Towards a Novel Approach to Railway Safety using STPA and Promise Theory*

Felix Schaber[1], Atif Mashkoor[2], Michael Leuschel[3]

[1] Hitachi Rail GTS, Austria
[2] Johannes Kepler University Linz, Austria
[3] Heinrich Heine University Düsseldorf, Germany

Abstract. The safety of railway systems requires cooperation between many interdependent subsystems. As safety responsibilities are split between these subsystems, modeling cooperation, and conditional dependencies between subsystems become a central issue. This paper proposes the safety promise assessment method (SafePAM), an iterative approach to modeling these dependencies formally. SafePAM enriches the STPA - a structured hazard analysis technique based on systems theory - resulting in a formal description of dependencies provided by the promise theory. Together, this yields a flexible method of iterative refinement, which allows the embedding of novel system designs within their environment while upholding the overall system safety properties. In contrast to previous approaches, SafePAM permits integrating conditional dependencies within the model description without assuming pairwise independence between conditions. We evaluate the proposed method in a case study from the railway domain. We describe the system behavior based on promises that allow a seamless link between domain-specific properties and the system's physical properties, allowing domain experts to validate the resulting model.

Keywords: STPA · Promise Theory · Railway Safety · Safety Analysis · Formal Methods

1 Introduction

Safety is a system property. In mission-critical software systems, this is often associated with complex interactions of multiple controllers and processes. System theoretic process analysis (STPA) [14] is an emerging method to model these interactions and judge their impact on safety. A safety-guided design process can be built upon these foundations, systematically considering safety from the

* The research presented in this paper has been partly conducted within the IVOIRE project, which is funded by "Deutsche Forschungsgemeinschaft" (DFG) and the Austrian Science Fund (FWF) grant # I 4744-N. The work of Atif Mashkoor has been partly funded by the LIT Secure and Correct Systems Lab, which is sponsored by the province of Upper Austria. The work of Felix Schaber has been partly funded by the European Union 🇪🇺 Grant Agreement # 101102001.

onset of system design. However, building executable software controllers based on these results calls for a systematic extension of the STPA method. The classical STPA method describes a methodology for analyzing a system for safety properties but does not prescribe a method for building a system that fulfills these properties. Building an executable system description requires formalizing the safety controller's process model (controller view of its environment) and control algorithm (behavioral patterns when the controller provides actions to its environment).

Deriving a correct controller process model depends on environmental assumptions. These assumptions are critical for the safety of the overall system. The promise theory [2] provides a systematic treatment of these assumptions. It allows for tracking and assessing dependencies, the dependability and stability of individual assumptions, and their impact on safety. A promise is a statement of behavior between agents. If the promise depends on a condition, it is called a conditional promise.

This paper uses conditional promises to model conditional dependencies between multiple assumptions, making the causal chains explicit. These causal chains link the behavior of multiple controllers, providing a basis for assessing if the overall system can uphold its safety promises in the real world. Therefore, we propose the safety promise assessment method (SafePAM) to model these conditional dependencies formally. Its key contribution is extending STPA to describe the details necessary to describe cooperation formally and systematically. We combine a top-down hazard analysis to generate a safety model (STPA) with a bottom-up approach to ensure the implementation of this safety model (formalized promises from the promise theory).

The SafePAM method exploits STPA to abstractly describe automated controllers by their process model and control algorithm. Multiple controllers' process models and control algorithms are viewed independently by default. The promise theory then adds explicit dependencies between the control algorithms of multiple controllers and their environment. This is especially beneficial for large cyber-physical systems, whose dependencies can span many controllers due to functional and physical interactions. Since SafePAM describes a method for systematically deriving the conditions of the analysis to be valid and provides an approach to assessing the stability of cooperation, this differs from existing STPA-based approaches, where the assumptions are taken as given, and the validity of the approach fades once assumptions are violated.

We evaluate the efficacy of the SafePAM method in a case study from the railway domain. In the next-generation train control system case study, we use the SafePAM method to systematically model the interaction of as many as five controllers (driver, onboard unit, braking system, trackside control, and tracks) to constrain the train movement successfully. The conditions under which these dependencies are valid can become quite complex to model otherwise.

The rest of the paper is structured as follows: In Section 2, we provide an overview of STPA and the promise theory. Section 3 presents the proposed safety promise assessment method (SafePAM). Section 4 applies the SafePAM method to the railway case study. Section 5 discusses the important lessons learned. Section 6 discusses the related work. The paper is concluded in Section 7, while hinting at the future direction.

2 Background

The SafePAM method builds upon STPA and promise theory results. The concepts used from these methods are described below for the context.

2.1 STPA

STPA is a top-down safety analysis method based on systems theory. It describes safety as emergent system behavior, including controller interactions and individual controller failures. The concept of controllers is very general and includes both automated systems and human controllers or organizations. Controllers enforce controller constraints by providing control actions and receiving feedback from their environment. The constraints are designed to prevent unsafe control actions. Loss scenarios build upon these unsafe control actions and describe concrete scenarios (traces) of how the unsafe control action could occur. The control structure can be decomposed hierarchically, allowing the controller model to be refined to the required level of detail. SafePAM uses controller constraints and loss scenarios as inputs.

2.2 Promise theory

The promise theory describes agent interactions as promises of intended behavior. It is based on a bottom-up approach and is used to model the stability of cooperative behavior. In SafePAM, agents are modeled after the STPA controllers and can be refined to fine-grained sub-controllers if needed. Formally, a promise is described as a tuple $\langle S, R, b(\tau, \chi_\tau), \pi_n \rangle$ where S is the sender, R the receiver, b the body of the promise and π_n the name of the promise. The body is parameterized by a type $\tau \in \{+, -\}$, representing either a promise to give(+) or to receive(-) and a constraint χ_τ, which expresses the constraint of the promise. The promise can be made conditional on keeping another promise c by another agent. This can also be expressed in infix notation, where the arrow symbolizes a promise, and the content of the promise body is written above the arrow. The body consists of the actual promise p, the symbol | indicating conditional dependence, and the actual condition c. A conjunction of multiple conditions is denoted by separating the conditions with a semicolon.

$$\pi_n : S \xrightarrow{\pm p|c} R \tag{1}$$

Stable cooperation requires a binding between a promise to give and a promise to receive. For the sake of brevity, we will focus solely on the promises to give (+) in the case study.

Together, the promises form a graph, which tracks the influence of the individual promises on the overall system behavior. An example of such a graph is shown in figure 1. Whether a promise is kept is judged by each agent individually.

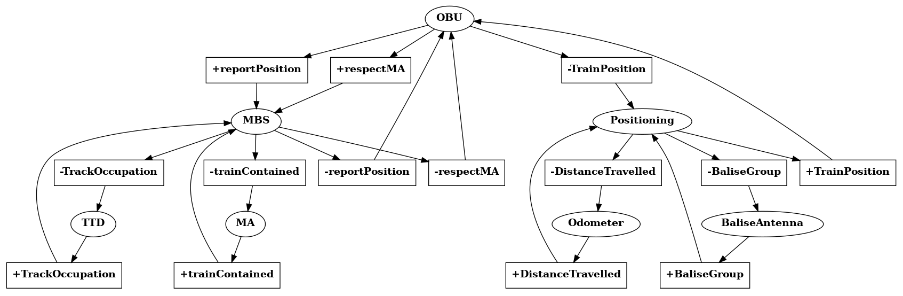

Fig. 1. Simplified example of a promise graph from the railway domain. Ellipsoids represent actors, while rectangles represent promise bodies between the actors connected by the arrow.

Formally, the assessment is a mapping from promises to the interval $[0, 1]$, where 1 represents complete confidence in keeping the promise. For SafePAM, this value depends on the judgment of domain experts and documented agent behavior and dependencies.

3 Safety promise assessment method

The SafePAM uses the STPA system analysis. This consists of defining the system boundary and purpose and performing the STPA analysis. Figure 2 gives a graphical overview of the SafePAM approach, which is based on these central ideas:

- Interpret STPA unsafe controller actions (which include the context of unsafe system behavior) as loss patterns, defining behavioral patterns that the control algorithms are designed to prevent. These unsafe controller actions are translated into formal controller constraints, which describe invariants on the controller behavior. STPA loss scenarios are concrete examples of how an unsafe controller action could occur. These scenarios can be used for acceptance tests to validate that the solution concept meets the controller's safety requirements.
- Build an abstract process model for the safety controller as the controller's view of the world, guided by the STPA hierarchical controller structure of the overall system.
- Sub-states and all their possible dependencies are added, driven by the need to describe and fulfill the formalized controller constraints.

– The analyst's assumptions are modeled using promise theory. Conditional promises systematically link the interaction of multiple controllers, allowing us to assess their stability and dependability. The concrete process model is derived from this, limiting all possible dependencies in the abstract world to those relevant to a concrete situation.
– Derive a dynamic concrete safety model for hazard assessment in real-time from the generic abstract safety model by combining it with observed data about the current real-time environment.

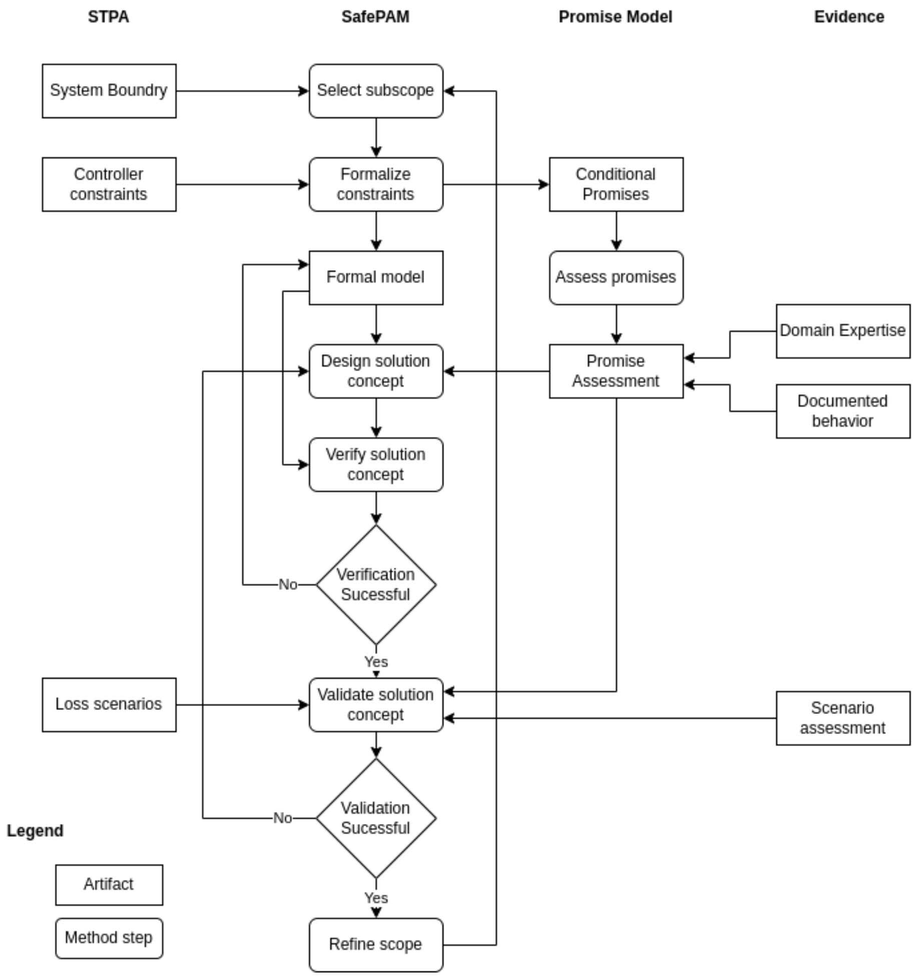

Fig. 2. Graphical overview of SafePAM as simplified flowchart

1. Formalize controller constraints. This step extends STPA by formalizing process model variables and controller constraints. It starts by selecting a system to model in detail (usually a single controller) and a scope for the iteration. Promise Theory enables the systematic modeling of assumptions and conditional dependencies. The scope can be extended in later iterations. However, all promises made at the previous iterations shall be kept, possibly by a different combination of promises.

1a. Formalize process model variables. The individual controller constraints are derived from the unsafe control actions. The context under which the control action is unsafe shall be formalized. All process model variables shall have delimited, non-overlapping states. The variables used in the context often depend on other process model variables. The conditional dependence of the context on these variables is modeled as well.

1b. Model conditional dependencies using promises. The conditional dependencies found in step 3a are described using promise theory. In addition, the influence of other controllers on the internal process model variables (transmitted through commands or feedback) is formulated as promises. If these, in turn, depend on the behavior of other controllers, this is modeled in the condition of the promise body.

2. Assess promises. All promises found in the previous step should be assessed for their dependability. This may depend on knowledge about the operating environment, system specifications, domain expert judgment, etc. The assessment results should be documented for each promise, along with supporting evidence. In addition, the dependability assessment may change when additional information about the dependencies becomes available later (e.g., information supplied at runtime). For conditional promises, the dependability of the conditions provides an upper limit for the dependability of the whole promise.

This also provides an early opportunity for system validation. Domain experts may identify promises likely to be broken, and the search for evidence during the assessment may highlight unpromised but necessary behavior or common environmental dependencies. When expanding the scope in a new iteration, the conditions of the promises shall also be checked to ensure they remain satisfied. If they are unsatisfied, this may indicate that the scope change affects assumptions within the solution concept.

3. Develop solution concepts. This step takes the found promises as inputs and provides a structured methodology for developing solution concepts based on these promises.

3a. Design solution concept. In this step, the desired properties of the solution concepts are modeled. This also reflects the environmental constraints under which the system will operate, as informed by the assessment promised in the previous step. Sometimes, choosing between different strategies may be necessary to handle inconsistencies and promises that are known to be unreliable. The choice of strategy should be documented and explained here. This is similar to documenting the architectural assumptions of the solution concepts. These assumptions should be relatively stable and change only infrequently.

3b. Model the solution concept and add it to the system model. Here, the concrete solution concept is built. The solution concept is described as an agent within the controller, promising the properties in step 3a. The dependencies on the environmental promises are documented, linking the solution model and the environmental conditions and assumptions under which the system was designed.

4. Verify solution concepts against the system model. Verification can be performed by model checking the controllers against the desired behavior. The promises from the solution concepts and the environmental promises shall be checked. The verification conditions can be expressed, e.g., as invariants or temporal properties.

5. Validate solution concept. Validate that the environment fulfills the promises of the solution concepts. Domain experts (e.g., end users and system suppliers) can be asked to assess the promises used during the development of the solution concepts if they have not already been performed in Step 2. Visualization of the solution concepts (e.g., animation) and their current reported state and actual state may greatly aid domain expert assessment. Loss scenarios found in the STPA can be used as example scenarios. As loss scenarios are known to lead to hazards, their unfolding should be prevented by the developed solution concepts.

6. Iterate model. Returning to Step 1, the model can be enhanced iteratively by adding environmental assumptions or modeling additional hazardous control actions. Care must be taken at each iteration to verify that the promises used when developing the solution concepts of previous situations still hold. Otherwise, the promises shall be updated, and the solution concepts will be adapted to continue to hold under the changed scope.

4 Case study

We use a simplified model of the European train control system (ETCS) train protection system as a case study. The fundamental purpose of such a system is to avoid trains colliding. This requires the system to detect the presence of trains and other obstacles within its control area. Detection may be performed by sensors with fixed positions along the track or by moving sensors. For example, the presence of trains may be determined by fixed sensors along the track-sensing train wheelsets (i.e., track-side train detection systems). Alternatively, the train itself may report where it believes to be within regular intervals (i.e., train position report). This section describes how the method was applied to the case study.

4.1 Controllers

Moving block system. The moving block system (MBS) is the trackside control system that assesses the risk of Movement Permission commands and object state change requests (i.e., changing the position of a railway point). It only grants a request if it does not result in a hazardous situation (now or in the future) and translates it into movement authorities (MA). MBS intervenes if the risk of a hazardous situation becomes intolerable after a movement is granted. In addition, it sends configuration values for the controlled region to the OBU.

Train. The train is the physical configuration of train cars moving on the track, with at least one traction unit. It can accelerate or brake but not determine its running path. The driver and the OBU (see below) can command the brakes. To simplify the analysis for this case study, we assume that the train consists of a single car only.

On-board unit. The on-board unit (OBU) continuously supervises the braking curve until the end of the movement authority (MA) and commands the brakes if the driver brakes too late. It also continuously supervises the permitted speed and commands the brakes if the speed is significantly exceeded. It shows the driver the current braking curve, permitted speed, and ETCS mode. These ETCS modes influence the split of responsibility between OBU, the train driver, and the trackside control. For example, under full supervision mode, the MBS is responsible for ensuring the train route is passable (i.e., railway points are in the correct position for the route) and that no obstacles are present within the MA. This contrasts with on-sight (OS) mode, where the driver monitors the track for potential obstacles. Finally, when in staff-responsible mode, the driver is responsible for monitoring the route and the track to ensure it is passable and free of obstacles.

The OBU also requests movement authorities from the MBS. When a new movement authority is received from the MBS, supervision is updated to the end of the new movement authority. It also sends train position reports to the MBS to inform them of the current train positions.

Trackside train detection. Trackside train detection (TTD) continuously supervises a fixed track section for the presence of vehicles. It provides feedback on whether the track section is free or occupied by the trackside control. TTDs are commonly implemented either by using track circuits or by axle counters. Track circuits detect the presence of train axles by utilizing the electrical conductance of the axles. In contrast, axle counters count the number of axes entering and leaving the track section and declare the section free if the difference between these numbers is zero. The trackside system can provide a command to reset the axle counter to zero to address miscounts.

Braking. The braking system reduces train speed when commanded by the driver or OBU. The system differentiates between service brakes for graceful deceleration and emergency brakes for maximum brake performance. It also selects the suitable physical brake type(s) for the commanded brake deceleration (e.g., recuperation brake, block brake, eddy brake, etc.). Actual braking performance also depends on the adhesion conditions on the track/wheel interface. It shall reduce skidding and sliding as much as possible for situations with reduced adhesion conditions.

Positioning. The positioning system measures the position of the train front relative to fixed location balises based on the track. It detects the passing of location balises and measures the distance traveled since reading the last valid location reference. It also calculates the confidence interval and train speed and determines the train running direction. This information is summarized in the train position report and regularly sent to the trackside system via a radio link.

Track. The track provides physical guidance and steering for the train. Track conditions (e.g., curve radius, cant deficiency, etc.) strongly influence the train's maximum running speed. Adhesion conditions between the track and the train wheel are essential to limit braking performance.

4.2 Formalized controller constraints

The STPA analysis starts with defining the losses. For this case study, the focus is on the first loss:

L-1. Loss of life or injury to people on the train

This, in turn, leads to the following hazards, where the identifier in the square brackets refers to the linked loss.

H-1. Trains don't maintain a safe distance from other trains. [L-1]

H-2. Train doesn't maintain a safe distance from other obstacles. [L-1]

For the train control system, we select the following unsafe control action found using the STPA approach as an example for further analysis:

UCA-MBS-1. MBS provides MA to the OBU when the risk of the MA intersecting with other trains or obstacles is intolerable. [H-1,H-2]

This unsafe control action corresponds to the following controller constraints:

C-MBS-1. MBS shall not provide a full supervision MA to the OBU when the MA may intersect with other trains or obstacles known at the time of the MA request. [UCA-MBS-1]

C-MBS-2. MBS shall not provide an on-sight MA to the OBU when the MA may potentially include infrastructure elements that are not passable. [UCA-MBS-1]

C-MBS-3. MBS shall not provide a staff-responsible MA to the OBU when the MA may potentially intersect with the movement authorizations of other trains. [UCA-MBS-1]

These, in turn, depend on process model inputs:

- Granted movement authorities
- Reported train positions
- Region between retarded train rear (min safe rear end) and advanced train front (max safe front end)
- Time of position report
- Reported track occupation
- Fixed region on track

Based on these inputs, the internal process model of the MBS consists of the following:

- Regions where train presence is known
- Regions where train presence is likely (now or in the future)
- Regions where train presence is possible (now or in the future)
- Regions where obstacle presence is likely (now or in the future)
- Movement authorities
- Regions where trains may move without authorization (uncontrolled regions)

For stable cooperative behavior, promise theory requires matching a promise to give (+) with a promise to receive (−) between agents. For the sake of brevity, this case study only lists the promises to give.

Using the process model inputs listed above to determine obstacle presence requires modeling additional domain assumptions. The remainder of this section describes how to model such promises.

One such assumption is that all trains within the control area report their position at regular time intervals *tReport* and known accuracy. The following promise expresses this assumption:

$$OBU \xrightarrow{+reportPosition|Connection,TrainPosition,Config(tReport)} MBS \tag{2}$$

However, this assumption is conditional on multiple other assumptions. For example, the train must have the equipment for position reporting (i.e., an OBU) with an established communication session. This, in turn, requires the train to have radio signal reception and, therefore, electrical power. The following promises express this.

$$Odometer \xrightarrow{+DistanceTravelled|ElectricalPower} Positioning \tag{3}$$

$$BaliseAntenna \xrightarrow{+BaliseGroup|ElectricalPower} Positioning \tag{4}$$

$$Positioning \xrightarrow{+TrainPosition|DistanceTravelled,BaliseGroup} OBU \tag{5}$$

In addition, information about the train position can also be obtained from trackside train detectors (TTD). These systems are installed at a fixed location along the tracks and detect whether a monitored section is occupied by a train or empty. For the MBS to notice the change, this requires an established connection between the TTD and MBS as well as enough time to detect the change ($tDetection$):

$$TTD \xrightarrow{+TrackOccupation|Connection,StateChange(tDetection)} MBS \tag{6}$$

The purpose of the movement authority (MA) is to constrain the train dynamics within a region permitted by MBS. An additional promise is therefore necessary to describe this behavior. We introduce MA as a separate agent to embody this behavior:

$$MA \xrightarrow{+trainContained|respectMA} MBS \tag{7}$$

This has the advantage of tightly linking the information contained within the MA to the desired behavior. However, the OBU needs environmental conditions to be fulfilled to respect the constraints of the MA. One significant promise is that the braking performance is sufficient to stop the train within the MA region. This, again, can be modeled as a promise of the braking system on the train to the OBU.

$$OBU \xrightarrow{+respectMA|MA,sufficientBrakes} MBS \tag{8}$$

$$Braking \xrightarrow{+sufficientBrakes|sufficientAdhesion} OBU \tag{9}$$

As promise (9) shows, sufficient adhesion between the train wheels and the track is another requirement for sufficient braking. This depends on the physical track conditions and is therefore modeled as a promise of the physical TrackState.

$$TrackState \xrightarrow{+sufficientAdhesion} OBU \tag{10}$$

4.3 Assess promises

Assessing whether these dependencies hold for the given system depends on domain knowledge.

Train position reporting. Starting with promise (2), we see the ability to determine the train position ($TrainPosition$) is itself conditional. The $TrainPosition$ promise (5) indirectly depends through promises (3) and (4) on the availability of *ElectricalPower* to the *Odometer* and *BaliseAntenna*, which are located onboard the train. Therefore, we have to assume that train position reports could be missing if a radio hole or *ElectricalPower* is unavailable to *Odometer* or *BaliseAntenna*. The promise (2) to report positions regularly is violated, necessitating the design of solution concepts with this restriction in mind. We treat this as a possible condition that can occur at any time, as the numerical conditional probabilities between multiple dependencies depend on the concrete operational situation and may be unknown for the general case.

Enforcing the movement authority. Limiting the area where the train is allowed to run must be enforced by other means. Promise (8) shows the mechanism within ETCS for that purpose: the train OBU promises to the trackside MBS that the end of the movement authority will be respected. This is achieved by calculating the train braking curve on-board and preventively applying the brakes if the train driver does trigger the brakes before the relevant supervision limit of the braking curve. To calculate the braking curve, promise (9) relating to the assumed braking performance is required. However, sufficient adhesion between the physical tracks and train wheels expressed by promise (10) is required to achieve the specified braking performance.

The OBU depends on additional input to adjust for reduced adhesion conditions. These inputs can either come from the MBS or the train driver. But even then, the reduced adhesion factor does sometimes not enter the braking curve calculation. Therefore, even a reliable promise to provide the reduced adhesion factor is insufficient to fulfill the promise (8). The MBS will require additional solution concepts to ensure the train remains within its allowed region in case of reduced conditions.

Combining TTDs with train position reports. The train position reports may be combined with reports from the track-side train detection systems to detect non-communicating trains and reduce the time to position updates. Promise (6) describes the information about track occupation transmitted to MBS. However, this combination is non-trivial.

The two types of reports describe fundamentally different information at different times. Train position is reported as distance to a known location, while train detection monitors a fixed section along the track and transmits the state change after *tDetection*. The reports have different transmission times and transmission triggers. Therefore, they can also interleave and remain inconsistent for short enough periods. A systematic way to model and discuss these domain assumptions with domain experts in an easily accessible way for all parties is beneficial in such cases.

4.4 Develop solution concept.

With the promise model, the controller constraint can be restated as follows:

C-MBS-1. The MBS shall not provide a full supervision MA to the OBU when the MA may contain regionWithObstacles.

regionWithObstacles is defined as the region where a train presence (i.e., by train position reports) is either known or is expected (i.e., due to a granted movement authority).

To compute this concept, we must know when a train has reliably vacated an area within an MA. Only then can this region be declared free of obstacles, allowing the MBS system to reduce the size of the granted MA. This requires that the vacated region always increase, as future observations may indicate obstacles after the region has already been declared vacant. Nevertheless, complete removal of the vacated region is allowed to reduce an MA region.

We introduce a new agent *vacatedMARegion* instead of overloading the existing MBS with this solution concept. This allows a clear separation between the dependencies of this concept and the rest of MBS.

In addition, a promise is also dependent on the concepts always (**A**) and until (**U**) from linear temporal logic. Where there is a need to separate previous from current states, states with a tick($'$) represent the new state, while states without represent the previous state.

$$vacatedMARegion \xrightarrow{+\chi_{noTrainPresence}|monotonicallyIncreasing(vacatedMARegion)} MBS$$

$$\chi_{noTrainPresence} \equiv \mathbf{A}(trainPresenceImplausible\mathbf{U}MARegionReduced)$$

A train is known to only move forward, with the running direction being enforced by the OBU.

$$OBU \xrightarrow{+runningDirectionEnforced|MA(runningDirection)} MBS \tag{11}$$

$$physicalTrain \xrightarrow{+\chi_{movingForward}|runningDirectionEnforced} MBS \tag{12}$$

$$\chi_{movingForward} \equiv trainPosition(rear)' \geq trainPosition(rear) \tag{13}$$

The reported track occupation and train position always describe a past situation. Under the condition that the running direction is enforced ($runningDirectionEnforced$), the last known rear position is always in the rear of the physical train position.

$$knownTrainPosition \xrightarrow{+\chi_{confirmedRear}|Connection(tDelivery>0)} MBS \tag{14}$$

$$\chi_{confirmedRear} \equiv \mathbf{A}knownTrainPosition(rear) <= physicalTrainPosition(rear)$$

knownTrainPosition shall then delimit the region where the train was at some time in the past based on the information received by MBS. To keep the promise to *vacatedMARegion*, it is vital that $runningDirectionEnforced$ holds, so the rear of the known train position moves only forward. $\chi_{confirmedRear}$ expresses that this shall hold even if reports (of train position or track occupation) are missing or arrive in the wrong sequence.

$$knownTrainPosition \xrightarrow{+increasing(vacatedMARegion)|\chi_{rearMoved}} vacatedMARegion$$

$$\chi_{rearMoved} \equiv knownTrainPosition(rear)' \geq knownTrainPosition(rear)$$

This constrains the update behavior of *knownTrainPosition*. The rear shall only be moved forward once it is inevitable that new reports do not invalidate previous updates. This assumption shall guarantee that the rear may only move forward or remain in the same position.

We know that the information we received about the train position will only eventually be consistent. An update should, therefore, be delayed until enough time has passed for the information to be consistent, or additional domain promises allow an update beforehand. Such domain promises are defined below.

$$TTD \xrightarrow{+\chi_{forwardMovingOccupation}|MA,!(trainPresenceKnown \vee trainPresenceExpected)} MBS$$

$$\begin{aligned}\chi_{forwardMovingOccupation} \equiv \mathit{state} = \mathit{occupied} \quad \& \quad \mathit{state}' = \mathit{free} \\ \Rightarrow \mathbf{A}\mathit{state} = \mathit{free}\end{aligned}$$

$$OBU \xrightarrow{+\chi_{forwardMovingRear}|MA,Odometer} MBS$$

$$\begin{aligned}\chi_{forwardMovingRear} \equiv \mathbf{A}\mathit{time}(\mathit{PosReport}2) > \mathit{time}(\mathit{PosReport}1) \\ \Rightarrow \mathit{minSafeRear}(\mathit{PosReport}2) > \mathit{minSafeRear}(\mathit{PosReport}1)\end{aligned}$$

Implementing this promise amounts to implementing the solution concepts. We use a formal description in the promise bodies to allow a formal verification and validation of the solution concepts.

As the formal verification of the case study is ongoing, the following sections give an overview of these steps.

4.5 Verify that solution concepts ensure controller constraint

Verification (i.e., is the model sound and do all invariants hold) can be performed, i.e., model checking of the controllers against the desired behavior. The promises from the solution concepts and the environmental promises shall be checked. The verification conditions can be directly derived from the promises and expressed as invariants or temporal properties.

4.6 Validate the solution behaves as expected

Validate (i.e., does the model match the end-user's perception) that the environment fulfills the promises of the solution concepts. Domain experts can be asked to assess the promises used to develop the solution concepts. Visualization of the solution concepts, together with its current reported state and actual state, may greatly aid expert assessment.

4.7 Iterative model enhancement

The model can be iteratively enhanced by adding environmental assumptions or modeling additional hazardous control actions. Care must be taken at each iteration to verify that the promises used when developing the solution concepts of previous situations still hold. If this is not the case, then the promises shall be updated, and the solution concepts will be adapted to continue to hold under the new promises.

5 Lessons learned

By applying SafePAM, we have successfully established solution concepts for the MBS train protection system case study.

- We learned that to limit train movement to known regions, the concept of MA needs to be extended with additional solution concepts to contain the train. As shown by the assessment of the conditional promises, this depends on other promises beyond the MBS control (*sufficientAdhesion*).
- Even if the MBS provides the OBU with information about the current adhesion conditions, this information is not always used to calculate the braking characteristics.
- We also learned under which conditions an MA can be released behind the train and showed how this is connected to sequential promises of train position reports.

6 Related work

Formalization of STPA. Formal descriptions of the STPA artifacts have been proposed early on. Thomas described an extension of STPA for requirements generation and analysis [20]. This work formalizes the process model and addresses the issue of completeness through a systematic and exhaustive state space search. Another approach proposed by Colley and Butler is to iteratively formalize the system requirements using the Event-B language, which supports a refinement workflow [4]. Howard et al. also proposed to combine the analysis of safety and security of critical infrastructure [12] and formalized the requirements using Event-B [11]. To detect when changes in assumption may impact the STPA analysis results, Leveson and Thomas also proposed introducing assumption-based leading indicators [15] within the context of STPA. These indicators are designed to detect the violation of assumptions early, which indicates a migration to a state of higher risk and increases the likelihood of an accident.

Formal description of conditional dependencies. A well-known approach to formalize conditional dependencies is the Rely-Guarantee Method [16], also known as the Assume-Guarantee method [7]. This method describes the assumptions of each agent regarding its environment, relying on invariants. Only if the rely conditions are fulfilled will the agent provide its guarantees to its environment. In contrast to the Promise Theory, each rely-guarantee binding assessment is considered unanimous and globally valid. In addition, partial fulfillment of bindings is not considered in the Rely-Guarantee Method. Specific frameworks indented for risk modeling and supporting conditional dependencies have also been proposed [9, 10]. Within model-based system engineering, an approach to safety analysis by probabilistically modeling faulty dependencies has also been proposed [19]. In contrast to promise theory, faults are assumed to occur independently and the assessment if a fault occurred is global rather than local for each agent.

Formalization of moving block. The verification of ETCS has been studied for some time in the formal methods community. For example, [17] discusses the formal verification of ETCS as a case study. There are few recent surveys about the use of formal methods for railways, in general, [8, 1] and for using the B-method, in particular, [3]. There are several industrial Event-B models for the safety analysis of railway systems [5, 6, 18]. Of particular relevance for our case study is the Event-B model in [13] (inspired by [18]), which focuses on the core safety aspect of an ETCS moving block system. This could, in principle, be integrated into SafePAM.

7 Conclusion

This paper proposes SafePAM, a method to iteratively model and design a critical safety system based on promises. We validated the approach for a selected subset of the safety analysis of a train protection system based on the moving block concepts. Results show that the required domain details can be formalized, and the resulting promises can describe the system in sufficient detail to be assessed by a domain expert. The method includes a systematic validation of domain assumptions based on promise assessment. Finally, the solution concepts are validated based on loss scenarios.

The use case in this paper represents a simplified subset of the complete train protection system. All relevant parts of the safety analysis must be included to study the overall system's real-life complexity. Therefore, in the future, we would like to iteratively expand the model to include all relevant constraints of the safety analysis and study the method's behavior as it scales to a system of real-life complexity.

References

[1] Maurice H. ter Beek et al. "Adopting Formal Methods in an Industrial Setting: The Railways Case". In: *Formal Methods – The Next 30 Years*. Ed. by Maurice H. ter Beek, Annabelle McIver, and José N. Oliveira. Cham: Springer International Publishing, 2019, pp. 762–772. ISBN: 978-3-030-30942-8.

[2] J. A. Bergstra and Mark Burgess. *Promise Theory: Principles and Applications.* Oslo: XtAxis, 2014. ISBN: 978-1-4954-3777-9.

[3] Michael J. Butler et al. "The First Twenty-Five Years of Industrial Use of the B-Method". In: *Proceedings FMICS 2020.* LNCS 12327. 2020, pp. 189–209. DOI: 10.1007/978-3-030-58298-2_8. URL: https://doi.org/10.1007/978-3-030-58298-2%5C_8.

[4] John Colley and Michael Butler. *A Formal, Systematic Approach to STPA Using Event-B Refinement and Proof.* 2013. URL: https://eprints.soton.ac.uk/352155/1/STPAandEventB.pdf.

[5] Mathieu Comptier et al. "Property-Based Modelling and Validation of a CBTC Zone Controller in Event-B". In: *Proceedings RSSRail 2019.* Ed. by Simon Collart Dutilleul, Thierry Lecomte, and Alexander B. Romanovsky. LNCS 11495. Springer, 2019, pp. 202–212. ISBN: 978-3-030-18743-9. DOI: 10.1007/978-3-030-18744-6_13. URL: https://doi.org/10.1007/978-3-030-18744-6%5C_13.

[6] Mathieu Comptier et al. "Safety Analysis of a CBTC System: A Rigorous Approach with Event-B". In: *Proceedings RSSRail 2017.* Ed. by Alessandro Fantechi, Thierry Lecomte, and Alexander B. Romanovsky. LNCS 10598. Springer, 2017, pp. 148–159. ISBN: 978-3-319-68498-7. DOI: 10.1007/978-3-319-68499-4_10. URL: https://doi.org/10.1007/978-3-319-68499-4%5C_10.

[7] Xinyu Feng, Rodrigo Ferreira, and Zhong Shao. "On the Relationship Between Concurrent Separation Logic and Assume-Guarantee Reasoning". In: *Programming Languages and Systems.* Ed. by Rocco De Nicola. Red. by David Hutchison et al. Vol. 4421. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 173–188. DOI: 10.1007/978-3-540-71316-6_13. URL: http://link.springer.com/10.1007/978-3-540-71316-6_13 (visited on 04/22/2024).

[8] Alessio Ferrari et al. "Survey on Formal Methods and Tools in Railways: The ASTRail Approach". In: *Proceedings RSSR 2019.* Ed. by Simon Collart-Dutilleul, Thierry Lecomte, and Alexander Romanovsky. Springer, 2019, pp. 226–241. ISBN: 978-3-030-18744-6.

[9] Mario Gleirscher and Radu Calinescu. "Safety Controller Synthesis for Collaborative Robots". In: *2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS).* Oct. 2020, pp. 83–92. DOI: 10.1109/ICECCS51672.2020.00017. arXiv: 2007.03340 [cs, eess]. URL: http://arxiv.org/abs/2007.03340 (visited on 04/24/2024).

[10] Mario Gleirscher, Radu Calinescu, and Jim Woodcock. "RiskStructures : A Design Algebra for Risk-Aware Machines". In: *Formal Aspects of Computing* 33.4-5 (Aug. 2021), pp. 763–802. DOI: 10.1007/s00165-021-00545-4. URL: https://dl.acm.org/doi/10.1007/s00165-021-00545-4.

[11] Giles Howard et al. "A Methodology for Assuring the Safety and Security of Critical Infrastructure Based on STPA and Event-B". In: *Int. J. Critical Computer-Based Systems* 9 (2019), pp. 56–74.

[12] Giles Howard et al. "Formal Analysis of Safety and Security Requirements of Critical Systems Supported by an Extended STPA Methodology". In: *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 2017 IEEE European Symposium on Security and Privacy: Workshops (EuroS&PW). Paris: IEEE, Apr. 2017, pp. 174–180. ISBN: 978-1-5386-2244-5. DOI: 10.1109/EuroSPW.2017.68. URL: http://ieeexplore.ieee.org/document/7966988/ (visited on 02/08/2024).

[13] Michael Leuschel and Nader Nayeri. "Modelling, Visualisation and Proof of an ETCS Level 3 Moving Block System". In: *Proceedings RSSRail 2023*. 2023, pp. 193–210. DOI: 10.1007/978-3-031-43366-5_12. URL: https://doi.org/10.1007/978-3-031-43366-5%5C_12.

[14] N. Leveson. *Engineering a Safer World: Systems Thinking Applied to Safety*. The MIT Press, 2016.

[15] Nancy Leveson and John Thomas. *STPA Handbook*. 2018. 188 pp. URL: https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf.

[16] Leonor Prensa Nieto. "The Rely-Guarantee Method in Isabelle/HOL". In: *Programming Languages and Systems*. Ed. by Pierpaolo Degano. Red. by Gerhard Goos, Juris Hartmanis, and Jan Van Leeuwen. Vol. 2618. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 348–362. DOI: 10.1007/3-540-36575-3_24. URL: http://link.springer.com/10.1007/3-540-36575-3_24.

[17] André Platzer and Jan-David Quesel. "European Train Control System: A Case Study in Formal Verification". In: *Formal Methods and Software Engineering*. Ed. by Karin Breitman and Ana Cavalcanti. Red. by David Hutchison et al. Vol. 5885. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 246–265. DOI: 10.1007/978-3-642-10373-5_13. URL: http://link.springer.com/10.1007/978-3-642-10373-5_13 (visited on 04/25/2024).

[18] Denis Sabatier. "Using Formal Proof and B Method at System Level for Industrial Projects". In: *Proceedings RSSRail 2016*. Ed. by Thierry Lecomte, Ralf Pinger, and Alexander B. Romanovsky. LNCS 9707. Springer, 2016, pp. 20–31. ISBN: 978-3-319-33950-4. DOI: 10.1007/978-3-319-33951-1_2. URL: https://doi.org/10.1007/978-3-319-33951-1%5C_2.

[19] Danielle Stewart et al. "AADL-Based safety analysis using formal methods applied to aircraft digital systems". In: *Reliability Engineering & System Safety* 213 (2021), p. 107649.

[20] John Thomas. *Extending and Automating a Systems-Theoretic Hazard Analysis for Requirements Generation and Analysis*. SAND2012-4080, 1044959. May 1, 2012, SAND2012–4080, 1044959. DOI: 10.2172/1044959. URL: https://www.osti.gov/servlets/purl/1044959/.