying full service or emergency

brake, ATO-OB is localized on a specific Segment Profile sent by trackside, and train direction is forward), the remote driver can engage ATO-OB enabling autonomous driving.

The ATO's design is subject to complex requirements, and to meet specific project goals, certain standard UNISIG subset requirements had to be customized, with some being added or discarded. For instance, the function for localizing the train was moved on board from the trackside; all requirements related to door management were discarded as the prototype light-vehicle doesn't have doors; camera requirements were added to compensate for the absence of a driver on board, and the interface protocol was customized to manage data for the new functions added.

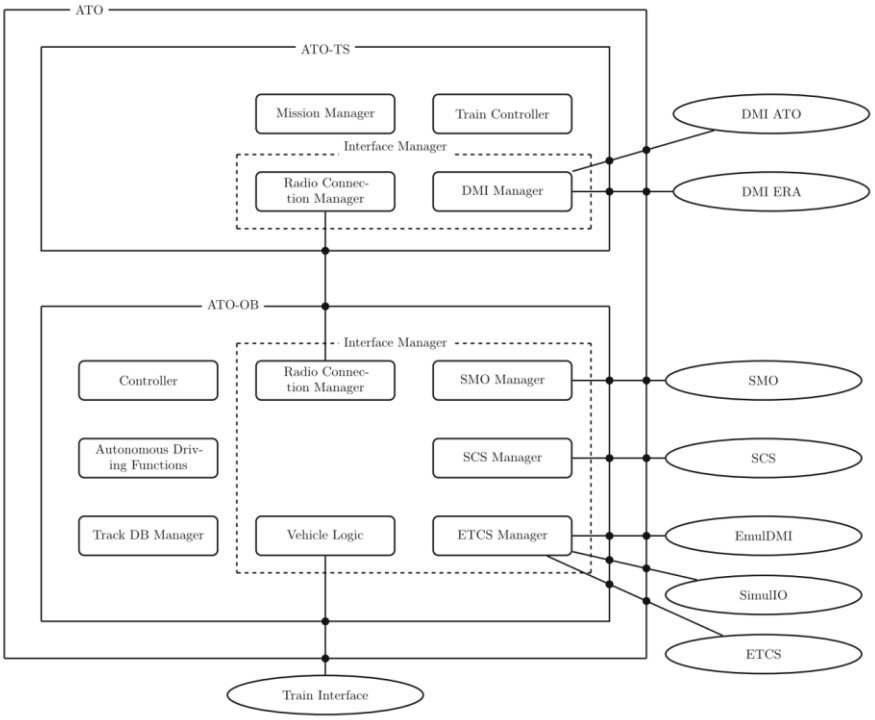


Fig 2. ATO Architecture

Formal Design ATO: The Automatic Train Operation (ATO) system has been designed following formal methods to ensure system robustness, safety, and reliability. The development process follows the V-Model ***Fig,3*** as per the CENELEC EN-50128 standard. This involves progressing from system requirement definitions to software design, coding, unit testing, integration, and system testing, culminating in the operation and maintenance phases.

The development of ATO follows an evolved version of the V-Model, which incorporates aspects of Agile philosophy to manage the complexity and volatility associated with such a project. This approach divides the development process into multiple phases, each focused on a subset of system features. In each phase, the V-model's steps are iterated — refining requirement analysis, system architecture, implementation, unit and integration test — until stability is reached.

The system integration phase is particularly crucial when managing distributed teams and heterogeneous components. During this phase, the source code generated from different models,

such as SCADE and Simulink, is linked together through a feature in the SCADE Suite language called 'external operators'. This feature maps the interface of an operator in the SCADE Suite language to the interface of a corresponding external module through some glue code.

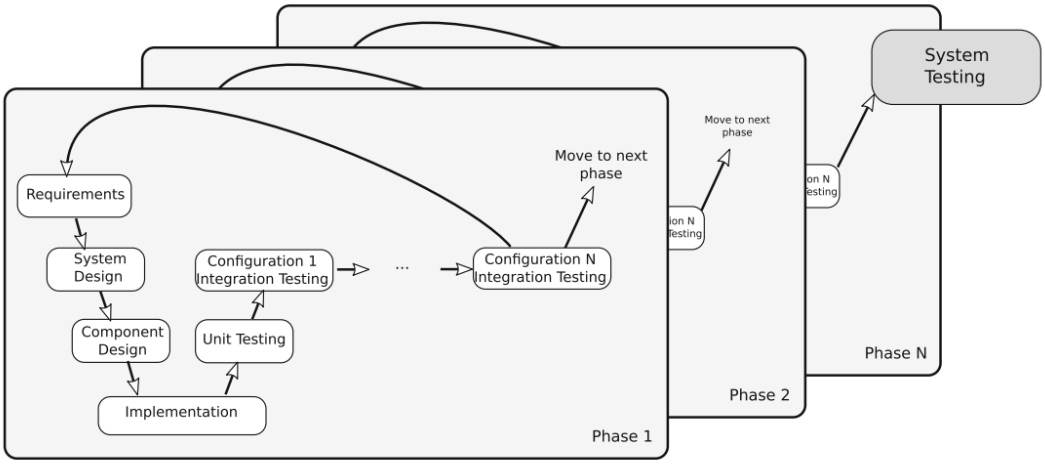


Fig 3: V-Model Architecture

For verification and validation, the process leverages multiple and complementary approaches. Early model validation is performed using SCADE, which checks for compatibility between component interfaces and validates the hierarchical composition of the architecture. Scenario validation, where specific input values are provided and outputs are checked for expected results, is performed using SCADE Suite. This is carried out from the component level up to the system level.

The outcomes of the ATO system can be visualized and verified after executing a scenario through a logging system. This records all relevant diagnostic information, represented by the messages exchanged among components, which can then be processed to verify the scenario execution results align with expected behavior.

Finally, once system integration is consolidated and all individual submodules have been validated, property-based (runtime) verification is performed using model checking (MC) techniques. This process employs a custom tool-chain based on the nuXmv model checker and NuRV, an extension of NuXmv for runtime verification. Through this process, many verifiers can be automatically generated from the properties while requiring negligible effort for refactoring, even when the module interface evolves.

IV. Scenario-Based Modeling for ETCS Level 2

A. Explanation of scenario-based modeling

Scenario-based modeling is a technique used in systems design and testing to simulate real-world scenarios that a system might encounter during its operation. By creating virtual scenarios that reflect real-life conditions, it's possible to examine the system's responses and identify potential issues before actual implementation.

B. Importance and application of scenario-based modeling in ETCS Level 2

In the context of ETCS Level 2, scenario-based modeling is an invaluable tool in system testing and validation. It allows the system to be evaluated in diverse operational conditions, assessing how effectively it can handle potential challenges. This can range from routine operation scenarios to complex or critical situations, such as responding to unexpected obstructions, system failures, or emergency conditions.

Scenario-based modeling helps in identifying any inconsistencies or issues in the system's response, offering opportunities to improve and optimize the system before actual deployment. It also facilitates a more comprehensive understanding of how the Automatic Train Operation Control System behaves under different conditions.

C. Case studies where scenario-based modeling has been applied in ETCS Level 2

Take a detailed look at the operational scenarios associated with the ETCS Level 2 System Requirements Specifications (SRS), focusing particularly on the "Shunting Initiated by Driver" scenario as a case study.

Overview of the 12 procedures required for interoperability within the scope of ETCS Level 2, which have been simplified from specialized terminologies into common language to be easily understood by nonexperts. The procedures include Start of Mission, End of Mission, Shunting Initiated by Driver, Entry in Shunting with Order from Trackside, Override EoA, On-Sight, Level Transitions, Train Trip, Change of Train Orientation, Train Reversing, Joining/Splitting, and RBC/RBC Handover.

The "Shunting Initiated by Driver" scenario for in-depth examination due to its complexity and entity inclusiveness, which embody the basic operations of train movement. The procedure was then described in natural language for accessibility. After that, the authors laid out their approach to Scenario-based modeling.

This process involves the creation of a one-to-one mapping between operational scenarios and Unified Modeling Language (UML) sequence diagrams to ensure model correctness and consistency. Seven rules were provided to guide this mapping, dealing with the translation of entities, event occurrences, operations, time orders, selections, conditions, and concurrent events in the scenarios into the sequence diagram. This translation was carried out following specific steps which included scenario extraction, collection of modeling elements, mapping, and model construction. The resulting sequence diagram was then checked for consistency with the original scenarios.

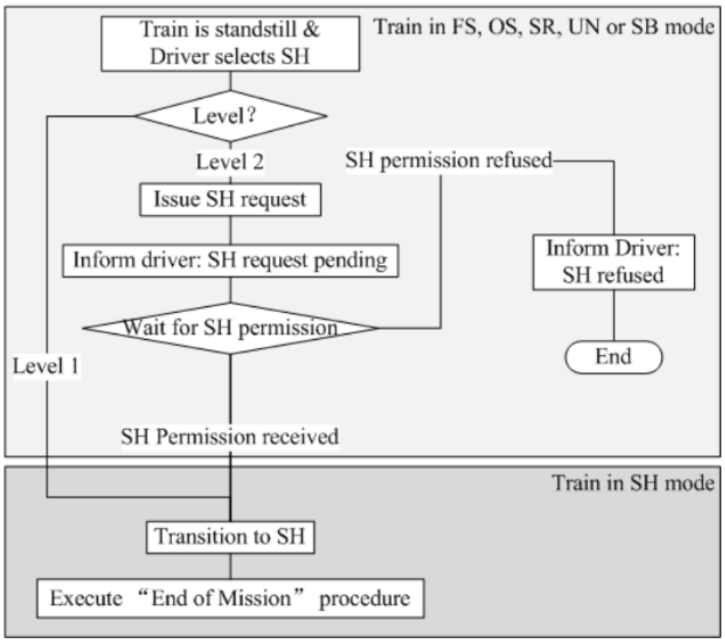
Scenario/Sequence	Diagram	
Entities/ Participants	Event Occurrences or operations	Messages
On-Board RBC Driver	select "Shunting"	Shunting
	Transition to SH mode	Enter SH Mode
	Send/receive the "Request or Shunting" message	Request For Shunting
	Indicate that SH permission request to permission RBC in pending, and await SH	Wait For SH Permission
	Send/receive the SH permission	SH Permission
	Report the mode change	Mode Change
	Send/receive/indicate the "SH refused"	SH Refused

Table 1: The mapping relationship between the scenario and sequence diagrams.

For verification, the presents a set of domain-independent and domain-dependent properties that must be satisfied by the ETCS SRS. The domain-independent properties are general qualities expected of any safety-critical system, such as reachability, transition, deadlock, mutual exclusion, and definitiveness. The domain-dependent properties are specific to the railway domain system, and include safety, liveness, and scenario compatibility.

The sequence diagram was then translated into a format compatible with the NuSMV model checker using defined translation rules. The specified properties were then verified using CTL (Computational Tree Logic) notations, each of which corresponds to a specific property.

Finally, the results of the property showing that the “Shunting Initiated by Driver” scenario satisfies all the required properties. This successful verification demonstrates the effectiveness of the proposed scenario-based modeling method for ensuring the quality and correctness of the ETCS SRS.



In the case of the fourth property's counterexample, a scenario was identified where the onboard equipment transitions to the On Sight (OS) mode before obtaining the confirmation for shunting, and while the train is still in motion. The cause of this error could be a disruption of radio communication with the Radio Block Centre (RBC) or a delay in message transmission.

Figure 4: Graphic Description for Shunting Scenarios

For the fifth property, the counterexample illustrates a scenario where the onboard equipment is stuck in the "Wait For SH Permission" state. This happens when the model does not account for the case when the Shunting (SH) permission message is not received from the RBC.

Lastly, the sixth property's counterexample reveals that if the ETCS Level 1 is in control of the train, a "SH Refused" message from the RBC isn't properly accounted for. Consequently, the onboard equipment does not make the expected switch to the SH mode upon receipt of the "SH Refused" message.

The provided counterexamples offer a valuable trace of potential errors. These errors could be the product of translation discrepancies, issues with the model's construction, or the specifications themselves. Therefore, it's critical to validate whether the semantics of the Computational Tree Logic (CTL) formulas align with the properties as described in natural language. If discrepancies are found, model corrections should be carried out.

V. Formal Verification Methods: An Overview

A. Definition and explanation of formal verification methods

Formal verification methods refer to techniques used to prove or disprove the correctness of a system within a formal system of logic. These methods apply mathematical techniques to check whether a design satisfies certain desired properties, typically in the context of hardware or software systems. They are used to confirm that a system behaves as expected under all possible conditions.

B. Different types of formal verification methods

There are several types of formal verification methods, including model checking, theorem proving, and equivalence checking. Model checking involves testing the properties of a model to ensure its correctness. Theorem proving is a mathematical approach to show that a system's design follows from its specification. Equivalence checking is used to determine if two systems are functionally equivalent.

Formal verification methods applied to a system as intricate as the European Train Control System (ETCS) requires careful modeling of the system's properties and behaviors using mathematical logic or automata. A complete formal verification of such a complex system is a substantial task involving several stages and detailed mathematical understanding. While this exceeds the possibilities of this platform, I can provide a more detailed illustration of how each method can be applied in a simplified context of ETCS.

- **Model Checking**

Model Checking is a powerful technique for formally verifying finite-state concurrent systems. It provides a systematic way to explore all possible states and transitions of a system to validate its properties or detect errors. Given the complexity of modern systems, manual inspection or testing often falls short in achieving this, making model checking an essential tool in system design and validation.

1. Reachability Checking with Model Checking

Given a state machine $M = (S, s0, \delta, \Sigma)$, where S is the set of states, s0 is the initial state, $\delta: S \times \Sigma \rightarrow S$ is the transition function, and Σ is the set of symbols (representing commands or inputs), a reachability property is typically expressed as "$\exists s' \in S$ such that $s0\ \delta^*\ s'$", where δ^* represents the reflexive and transitive closure of δ.

In our example, let sA and sB represent the states "T1 is at station A" and "T1 is at station B" respectively. The reachability property would be "$\exists s' \in S$ such that sA δ^* s'", where s' = sB.

2. Deadlock Verification with Model Checking

Given a state machine M = (S, s0, δ, Σ), a deadlock property is typically expressed as "$\neg\exists s \in S$ (s δ σ s')", which means "there does not exist a state s in S such that there exists a transition from s to some other states' under some symbol $\sigma \in \Sigma$".

In our example, let sD represent a state where "Train T1 is waiting for Train T2 to clear section A, and Train T2 is waiting for Train T1 to clear section B". The deadlock property would be $\neg\exists s \in S$ (s δ σ s'), where s = sD. This means that there is no valid transition (i.e., movement command or control input) that would allow the system to transition out of state sD, indicating a deadlock.

3. Liveness Verification with Model Checking

Given a state machine M = (S, s0, δ, Σ), a liveness property is typically expressed as "$\forall s \in S$, $\exists s' \in S$ such that s δ^* s'", where δ^* represents the reflexive and transitive closure of δ.

In our example, let sA and sB represent the states "T1 is at station A" and "T1 is at station B" respectively. The liveness property would be $\forall s \in S$, $\exists s' \in S$ such that s δ^* s'", where s = sA and s' = sB.

4. Mutual Exclusion Verification with Equivalence Checking

Given two state machines M1 = (S1, s10, δ1, Σ1) and M2 = (S2, s20, δ2, Σ2), an equivalence relation $E \subseteq S1 \times S2$ is a mutual exclusion if $\forall s1, s2 \in S1$, $\forall s3, s4 \in S2$ $((s1, s3) \in E \land \delta1(s1, s2) \land \delta2(s3, s4)) \rightarrow (s2, s4) \in E$.

In our example, M1 could represent the optimized ETCS system and M2 the reference model, and E could represent the states where a single-track section is occupied by a train.

5. Definiteness Verification with Model Checking

Given a state machine M = (S, s0, δ, Σ), a definiteness property is typically expressed as "$\forall s, s1, s2 \in S$, $\forall \sigma \in \Sigma$ $(\delta(s, \sigma) = s1 \land \delta(s, \sigma) = s2) \rightarrow s1 = s2$".

In our example, let sA, sB, and sC represent the states "T1 is at station A", "T1 is at station B", and "T1 is at station C" respectively. The definiteness property would be $\forall s, s1, s2 \in S, \forall \sigma \in \Sigma (\delta(sA, \sigma) = s1 \land \delta(sA, \sigma) = s2) \rightarrow s1 = s2$, where σ represents a command to T1 and $s1, s2 \in \{sA, sB, sC\}$.

6. Safety Verification with Model Checking

A safety property for a system $M = (S, s0, \delta, \Sigma)$ is typically expressed as "$\forall s \in S, \forall \sigma \in \Sigma, \forall s' \in S (s\ \delta\ \sigma\ s') \rightarrow \neg P(s')$", where P is a predicate representing a 'bad' or 'error' state.

In our example, let sB represent the state "T1 and T2 are at the same track section". The safety property would be $\forall s \in S, \forall \sigma \in \Sigma, \forall s' \in S (s\ \delta\ \sigma\ s') \rightarrow \neg(s' = sB)$.

7. Liveness Verification with Model Checking

A liveness property for a system $M = (S, s0, \delta, \Sigma)$ is typically expressed as "$\forall s \in S (s\ \delta^*\ s0) \rightarrow \exists s' \in S (s\ \delta^*\ s') \land P(s')$", where P is a predicate representing a 'good' or 'desired' state.

In our example, let sA and sB represent the states "T1 is at station A" and "T1 is at station B" respectively. The liveness property would be $\forall s \in S (s\ \delta^*\ sA) \rightarrow \exists s' \in S (s\ \delta^*\ s') \land (s' = sB)$.

Properties	Formula	Properties	Formula
Reachability	**"∃s' ∈ S such that s0 δ* s'**	**Definitiveness**	**∀s ∈ S (s δ* s0) → ∃s' ∈ S (s δ* s') ∧ P(s')**
Transition	**"∀x1,...,xn (P1(x1,...,xn) → P2(f(x1,...,xn)))**	**Safety**	**"∀s ∈ S, ∀σ ∈ Σ, ∀s' ∈ S (s δ σ s') → ¬P(s')**
Deadlock	**"¬∃s ∈ S (s δ σ s')**	**Liveness**	**"∀s ∈ S, ∃s' ∈ S such that s δ* s'**
Mutual exclusion	**∀s1, s2 ∈ S1, ∀s3, s4 ∈ S2 ((s1, s3) ∈ E ∧ δ1(s1, s2) ∧ δ2(s3, s4)) → (s2, s4) ∈ E**		

Table 2: The properties and their Formula notations.

- **Theorem Proving**

Theorem Proving is a formal method used to demonstrate the correctness of systems, proving that certain properties hold based on a set of assumptions or premises. This technique, which has

roots in mathematical logic and computer science, is crucial for designing and validating complex systems, including safety-critical systems like the European Train Control System (ETCS).

Theorem proving differs from techniques like model checking in that it isn't limited to finite-state systems, and it doesn't involve exhaustive exploration of all system states. Instead, it leverages logical reasoning to establish the truth of assertions. Here are the ways theorem proving is employed in the context of the ETCS:

1. Theorem Proving for Train Route Validation

Let T be the set of all trains and R be the set of all routes. The predicate Route(T, R) means that a train $T \in T$ is assigned to a route $R \in R$. The predicate Conflict(R1, R2) indicates that routes R1 and $R2 \in R$ conflict.

The theorem to prove, $\forall T1, T2 \in T, \forall R1, R2 \in R. (Route(T1, R1) \land Route(T2, R2) \land T1 \neq T2) \rightarrow \neg Conflict(R1, R2)$, says that if two different trains are assigned to two routes, then these two routes do not conflict.

Using a theorem prover, one can prove the theorem given the rules for route assignments and route conflicts. If the theorem is proven, it validates that the route assignment strategy prevents conflicts between different trains.

2. Theorem Proving for Speed Limit Adherence

Let V be the set of all possible speeds and let T, R be as defined previously. The predicate Speed(T, V) means that a train $T \in T$ is running at speed $V \in V$. The predicate Limit(R, Vmax) indicates that the speed limit on route $R \in R$ is $Vmax \in V$.

The theorem to prove, $\forall T \in T, \forall R \in R, \forall V, Vmax \in V. (Route(T, R) \land Speed(T, V) \land Limit(R, Vmax)) \rightarrow V \leq Vmax$, says that if a train is on a route and its speed is V and the route's speed limit is Vmax, then the train's speed V is less than or equal to the speed limit Vmax.

Again, a theorem prover can be used to prove the theorem given the rules for speed control. If the theorem is proven, it confirms that the speed control strategy always keeps the trains within the speed limit.

- **Equivalence Checking**

Equivalence checking plays an important role in formal verification processes, particularly in comparing two versions of a system to ensure they behave identically under all circumstances.

In the context of ETCS, equivalence checking might be used to compare the behavior of an optimized version of the control system (maybe after an upgrade or update) to a baseline or reference model. The two versions of the system are considered equivalent if they produce the same output given the same input under all possible conditions.

Equivalence checking can be represented mathematically using state machine models. Given two state machines $M1 = (S1, s10, \delta1, \Sigma1)$ and $M2 = (S2, s20, \delta2, \Sigma2)$, an equivalence relation $E \subseteq S1 \times S2$ is established. The state machines M1 and M2 are considered equivalent if and only if the following conditions hold:

1. Initial States: $(s10, s20) \in E$

2. Transition Relation: For any pair of states $(s1, s2) \in E$, if $\delta1(s1, \sigma) = s1'$ for some $\sigma \in \Sigma1$, then there exists s2' such that $\delta2(s2, \sigma) = s2'$ and $(s1', s2') \in E$. And similarly, if $\delta2(s2, \sigma) = s2'$ for some $\sigma \in \Sigma2$, then there exists s1' such that $\delta1(s1, \sigma) = s1'$ and $(s1', s2') \in E$.

In simpler terms, two systems are equivalent if they start from corresponding states and for every transition in one system, there is a corresponding transition in the other system leading to another pair of corresponding states.

This mathematical formulation is used by automated equivalence checking tools to compare different versions of a system and can provide assurances about the correctness of system modifications, upgrades, or optimizations.

VI. Role of Formal Verification in ETCS

A. Significance of formal verification in ETCS

Formal verification plays a critical role in the design, development, and implementation of ETCS. It ensures that the various systems and sub-systems within ETCS are functioning as intended and adhere to the desired safety and performance specifications. Given that ETCS integrates complex systems and controls essential operations like signaling, train movement, and automated control, the role of formal verification is paramount to maintain high levels of safety and reliability.

B. Procedure involved in applying formal verification in ETCS

Formal verification in ETCS involves a sequence of processes. The system requirements are initially specified in a formal language. This formal specification then becomes the basis for the

design and implementation of the system. The formal verification process uses a variety of techniques, like model checking, theorem proving, and equivalence checking, to confirm that the implemented system adheres to the specifications. Anomalies or deviations from the specifications are identified, analyzed, and corrected through iterative development cycles until the system fully complies with the specifications.

VII. Towards Automatic Design and Verification for Level 3

A. Introduction and explanation of ETCS Level 3

ETCS Level 3 is the most advanced level of the European Train Control System. It introduces the concept of a 'moving block', where the train controls its safety distance, enabling a more efficient use of track capacity. Level 3 further enhances automation and optimizes train control, resulting in higher capacity and improved performance.

B. Significance and process of automatic design and verification for ETCS Level 3

With the increasing complexity and automation in ETCS Level 3, the design and verification processes become more intricate and challenging. Automatic design and verification for ETCS Level 3 involve the use of advanced formal methods and computer-aided tools to automate system design, testing, and validation. The objective is to streamline the system development process and achieve a high level of assurance in the system's safety and performance.

The key points for the scheduling and track layout of trains, including the mathematical notation, are as follows:

1. Single Occupation Constraint: Trains should always occupy at most one place of the network at any given time. For each train `tr` in `Trains` and for each time step `ti` in `{t0, ..., ttmax-1}`, this constraint is employed: `occupiesti_tr,e = 1 for one e in E and occupiesti_tr,f = 0 for all other f in E`. This ensures a train cannot be at two different places simultaneously.

$$\bigvee_{c \in chains(l^*_{tr})} \Big(\bigwedge_{e \in c} occupies^{t_i}_{tr,e} \bigwedge_{f \in E \setminus c} \neg occupies^{t_i}_{tr,f} \Big),$$

2. Movement Constraints: Movements of trains should align with the track layout, the maximum speed of each train, and the progression of time. If a train occupies a segment of the network in one time step, it should either occupy the same segment in the next time step or any

segment that can be reached within the train's speed in one time step. For each train `tr` in `Trains`, for each segment `e` in `E`, and for each time step `ti` in `{t0, ..., ttmax-2}`, the constraint is: `occupiesti_tr,e = 1 => occupiesti+1_tr,f = 1 for f in reachable(e, tr)`. Here, `reachable(e, tr)` denotes all segments that the train `tr` can reach when starting from `e` (including `e` itself).

$$occupies_{tr,e}^{t_i} \implies \bigvee_{f \in reachable(e,tr)} occupies_{tr,f}^{t_{i+1}},$$

1.**TTD/VSS Layout Constraints**: Train movement is also constrained by the TTD/VSS layout. A VSS section can only be occupied by at most one train. If two trains occupy the same TTD at the same time, they must be placed in separate VSS. This can be enforced by setting the 'border' variable for one of the nodes between the occupied segments to 1. For each pair of trains `tri, trj` in `Trains` with `tri ≠ trj`, for each segment `e` in `E`, `f` in `TTD(e)`, and for each time step `ti` in `{t0, ..., ttmax-1}`, the following constraint is employed: `(occupiesti_tri,e AND occupiesti_trj,f) => borderv = 1 for v in between(e, f)`. Here, `TTD(e)` is the set of all segments belonging to the same TTD as `e`, and `between(e, f)` denotes all `v` in `V` that belong to the chain connecting `e` and `f`.

$$\left(occupies_{tr_1,e}^{t_i} \wedge occupies_{tr_2,f}^{t_i}\right) \implies \bigvee_{v \in \text{between}(e,f)} border_v,$$

4. Collision Avoidance: It must be ensured that two trains moving in opposite directions cannot "go through one another". For trains `tri,trj` in `Trains`, `tri ≠ trj`, segments `e` in `E`, and for each time step `ti` in `{t0, ..., ttmax-2}`, this is enforced: `occupiesti_tri,e AND occupiesti+1_tri,f => (¬occupiesti_trj,g AND ¬occupiesti+1_trj,g for all g in paths(e, f, tri))`. Here, `paths(e, f, tr)` denotes all paths from `e` to `f` that can be traversed by the train `tr`.

$$occupies_{tr_1,e}^{t_i} \wedge occupies_{tr_1,f}^{t_{i+1}} \implies \bigwedge_{g \in \text{paths}(e,f,tr_1)} \neg occupies_{tr_2,g}^{t_i} \wedge \neg occupies_{tr_2,g}^{t_{i+1}}$$

These constraints ensure the trains move within their defined parameters, including not occupying more than one segment, not exceeding their maximum speed, respecting the VSS layout, and avoiding collisions with other trains. These rules contribute to developing an effective train schedule and track layout that avoids collisions and maximizes efficiency.

C. Challenges and potential solutions in implementing automatic design and verification for ETCS Level 3

The previous constraints ensure that the satisfiability solver only determines variable assignments representing valid scenarios. However, it is also necessary to ensure that the actual design or verification task is solved. This can be accomplished by adding the following final constraints and/or objective functions to the formulation:

1. Verification of Train Schedules: Schedules can be encoded as a list of triples `(tr, e, ti)` defining for each train `tr` in `Trains` what segment `e` in `E` it should occupy at a particular time step `ti` in `{t0, ..., ttmax-1}`. This can be enforced by setting the corresponding variable `occupiesti_tr,e` to 1. To check whether this schedule also works on a given TTD/VSS layout, enforce this layout by setting the corresponding `borderv`-variables to 1 if `v` in `V` separates two VSS sections, and 0 if it does not. The resulting instance is satisfiable if the schedule works with the VSS layout and unsatisfiable otherwise.

2. Generation of VSS Layouts: With a given schedule (which can be enforced as described above), the satisfiability solver is expected to generate a VSS layout that can execute the schedule. A simple approach is setting `borderv = 1` for all `v` in `V`, assigning each segment `e` in `E` to a different VSS. This ensures a train of length `l` occupies exactly `l` segments, minimizing train blockages. If designers want a layout that uses the least possible VSS sections, they can enforce it by adding the objective function.

$$\min : \sum_{v \in V} border_v.$$

3. Schedule Optimization Using the Potential of VSS: For this task, only the departure time and all stops of each train are considered. Enforce `Σ_tmax−1_i=0 occupiesti_tr,e` for each `tr` in `Trains` and its corresponding stops at `e` in `E`. The aim is for the satisfiability solver to determine routes for all these trains which are as efficient as possible. Efficiency can be interpreted in various ways, such as minimizing the total time steps or having each train arrive at its final stop as fast as possible.

$$\bigvee_{i=0}^{t_{max}-1} occupies_{tr,e}^{t_i}$$

To minimize the time steps until all trains reach their destination, introduce the Boolean variable `doneti_tr` where `doneti_tr = 1` states that, at time step `ti`, a train `tr` in `Trains` has left the network, and `doneti_tr = 0` states it is still within the network. Let `lastStop(tr)` in `E` be the last stop of train `tr` in `Trains`. A train can only leave the network after arriving at its final stop. This is encoded by the constraint `doneti_tr => Σ_ij=0 occupiestj_tr,lastStop(tr)`.

Define the fact that all trains have reached their goal at time `ti` by `doneti := ∀ tr∈Trains doneti_tr`. The number of time steps in which not all trains have arrived at their final stop should be minimized. This is enforced by adding the objective function `min : Σ_tmax−1_i=0 ¬doneti`.

$$done_{tr}^{t_i} \implies \bigvee_{j=0}^{i} occupies_{tr,lastStop(tr)}^{t_j}$$

$$done^{t_i} := \bigwedge_{tr \in Trains} done_{tr}^{t_i}$$

$$\min : \sum_{i=0}^{t_{max}-1} \neg done^{t_i}.$$

Combining all the constraints and, if applicable, the respective objective functions, a symbolic formulation describing the considered design or verification task is obtained. Passing this formulation to a satisfiability solver can yield either an assignment to all border- and occupies-variables, indicating a viable VSS layout and train positions (for verification tasks or layout/schedule generation tasks), or no such assignment exists, indicating that no viable train routes or VSS layout exists for the given constraints/objectives within the considered time interval. In either case, the designers have their design or verification task solved automatically.

VIII. Future Perspectives

Formal verification methods, with their ability to ensure safety and reliability, are becoming more relevant as the railway industry continues to evolve. The emerging trends in formal verification are promising, with a focus on increased automation and the use of more sophisticated tools and techniques.

There is a growing interest in leveraging artificial intelligence and machine learning in formal verification. This approach could potentially automate the verification process, improve efficiency, and identify errors or anomalies more accurately. This development could be a significant breakthrough in managing the increasing complexity of systems like ETCS.

Moreover, the concept of "digital twins" – creating a real-time digital replica of physical systems for testing and optimization – is gaining traction. This approach could be highly beneficial in the context of ETCS, allowing for comprehensive testing and verification under a wide range of operational scenarios.

IX. Conclusion

The European Train Control System (ETCS), especially at Levels 2 and 3, relies extensively on formal verification methods. These methods are instrumental in maintaining the high levels of safety and reliability required in railway operations. They provide a rigorous approach to validate system designs, thereby preventing operational failures and ensuring smooth, efficient operations.

The continued evolution and expansion of railway systems, coupled with increasing levels of automation, emphasize the growing significance of formal verification. While challenges exist, including the need for advanced tools and expertise, the future of formal verification in ETCS is promising. It's anticipated that with continued research and innovation, these methods will become even more effective in ensuring the safety, reliability, and efficiency of ETCS and other complex systems.

The journey towards a more advanced and secure railway system is ongoing. As we navigate this path, the insights and learning from formal verification methods in ETCS will undoubtedly be invaluable in guiding the way forward.

Reference

[1] Weidenhaupt, K., Pohl, K., Jarke, M. & Haumer, P., Scenarios in system development: current practice. IEEE Software, 15(2), pp. 34-45, 1998.

[2] Rolland, C., Achour, C.B., Cauvet, C., Ralyté, J., Sutcliffe, A., Maiden, N., Jarke, M., Haumer, P., Pohl, K., Dubois, E. & Heymans, P., A proposal for a scenario classification framework. Requirements Engineering, 3(1), pp. 23-47, 1998.

[3] Uchitel, S., Kramer, J. & Magee, J., Negative scenarios for implied scenario elicitation. ACM SIGSOFT Software Engineering Notes, 27(6), pp. 109-118, 2002.

[4] Bai, X. Tsai, W.-T., Feng, K., Yu, L. & Paul, R., Scenario-based modeling and its applications. Proc. of the 7th IEEE Int. Workshop. On ObjectOriented Real-Time Dependable Systems, IEEE Computer Society, Los Alamitos, pp. 0253, 2002

[5] Lima, V., Talhi, C., Mouheb, D., Debbabi, M., Wang, L. & Pourzandi, M., Formal verification and validation of UML 2.0 sequence diagrams using source and destination of messages. Electronic Notes in Theoretical Computer Science, 254, pp. 143-160, 2009.

[6] Tanuan, M.C., Automated Analysis of Unified Modeling Language (UML) Specifications, in Master's thesis, University of Waterloo, Canada. 2001.Specifications, in Master's thesis, University of Waterloo, Canada. 2001.

[7] Cavada, R., et al.: The nuXmv symbolic model checker. In: Biere, A., Bloem, R.(eds.) CAV 2014. LNCS, vol. 8559, pp. 334–342. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08867-9 22

[8] CENELEC: EN 50128, Railway applications - Communications, signaling and processing systems - Software for railway control and protection systems (2011)

[9] Cimatti, A., Tian, C., Tonetta, S.: NuRV: a nuXmv extension for runtime verification. In: Finkbeiner, B., Mariani, L. (eds.) RV 2019. LNCS, vol. 11757, pp. 382–392.Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32079-9 23

[10] Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge(2000)

[11] European Union Agency for railways: ERTMS - Making the railway system work better for society (2016)

[12] Fern´andez-Rodr´ıguez, A., Fern´andez-Cardador, A., Cucala, A., Dom´ınguez, M., Gonsalves, T.: Design of robust and energy-efficient ATO speed profiles of metropolitan lines

considering train load variations and delays. IEEE Trans. Intell. Transp. Syst. 16(4), 2061–2071 (2015)

[13] Ferrari, A., ter Beek, M.H.: Formal methods in railways: a systematic mapping study (2021)

[14] International Association of Public Transport: A global bid for automation: UITP Observatory of Automated Metros confirms sustained growth rates for the coming years, Belgium

[15] Licheng, T., Tao, T., Jing, X., Shuai, S., Tong., L.: Optimization of train speed curve based on ATO tracking control strategy. In: Chinese Automation Congress (2017)