On the Integration of Requirements Analysis and Safety Analysis for Safety—Critical Software

Rogério de Lemos, Amer Saeed and Tom Anderson
Department of Computing Science
University of Newcastle upon Tyne, NE1 7RU, UK

Abstract

Experience has shown that in computer based safety—critical systems, faults introduced during the phase of software requirements analysis can and do cause accidents. In this paper, we present an approach for integrating the activities of requirements analysis and safety analysis. The aim of safety analysis is to determine the risk associated with requirements specifications and assess whether this is acceptable within the context of system risk. The advantage of conducting the safety analysis during the early phases of software development is that safety related errors are less likely to propagate through to subsequent phases of development. The applicability of the proposed approach is demonstrated by conducting the safety analysis of an example based on a train set crossing. The example illustrates how the approach to safety analysis supports verification within a formal model of the train set system and how the validation of the formal model is performed.

Keywords: safety—critical systems, requirements analysis, safety analysis, formal models, verification and validation, safety analysis techniques.

1. Introduction

A major motivation for the work presented in this paper is the belief that substantial improvement in the dependability of computer based safety—critical systems can be achieved by applying a detailed safety analysis to the requirements specifications of the software before proceeding to any subsequent phases of development. The aim of this analysis is to locate and remove faults introduced during the requirements phase of software development, which provides the system context in which the software requirements must be considered. This is a fundamental issue for safety—critical systems because “safety” is essentially an attribute of the system rather than just software. Hence, the safety analysis of the software requirements specifications must be conducted with respect to their contribution to the overall system risk.

This paper focuses on an approach for the qualitative safety analysis of safety specifications (requirements specifications for safety) which aims to increase assurance that the level of risk associated with the safety specifications is acceptable [1, 2]. The safety analysis is conducted by identifying those circumstances which could violate a safety specification and cause the system to enter into a hazard state, establishing the level of risk to be associated with that safety specification, and assessing whether this risk is acceptable within the overall
system risk. If the risk is not acceptable the safety specification has to be modified or combined with other safety specifications in order to reduce the risk. In this paper, we examine the effectiveness of the approach by applying it to a case study based on a train set crossing [3].

The rest of this paper is organised as follows. The next section describes a methodology for the analysis of safety requirements which is based on the requirements analysis of the software, the safety analysis of the safety specifications, and a graph for recording the results of both analyses. In section 3, the proposed approach for integrating the safety analysis with the process of requirements analysis of the software is presented in the context of a case study based on a train set crossing. Finally, section 4 contributes some concluding remarks.

2. A Methodology for the Analysis of Safety Requirements

In this section, we overview the basic methodology for analysis of safety requirements; a more detailed discussion is given elsewhere [4, 5]. Assuming that a separation can be made between mission and safety requirements of a system, the proposed methodology focuses on the analysis of the safety requirements. The methodology consists of a framework which provides guidelines for conducting the analysis, a graph that depicts the relationships between the safety specifications produced during the analysis, and a set of informal and formal techniques appropriate for the issues to be analysed.

The framework for analysis, which partitions the analysis into smaller phases, is obtained from a hierarchical model of the structure of a control system by identifying the key system components and their interactions which establish, respectively, the domains of analysis and their inter—relationships. (A domain of analysis delineates a particular scope of analysis, providing horizontal abstraction.) This process of decomposition is conducted recursively, with each step leading to a lower level of analysis. (A level of analysis slices the system into layers, providing vertical abstraction.) Hence, the framework is defined by associating its phases with the domains of analysis and the ordering of the phases with the identified inter—relationships between the domains. In the following we enumerate those phases of the framework which are relevant for the case study.

The phase of Environment Analysis produces a statement of the aim and purpose of the system and determines those failure behaviours of the system which constitute accidents. The phase of Safety Plant Analysis identifies the potential hazards, and the properties of the plant that are relevant to the safety requirements, such as the physical laws and rules of operation that govern plant behaviour. The phase of Safety Plant Interface Analysis delineates the interface between the plant and controlling system, and specifies the behaviour that must be exhibited at that interface in terms of the properties of sensors and actuators. Finally, the phase of Safety Controlling System Analysis establishes a top level
organization for the controlling system in terms of the properties of its components, and their interactions.

At each phase of the framework two activities are conducted in parallel: requirements analysis which produces the safety specifications, and safety analysis which assesses whether the risk associated with the specifications is acceptable. The safety specifications obtained from a phase of requirements analysis will feed into the safety analysis and any defects identified by the safety analysis will be used to modify the safety specifications before proceeding to the next phase. Because of the iterative nature of the requirements and safety analysis, the overall process should be thought of as cyclic rather than sequential. Also, to ensure that essential system behaviour is not precluded, the safety analysis must be supplemented by an analysis of the restrictions that a safety specification will impose on the mission.

2.1. Requirements Analysis

Once a domain is identified and its behaviour specified, the process of conducting requirements analysis of a domain consists of identifying relevant entities, specifying their required behaviour and interactions with other entities, and confirming that the composition of the behaviour of the entities complies with the behaviour of the domain. The entities obtained from the decomposition of the domain are analysed at lower levels of analysis.

For the application of formal techniques, the approach adopted is to employ techniques in accordance with the characteristics of the system to be analysed within the different domains of analysis. Relevant formalisms are grouped into two classes: property-oriented and operational [6]. The degree of applicability of one class of formalism versus the other is related to the level of abstraction being considered: at higher levels of abstraction there is a natural tendency to use property-oriented formalisms, whereas at lower levels operational formalisms predominate.

2.2. Safety Specification Graph

The safety specifications produced from the different phases of analysis, are organized into a Safety Specification Graph (SSG) [4, 5]. An SSG is a directed acyclic graph, in which the vertices represent the safety specifications and the edges denote relationships between them.

A safety specification is composed of the following elements: a safety strategy (a scheme to maintain safe behaviour), a set of assumptions (hypotheses about the behaviour of the system), and unsafe system states (states of the system in which corrective action must be undertaken). A safety integrity level (an indicator of the likelihood of a safety specification achieving its required safety features under all stated conditions within a stated measure of use [7]) is associated with each safety specification, which must be inherited by related safety specifications at a lower level of abstraction.
The relationships between safety specifications must ensure that safe behaviour is maintained by the specifications. There are three kinds of relationships between safety specifications: *coverage* which ensures that an accident does not occur in the absence of the all the hazards associated with it, *exclusion* which ensures that the safety constraint should exclude all the associated hazards, and *refinement* which ensures that a safety strategy should maintain the safety specifications of the previous layer to which it is linked.

When more than one safety specification is related to a specification in a previous layer, then either the safety specifications are exclusive alternatives and a choice will have to be made in later phases of development to select and derive a single strategy, or the strategies complement each other and all are needed to attain sufficient confidence that the risks are acceptable. These situations are distinguished by annotating the edges with a “⊕” in the exclusive case, and a “⊙” in the complementary case.

In summary, the SSG provides support for: conducting the safety analysis of safety specifications, achieving traceability between safety specifications (hence providing support to analyse the impact in the violation of assumptions and the modification of safety specifications), and organizing and constructing a safety case (a collection of evidence and arguments that can be used to justify that the level of risk is acceptable). An example SSG is given in figure 2 (see section 3.2.3), and its role is illustrated by the case study.

2.3. Safety Analysis

In safety-critical systems, the key issue when assessing the risk of the safety specifications is to determine whether the contribution of the software to the overall system risk is acceptable. For this to be possible, a bridge has to be established between the safety analysis of the system and the software; in our approach this bridge is established through the SSG. The results of a safety analysis are used for *risk reduction* and subsequently for *certification* of the software within the context of the system in which it will be embedded.

For each domain of analysis, the safety analysis of the safety specifications is performed from two perspectives: *qualitative* and *quantitative*. The purpose of the qualitative analysis is to identify those circumstances in which a specification can be violated, and to analyse the impact of these violations upon the safety of the system. These circumstances are related, for example, to the violation of assumptions upon which a specification is based. The quantitative analysis complements the qualitative analysis by attaching occurrence probabilities to these circumstances. In the following, we overview how qualitative and quantitative safety analysis of safety specifications are conducted in order to provide assurance that system safe behaviour is maintained within an acceptable level of risk.

2.3.1. Qualitative Safety Analysis

For each domain of analysis, the qualitative safety analysis of the safety specifications comprises two activities: preliminary analysis and vulnerability analysis. It should be
recognised that the activities do not represent two temporally sequential, mutually exclusive, phases of qualitative safety analysis. Typically, these activities will frequently interact during the analysis.

Preliminary analysis, through verification, provides evidence that under defined circumstances the safety specifications maintains safe behaviour of the system, while vulnerability analysis, through validation, provides evidence that the risk associated with the violation of the safety specifications is acceptable. Alternatively, the preliminary analysis is concerned with functional requirements which maintain the system safety (captured by the safety specifications), and the vulnerability analysis is concerned with non–functional requirements which establish the safety targets for the system, and consequently the software.

2.3.1.1. Preliminary Analysis

The provision of evidence that the safety specifications maintain safe behaviour is determined by the consistency and accuracy of the safety specifications. Internal consistency of the safety specifications of the SSG is verified by conducting horizontal and vertical checks. The horizontal checks establish that at each layer of the SSG the safety specifications are not in conflict with each other. The vertical checks are applied between consecutive layers of the SSG, and the relationships that must be established, to ensure compliance between the layers, follow from the edges of the SSG. As a consequence of the preliminary analysis, the circumstances under which safe behaviour is maintained are clearly scoped and organised to reflect their contribution to system safe behaviour.

2.3.1.2. Vulnerability Analysis

The verification process from preliminary analysis should not be used as the only source of evidence that a safety specification is able to preserve the safety integrity levels dictated from the safety specifications from the previous layer. Moreover, the verification of a safety specification does not ensure that new hazards are not introduced into the system. Additional analysis must be conducted in order to show that a safety specification does not violate the specified safe behaviour of the system by influencing hazards from which it is not derived. Vulnerability analysis probes the safety specifications to identify circumstances which can affect the safe behaviour of the system, and once such circumstances are identified the safety specifications can be modified to be more robust. One of these probes is to check whether the assumptions used for the preliminary analysis are accurate in representing reality because during the preliminary analysis these assumptions are treated as axioms. The intent behind the vulnerability analysis is similar to that for a traditional safety analysis.

Although logical formulae are useful in obtaining a high–level view of the relationship between the safety specifications, and their associated assumptions, such formulae provide
limited support for a failure analysis. A suitable representation, for such analysis, is one which supports the identification of possible violations of the specifications that can lead to hazard states. Within our safety analysis methodology we advocate the use of traditional safety analysis techniques, such as fault tree analysis (FTA), failure modes, effects and criticality analysis (FMECA) and hazard and operability studies (HAZOPS). These techniques have been extensively employed in the process industries, and more recently to hardware and software designs [8, 9, 10] In this paper, to perform the vulnerability analysis of the safety specifications we employ fault tree analysis (FTA) [11]. A key feature of FTA that makes it suitable here, is that the analysis is restricted to the identification of violations in the safety specifications, and their associated assumptions, that lead to one particular undesired system state.

In the proposed approach, both vulnerability and quantitative analysis (to be discussed in the next section) can be performed in a straightforward manner by using FTA and minimal cut sets. Minimal cut sets define the “failure modes” of the top event of a fault tree, and are usually obtained when a fault tree is evaluated. A minimal cut set is the smallest combination of primary events sufficient for the top event to occur.

2.3.2. Quantitative Safety Analysis

Although the qualitative approach strives to achieve total assurance for the safety specifications, there are three basic limitations which suggest that this aim may not be completely achieved.

- Some faults introduced during the requirements phase may not be discovered despite verification and validation.

- Past experience has shown that verification may itself contain faults.

- Despite vulnerability analysis, assumptions upon which a specification relies may still be violated.

We infer that even after performing qualitative analysis we are still faced with uncertainties concerning the risk of the safety specifications; we must therefore quantify these uncertainties in order to obtain a “measured” level of confidence in the risk of the safety specifications. In other words, the aim is to obtain an early prediction of the contribution from the software to the risk of the system.

FTA and minimal cut sets can be used to perform the quantitative analysis, by associating relative weights to the primary events, instead of accurately predicting the occurrence probability of a primary event. These relative weights are similar to those employed by FMECA techniques to obtain the risk priority numbers that provide a notion of the relative risk associated with a particular failure [12]. Once weights are associated with each primary event, we calculate the relative quantitative importance of each cut set, from which we are
able to analyse to what extent the violation of an assumption may contribute to a hazard state [13].

3. Case Study: Train Set System

With the aim of exemplifying the proposed approach for conducting qualitative safety analysis of safety specifications, a train set crossing was selected as a case study that raises safety—critical issues similar to those found at the traditional level crossing (i.e. road—rail). A partial analysis of the safety requirements of this case study has been previously conducted [3], and the results of that analysis are compared in section 3.5 with the results of the safety analysis conducted in this paper. To conduct the requirements analysis of this example, we adopt Timed History Logic (THL) [14] (an overview of THL is given in Appendix B). The names of variables will be prefixed by indices that identify the domain under analysis.

Based on the safety specifications presented in the following two sections, we illustrate in sections 3.2.3 and 3.3.3 how the safety analysis of the safety specifications guides the selection and enhancement of the safety specifications, and provides evidence to increase the assurance that the level of safety is acceptable.

3.1. Environment Analysis

Of the various accidents possible on the train set, we consider only two.

*Accidents*

\[ AC_1 \] – trains of the same type collide;
\[ AC_2 \] – trains of different type collide.

3.2. Plant Analysis

In the following we describe the system model of the train set and the safety specifications imposed over the behaviour of the trains.

3.2.1. Model of the Train Set

The plant consists of two types of trains: primary \((p)\) and secondary \((s)\), the primary trains run on the circuit \(C(p)\) and the secondary trains on the circuit \(C(s)\), both types of trains travel in a clockwise direction only. The circuits are divided into sections (which are numbered in a clockwise direction) and there are two separate crossing sections at which they intersect, as shown in figure 1.

The model of the train set is described by the following sets, constants and functions.
Figure 1. The train set circuits and a crossing section

<table>
<thead>
<tr>
<th>Sets</th>
<th>Range</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>${p, s}$</td>
<td>The two types of trains.</td>
</tr>
<tr>
<td>$M$</td>
<td>${a, b}$</td>
<td>The two crossing sections, i.e. the sections where the circuits meet.</td>
</tr>
<tr>
<td>$Tr(c)$</td>
<td>${1, ..., Nt(c)}$</td>
<td>The trains of type $c$, $Nt(c)$ denotes the number of trains of type $c$.</td>
</tr>
<tr>
<td>$S(c)$</td>
<td>${0, ..., Ns(c)}$</td>
<td>The sections on the circuit $C(c)$, $Ns(c)+1$ denotes the number of sections on circuit $C(c)$.</td>
</tr>
</tbody>
</table>

Table 1. Sets for Plant Model

<table>
<thead>
<tr>
<th>Constants</th>
<th>Range</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CC(c, m)$</td>
<td>$S(c)$</td>
<td>The section on $C(c)$ that is part of crossing section $m$.</td>
</tr>
<tr>
<td>$DZ(c, m)$</td>
<td>$\mathcal{P}(S(c))$</td>
<td>Danger zone, $DZ(c, m) = (CC(c, m), CC(c, m) \cup 1)$.</td>
</tr>
<tr>
<td>$L_{\text{max}}$</td>
<td>$\mathbb{R}$</td>
<td>The maximum length of a train.</td>
</tr>
<tr>
<td>$m_{\text{nst}}$</td>
<td>$S(c)$</td>
<td>The maximum number of sections required to stop a train.</td>
</tr>
<tr>
<td>$V_{\text{max}}$</td>
<td>$\mathbb{R}$</td>
<td>The maximum velocity of a train.</td>
</tr>
<tr>
<td>$V_{\text{min}}$</td>
<td>$\mathbb{R}$</td>
<td>The minimum (non-zero) velocity of a train.</td>
</tr>
</tbody>
</table>

Table 2. Constants for Plant Model

<table>
<thead>
<tr>
<th>Functions</th>
<th>Signature</th>
<th>Comments</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{\text{max}}(i, j)$</td>
<td>$L \times S(c) \times S(c) \rightarrow \mathbb{R}$</td>
<td>The maximum distance between sections $i$ and $j$.</td>
<td>$m$</td>
</tr>
<tr>
<td>$d_{\text{min}}(i, j)$</td>
<td>$L \times S(c) \times S(c) \rightarrow \mathbb{R}$</td>
<td>The minimum distance between sections $i$ and $j$.</td>
<td>$m$</td>
</tr>
</tbody>
</table>

Table 3. Functions for Plant Model

The behaviour of the trains is captured by the variables defined in the table below.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Anticipated Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{\text{Contain}}(c, x)$</td>
<td>$\mathcal{P}(S(c))$</td>
</tr>
<tr>
<td>$PReserved(c, x)$</td>
<td>$\mathcal{P}(S(c))$</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>$PSection(c, x)$</td>
<td>$S(c)$</td>
</tr>
<tr>
<td>$PStop(c, x)$</td>
<td>{true, false}</td>
</tr>
<tr>
<td>$PVelocity(c, x)$</td>
<td>${PVelocity(c, x) \in \mathbb{R} \mid 0 \leq PVelocity(c, x) \leq V_{max}}$</td>
</tr>
</tbody>
</table>

**Table 4. Variables for Plant Model**

$PContain(c, x)$ denotes the set of sections that might contain any part of train $x$ on circuit $C(c)$, $PReserved(c, x)$ denotes the set of sections reserved for train $x$ on circuit $C(c)$, $PSection(c, x)$ uniquely denotes the section number containing the front of train $x$ on circuit $C(c)$, $PStop(c, x)$ denotes that train $x$ on circuit $C(c)$ is stopping or has stopped, and $PVelocity(c, x)$ denotes the velocity of train $x$ on circuit $C(c)$. In the following formulae, “$\oplus$” denotes addition and “$\ominus$” subtraction modulo the number of sections.

**Rules of Operation**

For the train set, we have the following rules of operation which describe the behaviour of trains. The first rule $pro1$ describes a static property of the trains. The rules $pro2$, $pro3$ and $pro4$, describe behaviour of trains from a discrete perspective. The rules $pro5$ and $pro6$ describe train behaviour from a continuous perspective. The system predicates and history predicates are interpreted, respectively, as invariants and history relations; that is, they are satisfied for all time points $t$ and all intervals $[T_0, T_1]$ for valid histories of the train set system.

**pro1.** A train occupies one section or (at most) two sections.

$$\forall c \in L: \forall x \in Tr(c): [PContain(c, x) \subseteq \{PSection(c, x), PSection(c, x) \oplus 1\}].$$

**pro2.** The direction in which the trains travel is the same as the direction in which the sections of the circuit are numbered, and trains travel in steps of one section.

$$\forall c \in L: \forall x \in Tr(c): [PSection(c, x)(T_0) \neq PSection(c, x)(T_1) \Rightarrow \exists t \in [T_0, T_1]: PSection(c, x)(t) = PSection(c, x)(T_0) \oplus 1].$$

**pro3.** The trains require at most $mnst$ sections to stop.

$$\forall c \in L: \forall x \in Tr(c): [\forall t \in [T_0, T_1]: PStop(c, x)(t) \Rightarrow PSection(c, x)(T_1) \in \{PSection(c, x)(T_0), \ldots, PSection(c, x)(T_0) \oplus mnst\}].$$

**pro4.** When the front of a secondary train $x$ is in any of the $mnst + mnst + 2$ sections that precede the danger zone $DZ(s, m)$, there is no secondary train in the sections $\{CC(s, m), \ldots, CC(s, m) \oplus 4\}$.

$$\forall m \in M: \forall x \in Tr(s): \quad PSection(s, x) \in \{CC(s, m) \oplus (mnst + mnst + 2), \ldots, CC(s, m) \oplus 1\} \Rightarrow \forall y \in Tr(s): PSection(s, y) \notin \{CC(s, m), \ldots, CC(s, m) \oplus 4\}.$$
**pro5.** When a secondary train is moving its velocity is at least $V_{min}$ (assumes instantaneous acceleration and deceleration between 0 and $V_{min}$).

\[ \forall y \in Tr(s): PVelocity(s, y) = 0 \lor PVelocity(s, y) \geq V_{min}. \]

**pro6.** When the front of a secondary train is in a danger zone its velocity is at least $V_{min}$ (assumes that a secondary train does not stop in a danger zone).

\[ \forall m \in M: \forall y \in Tr(s): PSection(s, y) \in DZ(s, m) \Rightarrow PVelocity(s, y) \geq V_{min}. \]

### 3.2.2. Plant Safety Specifications

In the following, we present the plant safety strategies, without detailing the process of requirements analysis from which these specifications were obtained. In this study we focus on the safety specifications derived to avoid accident $AC_2$.

**Hazard**

An *hazard* is a state, or a set of states, of the system that, together with certain conditions in the environment of the system, can lead to an accident.

**HZ$_{2,1}$** – Some part of a primary train and a secondary train are in the same crossing section.

\[ \exists m \in M: \forall c \in L: \exists x \in Tr(c): [ CC(c, m) \in PContain(c, x) ]. \]

**Safety Constraint**

A *safety constraint* is a set of states over the plant which excludes at least one hazard, and incorporates appropriate safety margins. In this case, the safety constraint is defined as the negation of the specified hazard, and re-written in terms of $PSection(c, x)$ assuming property **pro1**.

**SC$_{2,1}$** – Either the front of no primary train is in a danger zone $DZ(p, m)$ or the front of no secondary train is in the danger zone $DZ(s, m)$:

\[ \forall m \in M: \exists c \in L: \forall x \in Tr(c): [ PSection(c, x) \notin DZ(c, m) ]. \]

**Unsafe State**

An *unsafe* state is a state which could lead the system into a hazard state in the absence of corrective action and in the absence of subsequent initiating events (*initiating event* is an event to which a subsequent occurrence of a hazard could be attributed). If a state is not an unsafe state then it is said to be *safe*. In this case study, the unsafe state is defined as the negation of the specified safety constraint plus a certain margin; while the system is in an unsafe state, this margin allows for corrective action to be applied, in order to avoid the system to enter into a hazard state.

**US$_{2,1}$** – There is a primary train and a secondary train within $mnst + 1$ sections of the same crossing section.
\[ \exists m \in M: \forall c \in L: \exists x \in Tr(c): \]
\[ PSection(c, x) \in \{ CC(c, m) \oplus (mnst + 1), ..., CC(c, m) \oplus 1 \}. \]

For the purpose of the plant analysis, the following initial condition assumes that the trains are not in an unsafe state.

**pic1.** The position of all trains on circuit \( C(c) \) must not be within the danger zone \( DZ(c, m) \) or any of the \( mnst + 1 \) sections that precede \( DZ(c, m) \).

\[ T_0 = S(T) \Rightarrow \exists m \in M: \forall c \in L: \forall x \in Tr(c): \]
\[ PSection(c, x)(T_0) \notin \{ CC(c, m) \oplus (mnst + 1), ..., CC(c, m) \oplus 1 \}. \]

**Plant Safety Strategies**

A plant safety strategy is a scheme for maintaining a safety constraint, defined as a set of conditions imposed on controllable factors, over the plant.

In terms of the plant, two schemes are devised to maintain the safety constraint \( SC_{2,1} \). Both schemes are concerned with stopping a train in order to avoid hazard states. The analysis and specification of the activities related to restarting a stopped train are not considered in this paper.

**PSS_{2,1,1} – Reservation Scheme.**

**pssa.** If a train \( x \) is within \( mnst \) sections of \( CC(c, m) \oplus 1 \) or in the danger zone \( DZ(c, m) \) then the train is stopped if \( CC(c, m) \) is already reserved for a train on another circuit otherwise \( CC(c, m) \) is reserved for train \( x \).

\[ \forall m \in M: \forall c \in L: \forall x \in Tr(c): \]
\[ [ PSection(c, x) \in \{ CC(c, m) \oplus (mnst + 1), ..., CC(c, m) \oplus 1 \} \Rightarrow \]
\[ PAlreadyReserved(c, m) \Rightarrow PStop(c, x) \land \]
\[ \neg PAlreadyReserved(c, m) \Rightarrow CC(c, m) \in PReserved(c, x)]. \]

\[ \forall m \in M: \forall c \in L: \forall x \in Tr(c): \]
\[ \exists d \in L: (d \neq c \land \exists x \in Tr(d): CC(d, m) \in PReserved(d, x)). \]

**pssb.** Once a crossing section \( CC(c, m) \) is reserved for a train \( x \) it remains reserved while the train is within \( mnst \) sections of \( CC(c, m) \oplus 1 \) or in the danger zone \( DZ(c, m) \).

\[ \forall m \in M: \forall c \in L: \forall x \in Tr(c): \]
\[ [ CC(c, m) \in PReserved(c, x)(T_0) \land \forall t \in [T_0, T_1]: \]
\[ PSection(c, x)(t) \in \{ CC(c, m) \oplus (mnst + 1), ..., CC(c, m) \oplus 1 \} \Rightarrow \]
\[ \forall t \in [T_0, T_1]: CC(c, m) \in PReserved(c, x(t)). \]

**pssc.** Section \( CC(p, m) \) and section \( CC(s, m) \) cannot both be reserved.

\[ \forall m \in M: \exists c \in L: \forall x \in Tr(c): [ CC(c, m) \notin PReserved(c, x) ]. \]
PSS\	extsubscript{2,1,2} – Priority Scheme.

A primary train should not have to wait to enter a crossing section, and a secondary train must stop within 2mnst + 2 sections of the danger zone DZ(s, m) to allow a primary train to enter the crossing section, or travel through the crossing section (the additional mnst + 1 sections are needed to ensure that a secondary train does not reserve a crossing section under the reservation scheme).

\[ \forall m \in M: \forall x \in Tr(s): \]
\[ \{ PSection(c, x) \in \{ CC(s, m) \ominus (2mnst + 2), ..., CC(c, m) \ominus (mnst + 2) \} \land PNear(m) \Rightarrow PStop(s, x)(t) \}. \]

The system predicate \( PNear(m) \) is derived by analyzing the worst case time (i.e. the minimum time) taken by the primary train to enter the section \( CC(p, m) \ominus (mnst + 1) \) from the section \( PSection(p, x) \).

\[ \forall m \in M: PNear(m) \iff \exists x \in Tr(p): \]
\[ d_{min}(p, PSection(p, x), CC(p, m) \ominus (mnst + 1))/V_{max} \leq T_{Max}(m) \lor PSection(p, x) \in \{ CC(p, m) \ominus (mnst + 1), ..., CC(p, m) \oplus 1 \}. \]

\( T_{Max}(m) \) represents the worst case time (i.e. the maximum time) that a secondary train will take to travel from \( CC(s, m) \ominus (2mnst + 2) \) through the danger zone \( DZ(s, m) \), under rules of operation \textit{pro5} and \textit{pro6}, assuming the train does not stop because there is no secondary train ahead of it (\textit{pro4}).

\[ T_{Max}(m) = d_{max}(s, CC(s, m) \ominus (2mnst + 2), CC(s, m))/V_{min}. \]

3.2.3. Safety Analysis of the Plant Safety Specifications

The safety analysis of safety specifications of the train set is conducted by following the SSG for accident \( AC_2 \), shown in figure 2. From the SSG we infer that the safety strategies PSS\textsubscript{2,1,1} and PSS\textsubscript{2,1,2} are required to maintain the safety constraint \( SC_{2,1} \).

\[ \begin{align*}
\text{AC}_2 & \quad \downarrow \\
\text{HZ}_{2,1} & \quad \downarrow \\
\text{SC}_{2,1} & \quad \bigcirc \\
PSS_{2,1,1} & \quad \downarrow \\
\text{CSS}_{2,1,1,1} & \\
PSS_{2,1,2} & \quad \downarrow \\
\text{CSS}_{2,1,2,1} & \\
\end{align*} \]

Figure 2. Safety Specification Graph of Train Set Example

Instead of performing the safety analysis of the whole SSG, in the following we first present the vulnerability analysis of the safety constraint, and then we present both the preliminary
and vulnerability analysis of the plant safety strategy which is based on the reservation scheme.

Although the results of vulnerability analysis appear obvious once the appropriate logical relations between safety specifications are obtained, the process of performing the preliminary and vulnerability analysis is rather iterative. In the example that follows, on several occasions, we had to modify the logical relations (either the safety strategies or the assumptions) in order to resolve inconsistencies identified by the vulnerability analysis.

3.2.3.1. Vulnerability Analysis of the Safety Constraint

As previously mentioned, the vulnerability analysis aims to identify those circumstances which can lead to the violation of the safety specifications. In this section, we perform the vulnerability analysis of the safety constraints by applying the FTA technique.

From the SSG of figure 2, the first safety specification to be analysed is the safety constraint SC$_{2,1}$. Starting from the undesired event, which in this case is the hazard HZ$_{2,1}$, the aim is to identify those assumptions that if violated can cause the system to enter into a hazard state despite the associated safety constraint being maintained. The result of this analysis is the fault tree shown in figure 3, which has the following logical relation:

$$\text{HZ}_{2,1} \Rightarrow (\neg \text{SC}_{2,1} \lor \neg \text{pro}l)$$

![Fault Tree of SC$_{2,1}$](image)

This logical relation shows that if rule of operation pro$l$ is violated, the system can enter into a hazard state without violating the safety constraint.

3.2.3.2. Preliminary Analysis of the Plant Safety Strategy

By inspecting the SSG, we can infer that to conduct the preliminary analysis of the safety strategies we must confirm one horizontal check (i.e. between PSS$_{2,1,1}$ and PSS$_{2,1,2}$) and two vertical checks (i.e. between SC$_{2,1}$ and PSS$_{2,1,1}$, and between SC$_{2,1}$ and PSS$_{2,1,2}$). To demonstrate the preliminary analysis, we focus on those checks involving PSS$_{2,1,1}$. 

13
Horizontal Check

The *horizontal check* between the reservation scheme plant safety strategy PSS$_{2,1,1}$ and priority scheme plant safety strategy PSS$_{2,1,2}$ was conducted by stating and confirming the following lemma.

**Lemma 3.1.** The plant safety strategy PSS$_{2,1,2}$ is not in conflict with the plant safety strategy PSS$_{2,1,1}$, under rules of operation *pro2, pro3, pro4, pro5* and *pro6*.

$$\neg (PSS_{2,1,1} \land PSS_{2,1,2} \land \text{pro2} \land \text{pro3} \land \text{pro4} \land \text{pro5} \land \text{pro6} \Rightarrow \text{FALSE}).$$

**Proof.** (See Appendix A.)

Vertical Check

The *vertical check* between the reservation scheme plant safety strategy PSS$_{2,1,1}$ and safety constraint SC$_{2,1}$ was conducted by stating and confirming the following lemma.

**Lemma 3.2.** A history that is not in an unsafe state, or satisfies the plant safety strategy PSS$_{2,1,1}$ and rules of operation *pro2* and *pro3*, must satisfy safety constraint SC$_{2,1}$.

$$\neg \text{US}_{2,1} \lor (PSS_{2,1,1} \land \text{pro2} \land \text{pro3}) \Rightarrow \text{SC}_{2,1}.$$  

**Proof.** (See Appendix A.)

3.2.3.3. Vulnerability Analysis of the Plant Safety Strategy

The vulnerability analysis for the reservation scheme plant safety strategy (PSS$_{2,1,1}$) identifies those circumstances which are necessary for the violation of the safety constraint SC$_{2,1}$ while the system is in unsafe state, namely, the violation of the plant safety strategy or the violation of the rules of operation *pro2* and *pro3*. This is captured by the following relation:

$$\neg \text{SC}_{2,1} \Rightarrow \text{US}_{2,1} \land (\neg PSS_{2,1,1} \lor \neg \text{pro2} \lor \neg \text{pro3}).$$

This logical relation describes the fault tree, presented in figure 4. The fault trees of figures 3 and 4 can be combined through the box representing the violation of the safety constraint ($\neg \text{SC}_{2,1}$).

If we had presented the vulnerability analysis for the priority scheme plant safety strategy (PSS$_{2,1,2}$), it would have shown that the priority scheme relies upon stronger assumptions than the reservation scheme. Hence, more confidence can be placed in the reservation scheme. On the other hand, from the perspective of the mission, only the priority scheme preserves the priority of the primary trains. A consequence of the safety analysis is that the two strategies are complementary (indicated by the $\odot$). Normally SC$_{2,1}$ will be maintained by the priority scheme, however, the reservation scheme will have an effect only when either PSS$_{2,1,2}$ or one of the rules of operation *pro2, pro3, pro4, pro5* and *pro6* is violated.
3.3. Plant Interface Analysis

In this section, we present the model of the plant interface in terms of the properties of sensors and actuators, comprising the specification of their standard and failure behaviours.

Model of the Sensors and Actuators

At the beginning of every section there is a sensor that detects the presence of a train, and for each train there is an actuator that can stop the train within a fixed number of sections. The behaviour of the train in terms of these sensors and actuators is represented by the following system variables.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Anticipated Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActIn(c, x)</td>
<td>{true, false}</td>
</tr>
<tr>
<td>ActOut(c, x)</td>
<td>{true, false}</td>
</tr>
<tr>
<td>SensIn(c, i)</td>
<td>{true, false}</td>
</tr>
<tr>
<td>SensOut(c, i)</td>
<td>{true, false}</td>
</tr>
</tbody>
</table>

Table 5. Variables for Plant Interface Model
ActIn\(c, x\) represents the input signal of an actuator, ActOut\(c, x\) represents the output signal of an actuator, SensIn\(c, i\) represents the input signal of a sensor, and SensOut\(c, i\) represents the output signal of the sensor.

The constant associated with the behaviour of sensors is presented in the table below.

<table>
<thead>
<tr>
<th>Constants</th>
<th>Range</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcsf</td>
<td>(S(c))</td>
<td>The maximum number of consecutive sensor failures.</td>
</tr>
</tbody>
</table>

Table 6. Constants for Plant Interface Model

**Sensor Relations**

In its standard behaviour, the sensor at the beginning of section \(i\) signals true iff a train is in section \(i\) on circuit \(C(c)\).

**sr1.** The standard behaviour of a sensor is represented by the following predicate.

\[
\forall c \in L: \forall i \in S(c): [\text{SensOut}(c, i) \iff \text{SensIn}(c, i)].
\]

When a sensor fails, either it never detects the presence of a train — “miss a train”, or it always detects the presence of a train — “ghost train”.

**sr2.** The failure behaviour of a sensor in not detecting the presence of a train is represented by the following predicate.

\[
\exists c \in L: \exists i \in S(c): \neg \text{SensOut}(c, i) \land \text{SensIn}(c, i).
\]

**sr3.** The failure behaviour of a sensor in detecting the presence of a “ghost train” is represented by the following relation.

\[
\exists c \in L: \exists i \in S(c): \text{SensOut}(c, i) \land \neg \text{SensIn}(c, i).
\]

In the following, we concentrate on sensor failures that “miss a train” because the presence of “ghost trains” should only affect the mission of the system.

**Actuator Relations**

In the train set, an actuator stops a train by stopping the feeding power to the electrical motor of the train. There is no alternative mechanism, such as brakes, to enforce rule of operation pro3. If the train takes more than mnsi sections to stop then pro3 is violated; depending on the risks involved this will require a re-evaluation of PSS2,1,1. The behaviour of an actuator can be described by the following three relations.

**ac1.** The standard behaviour of an actuator is represented by the following relation.

\[
\forall c \in L: \forall x \in T(c): [\text{ActOut}(c, x) \iff \text{ActIn}(c, x)].
\]

**ac2.** The failure of an actuator in stopping a train inadvertently is represented by the following relation.
\[ \exists c \in L : \exists x \in Tr(c) : ActOut(c, x) \land \neg ActIn(c, x). \]

\( ac3 \). The failure of an actuator of not stopping a train when required to do so is represented by the following relation.

\[ \exists c \in L : \exists x \in Tr(c) : \neg ActOut(c, x) \land ActIn(c, x). \]

In the context of the train set, only the actuator relation \( ac3 \) adversely affects safety, and the only feasible strategy to tolerate such faults, without changing the basic design of a train, is to shut down the whole system [15]. Assuming that an actuator is capable of maintaining the rule of operation \( pro3 \), the analysis conducted below will not be affected by the above actuator relations.

### 3.4. Controlling System Analysis

The properties of sensors and actuators defined in the previous section are used to derive the controlling system safety strategies from the plant safety strategies. The safety analysis of a controlling system safety strategy is performed by verifying that it complies with its respective plant safety strategy, and by identifying those circumstances under which that controlling system safety strategy is unable to maintain safe behaviour.

#### 3.4.1. Model of the Controlling System

The behaviour of the trains, as perceived by the controlling system, is captured by the variables defined in the table below.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Anticipated Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_{\text{Reserved}}(c, x) )</td>
<td>( p(S(c)) )</td>
</tr>
<tr>
<td>( C_{\text{Section}}(c, x) )</td>
<td>( S(c) )</td>
</tr>
<tr>
<td>( C_{\text{Stop}}(c, x) )</td>
<td>{true, false}</td>
</tr>
</tbody>
</table>

Table 7. Variables for Controlling System Model

The interpretation of these variables in terms of the controlling system follows the meaning given in the plant analysis. \( C_{\text{Reserved}}(c, x) \) denotes the set of sections reserved for train \( x \) on circuit \( C(c) \), \( C_{\text{Section}}(c, x) \) uniquely denotes the section number containing the front of train \( x \) on circuit \( C(c) \), and \( C_{\text{Stop}}(c, x) \) denotes whether the controlling system has requested the actuator to stop train \( x \) on circuit \( C(c) \).

**Rules of Operation**

The following rules of operation describe the behaviour of the train set in terms of the controlling system.

\( cro1 \). The following two relations map the properties of a sensor into the variables of the plant and controlling system.
∀c ∈ L: ∀i ∈ S(c): [SensIn(c, i) ⇔ ∃x ∈ Tr(c): PSection(c, x)=i].

∀c ∈ L: ∀i ∈ S(c): [SensOut(c, i) ⇔ ∃x ∈ Tr(c): CSection(c, x)=i].

cro2. The following two relations map the properties of an actuator into the variables of the plant and controlling system.

∀c ∈ L: ∀x ∈ Tr(c): [ActIn(c, x) ⇔ CStop(c, x)].

∀c ∈ L: ∀x ∈ Tr(c): [ActOut(c, x) ⇔ PStop(c, x)].

cro3. From the sensor relations sr1 and sr2 (assuming that mcsf is the maximum number of consecutive sensor failures), and the two relations of cro1, we can infer that the position of a train as recorded by CSection is at most mcsf sections behind the position recorded by PSection.

∀c ∈ L: ∀x ∈ Tr(c):
    [CSection(c, x) ∈ \{PSection(c, x) \dot{\oplus} mcsf, ..., PSection(c, x)\}].

cro4. From rules of operation pro2 and cro3 we infer that the position of a train as recorded by CSection can jump by at most mcsf sections.

∀c ∈ L: ∀x ∈ Tr(c):
    [CSection(c, x)(T_0) \neq CSection(c, x)(T_1) ⇒ \exists t ∈ [T_0, T_1]: CSection(c, x)(t) ∈
    \{CSection(c, x)(T_0) \dot{\oplus} 1, ..., CSection(c, x)(T_0) \dot{\oplus} mcsf\}].

cro5. From the actuator relation ac1 and the two relations of cro2 we can infer that the train is stopped if and only if the controlling system has commanded the train to stop.

∀c ∈ L: ∀x ∈ Tr(c): PStop(c, x) ⇔ CStop(c, x).

cro6. From pro3, the controlling system assumes that trains require at most mcsf sections to stop (for mcsf ≥ mnst, otherwise remains similar to pro3).

∀c ∈ L: ∀x ∈ Tr(c): [∀t ∈ [T_0, T_1]: CStop(c, x)(t) ⇒ CSection(c, x)(T_1) ∈
    \{CSection(c, x)(T_0), ..., CSection(c, x)(T_0) \dot{\oplus} (mcsf + 1)\}].

cro7. A crossing section CC(c, m) is reserved for a train x at the controlling system level if and only if it is reserved for that train at the plant level.

∀m ∈ M: ∀c ∈ L: ∀x ∈ Tr(c):
    CC(c, m) ∈ CReserved(c, x) ⇔ CC(c, m) ∈ PReserved(c, x).

The unsafe state of the plant, previously defined, has also to be redefined in accordance with the rule of operation cro3.

US_{2,1}’ – There is a primary train and a secondary train within mcsf + mnst + 1 sections of the same crossing section.
\[ \exists m \in M: \forall c \in L: \exists x \in Tr(c): \]
\[ \text{CSection}(c, x) \in \{ CC(c, m) \ominus (mcsf + mnst + 1), \ldots, CC(c, m) \oplus 1 \}. \]

The initial condition previously defined for the plant has to be redefined in order to consider failures in the sensors. Also, another initial condition has to be introduced which enforces consistency between the states of the plant and the readings provided by the sensors.

**cic1.** The position of all trains as recorded by \text{CSection} must not be within the danger zone \( DZ(c, m) \) plus all the \((mnst + 1)\) sections that precede \( DZ(c, m) \).

\[ T_0 = S(T) \Rightarrow \forall c \in L: \forall x \in Tr(c): \]
\[ \text{CSection}(c, x)(T_0) \notin \{ CC(c, m) \ominus (mnst + 1), \ldots, CC(c, m) \oplus 1 \}. \]

**cic2.** The position of all trains as recorded by \text{CSection} should be equal to the actual position of the trains, as recorded by \text{PSection}.

\[ T_0 = S(T) \Rightarrow \forall c \in L: \forall x \in Tr(c): [\text{CSection}(c, x)(T_0) = \text{PSection}(c, x)(T_0)]. \]

### 3.4.2. Controlling System Safety Specifications

In the following, we focus on the controlling system safety strategy based on the reservation scheme, which is derived from the plant safety strategy \text{PSS}_{2,1,1} by taking into consideration the rules of operation \( \text{cro3} \) and \( \text{cro5} \).

**Controlling System Safety Strategy**

\text{CSS}_{2,1,1,1} \rightarrow \text{Reservation Scheme}.

A **controlling system safety strategy** is a refinement of the respective plant safety strategy, incorporating properties of sensors and actuators.

**cssa.** If the recorded position of a train \( x \) on circuit \( C(c) \) is within \( mcsf + mnst \) sections of \( CC(c, m) \ominus 1 \) then the train is stopped if \( CC(c, m) \) is already reserved for a train on another circuit, otherwise \( CC(c, m) \) is reserved for train \( x \).

\[ \forall m \in M: \forall c \in L: \forall x \in Tr(c): \]
\[ [\text{CSection}(c, x) \in \{ CC(c, m) \ominus (mcsf + mnst + 1), \ldots, CC(c, m) \oplus 1 \} \Rightarrow \]
\[ \text{CAlreadyReserved}(c, m) \Rightarrow \text{CStop}(c, x) \land \]
\[ \neg \text{CAlreadyReserved}(c, m) \Rightarrow CC(c, m) \in \text{CReserved}(c, x)]. \]

\[ \forall c \in L: [\text{CAlreadyReserved}(c, m) \leftrightarrow \]
\[ \exists d \in L: (d \neq c \land \exists x \in Tr(d): CC(d, m) \in \text{CReserved}(d, x))]. \]

**cssb.** Once a crossing section \( CC(c, m) \) is reserved for a train \( x \) it remains reserved while the recorded position of the train is within \( mcsf + mnst \) sections of \( CC(c, m) \ominus 1 \) or in the danger zone \( DZ(c, m) \).

\[ \forall m \in M: \forall c \in L: \forall x \in Tr(c): \]
\[ [CC(c, m) \in \text{CReserved}(c, x)(T_0) \land \forall t \in [T_0, T_1]: \]
\[ C_{Section}(c, x)(t) \subseteq \{ CC(c, m) \oplus (mcst + mnst + 1), ..., CC(c, m) \oplus 1 \} \Rightarrow \\
\forall t \in [T_0, T_1]: CC(c, m) \subseteq C_{Reserved}(c, x)(t) . \]

cssc. Section \( CC(p, m) \) and section \( CC(s, m) \) cannot both be reserved;

\[ \forall m \in M: \exists c \in L: \forall x \in Tr(c): [ CC(c, m) \notin C_{Reserved}(c, x) ] . \]

3.4.3. Safety Analysis of the Controlling System Safety Specifications

3.4.3.1. Preliminary Analysis of the Controlling System Safety Strategy

By inspecting the SSG, we can infer that to conduct the preliminary analysis of the controlling system safety strategies we must confirm one horizontal check (i.e. between CSS\(_{2,1,1,1}\) and CSS\(_{2,1,2,1}\)) and two vertical checks (i.e. between PSS\(_{2,1,1}\) and CSS\(_{2,1,1,1}\), and between PSS\(_{2,1,2}\) and CSS\(_{2,1,2,1}\)). Following the preliminary analysis of the plant specifications, we only confirm the vertical check involving PSS\(_{2,1,1}\).

**Vertical Check**

The vertical check between the reservation scheme controlling system safety strategy CSS\(_{2,1,1,1}\) and the reservation scheme plant safety strategy PSS\(_{2,1,1}\) was conducted by stating and confirming the following lemma.

**Lemma 3.3.** A history that satisfies the controlling system safety strategy CSS\(_{2,1,1,1}\), and rules of operation \( cro3, cro5 \) and \( cro7 \) must satisfy plant safety strategy PSS\(_{2,1,1}\).

\[ CSS_{2,1,1,1} \land cro3 \land cro5 \land cro7 \Rightarrow PSS_{2,1,1} . \]

**Proof.** (See Appendix A.)

3.4.3.2. Vulnerability Analysis of the Controlling System Safety Strategy

The vulnerability analysis of the controlling system safety strategy (CSS\(_{2,1,1,1}\)) follows from that conducted for the plant safety strategy (PSS\(_{2,1,1}\)). The logical relation that describes the fault tree, shown in figure 5, is the following:

\[ \neg PSS_{2,1,1} \Rightarrow \neg CSS_{2,1,1} \lor \neg cro3 \lor \neg cro5 \lor \neg cro7 . \]

The fault trees of figures 4 and 5 can also be combined by replacing, in figure 4, the subtree representing the violation of the plant safety strategy (\( \neg PSS_{2,1,1} \)) by the fault tree of figure 5.

3.5. Discussion of the Train Set Example

In this paper, the safety analysis conducted of the safety specifications has identified many misconceptions in previous solutions of the crossing section problem [3]. This was primarily due to the fact that earlier work had focused on deriving the safety specifications and confirming their consistency within a formal model of the train set. Comparing the
presented approach with earlier work, we are able to identify two limitations: the assumptions were neither formally recorded nor their impact on the safety specifications determined, and the validation of the formal model of the train set was not adequately conducted.

The safety analysis of the plant safety specifications was able to confirm whether all relevant assumptions over the behaviour of the plant (i.e rules of operation and initial conditions) were considered and recorded. As a result the reservation scheme plant safety strategy was revised to take into account the movement of trains (pro2) and the distance required to stop a train (pro3). During the plant interface analysis, the properties of the sensors and actuators are identified, in terms of their standard and failure behaviours. These properties are used during the controlling system analysis, to derive the controller safety strategy from the plant safety strategy, by taking into account the uncertainty associated with the position of a train (cro3).

An advantage in explicitly documenting the assumptions as part of a safety specification is that it enables the impact of changes in assumptions to be traced through the SSG. For example, if we consider a change in the assumption on the distance required to stop a train (pro3), this will not affect the analysis conducted on the safety constraint, however, rules pssa and pssb of the reservation scheme plant safety strategy must be updated and consequently rules cssa and cssb of the controlling system safety strategy will also need to be modified.
The rules of operation in the plant domain are considered as assumptions in the controlling system domain and can be used to specify tests to detect exceptional circumstances (i.e. those circumstances under which a safety specification may not maintain safe behaviour) and to define appropriate safety precautions. For example, an additional rule could be added to the controlling system safety strategy that performs a system shut down (i.e. “stops all trains”) if a violation of rule cro5 is detected [15].

4. Concluding Remarks

Safety analysis has played a pivotal role in the development of traditional safety-critical systems, in assessing risk and guiding system development. Recently, it has been recognised that software development should also be subjected to safety analysis, however, unlike traditional systems, problems still exist in selecting the appropriate methods and techniques to effectively conduct that analysis.

Instead of conducting safety analysis at the later stages of software development, in this paper we have presented a novel approach in which safety analysis is conducted in parallel with the production of the software safety specifications. A major advantage in integrating safety analysis with the process of requirements analysis is that the whole process of conducting the analysis of the safety requirements becomes more cost effective because the safety analysis is conducted as early as possible, facilitating immediate feedback of its results into the software development process.

In future work, we intend to apply, apart from FTA, other traditional safety analysis techniques in the process of conducting vulnerability analysis of the safety specifications.

Acknowledgements

The authors would like to acknowledge the financial support of the EPSRC (UK) SCHEMA project, and the EC COPERNICUS Joint Research Project ISAT.

References


Appendix A — Proofs of the Lemmas

Lemma 3.1. The plant safety strategy PSS_{2,1,2} is not in conflict with the plant safety strategy PSS_{2,1,1}, under rules of operation pro2, pro3, pro4, pro5 and pro6.

Proof. The two strategies will not conflict if the priority scheme allows a secondary train to be in the set of sections \{CC(s, m) \oplus (mnst + 1), ..., CC(s, m) \oplus 1\} only if no primary train is in the set of sections \{CC(p, m) \oplus (mnst + 1), ..., CC(p, m) \oplus 1\}. This is expressed as assertion 1.

1. \( \forall m \in M: \forall y \in \text{Tr}(s): \) 
   
   \[ \{P\text{Section}(s, y) \in \{CC(s, m) \oplus (mnst + 1), ..., CC(s, m) \oplus 1\} \Rightarrow \] 
   
   \[ \forall x \in \text{Tr}(p): P\text{Section}(p, x) \notin \{CC(p, m) \oplus (mnst + 1), ..., CC(p, m) \oplus 1\} \} \].

2. \( \forall m \in M: \forall y \in \text{Tr}(s): P\text{Section}(s, y)(T_0) = CC(s, m) \oplus (mnst + mnst + 2) \land \) 
   
   \[ \forall t \in [T_0, T_1]: [P\text{Near}(m)(t) \Rightarrow \) 
   
   \[ P\text{Section}(s, y)(t) \notin \{CC(s, m) \oplus (mnst + 1), ..., CC(s, m) \oplus 1\} \].

2.1. Assume:

   \[ \exists m \in M: \exists y \in \text{Tr}(s): \exists T_0, T_1 \in T: \] 
   
   \[ P\text{Section}(s, y)(T_0) = CC(s, m) \oplus (mnst + mnst + 2) \land \forall t \in [T_0, T_1]: P\text{Near}(m)(t). \]

2.2. \( P\text{Section}(s, y)(T_0) = CC(s, m) \oplus (mnst + mnst + 2) \land \forall t \in [T_0, T_1]: P\text{Stop}(s, y)(t), \) 
   
   from 2.1 and the priority scheme.

2.3. \( \forall t \in [T_0, T_1]: \) 
   
   \[ P\text{Section}(s, y)(t) \in \{CC(c, m) \oplus (mnst + mnst + 2), ..., CC(c, m) \oplus (mnst + 2)\}, \] 
   
   from 2.2 and pro3.

2.4. \( \forall t \in [T_0, T_1]: P\text{Section}(s, y)(t) \notin \{CC(s, m) \oplus (mnst + 1), ..., CC(s, m) \oplus 1\}, \) 
   
   from 2.3.

3. Assume:

   \[ \exists m \in M: \exists y \in \text{Tr}(s): \] 
   
   \[ \exists t \in T: P\text{Section}(s, y)(t) \in \{CC(s, m) \oplus (mnst + 1), ..., CC(s, m) \oplus 1\}. \]

3.1. \( \exists t_1 \in T: t_1 > t - T\text{Max}(m): \) 
   
   \[ P\text{Section}(s, y)(t_1) \in \{CC(c, m) \oplus (mnst + mnst + 2), ..., CC(c, m) \oplus (mnst + 2)\} \land \] 
   
   \[ \neg P\text{Near}(m)(t_1), \] 
   
   from 2, definition of \( T\text{Max}(m), \) pro4, pro5 and pro6.

3.2. \( \forall t_2 \in [t_1, t_1 + T\text{Max}(m)]: \forall x \in \text{Tr}(p): \) 
   
   \[ P\text{Section}(p, x)(t_2) \notin \{CC(p, m) \oplus (mnst + 1), ..., CC(p, m) \oplus 1\}, \] 
   
   from 3.1 and definition of \( P\text{Near}(m). \)

3.3. \( P\text{Section}(s, y)(t) \in \{CC(s, m) \oplus (mnst + 1), ..., CC(s, m) \oplus 1\} \Rightarrow \) 
   
   \[ \forall x \in \text{Tr}(p): P\text{Section}(p, x)(t) \notin \{CC(p, m) \oplus (mnst + 1), ..., CC(p, m) \oplus 1\}, \] 
   
   from 3.1 and 3.2. This confirms assertion 1.
Lemma 3.2. A history that satisfies the plant safety strategy PSS$^2_{2,1,1}$, rules of operation pro2 and pro3, and initial condition pic1 must satisfy safety constraint SC$^2_{2,1}$.

\[ \neg \text{US}^2_{2,1} \lor (\text{PSS}^2_{2,1,1} \land \text{pro2} \land \text{pro3}) \Rightarrow \text{SC}^2_{2,1}. \]

Proof. The proof is presented in three parts.

A. pic1 $\Rightarrow$ SC$^2_{2,1}$

The above follows directly from pic1.

B. \( \neg \text{US}^2_{2,1} \Rightarrow \text{SC}^2_{2,1} \)

The above follows directly from the definitions of US$^2_{2,1}$ and SC$^2_{2,1}$, respectively.

C. pssa $\land$ pssb $\land$ pssc $\land$ pro2 $\land$ pro3 $\Rightarrow$ SC$^2_{2,1}$

Firstly, we prove that if a train is in the danger zone DZ(c, m), CC(c,m) is reserved for the train.

1. \( \forall m \in M: \forall c \in L: \forall t \in \text{Tr}^c: \text{PSection}(c, x) \in DZ(c, m) \Rightarrow \)

\( CC(c, m) \in \text{PReserved}(c, x) \).

Assertion 1 is proven by contradiction

1.1. Assume: \( \exists m \in M: \exists c \in L: \exists t \in \text{Tr}^c: \exists t_2 \in T: \)

\( \text{PSection}(c, x)(t_2) \in DZ(c, m) \land CC(c, m) \not\in \text{PReserved}(c, x)(t_2) \).

Examine two cases: 1.2. \( \neg \text{PAlreadyReserved}(c, m)(t_2) \), and

1.3. \( \text{PAlreadyReserved}(c, m)(t_2) \).

1.2. \( \text{PSection}(c, x)(t_2) \in DZ(c, m) \land \neg \text{PAlreadyReserved}(c, m)(t_2) \Rightarrow \)

\( CC(c, m) \in \text{PReserved}(c, x)(t_2) \), from pssa. This contradicts 1.1.

1.3. \( \text{PAlreadyReserved}(c, m)(t_2) \Rightarrow \text{PSection}(c, x)(t_2) \not\in DZ(c, m) \)

Assertion 1.3 is proven by contradiction.

1.3.1. Assume: \( \text{PAlreadyReserved}(c, m)(t_2) \land \text{PSection}(c, x)(t_2) \in DZ(c, m) \).

1.3.2. \( \exists t_1 \in T: t_1 < t_2 \land \text{PSection}(c, x)(t_1) = CC(c, m) \oplus (mnst + 1) \land \)

\( \forall t_3 \in [t_1, t_2]: \text{PSection}(c, x)(t_3) \in \{CC(c, m) \oplus (mnst + 1), ..., CC(c, m) \oplus 1\}, \)

from 1.3.1, pro2 and pic1.

1.3.3. \( CC(c, m) \not\in \text{PReserved}(c, x)(t_2) \), from 1.3.1 and pssc.

1.3.4. \( \forall t_3 \in [t_1, t_2]: CC(c, m) \not\in \text{PReserved}(c, x)(t_3) \land \)

\( \text{PSection}(c, x)(t_3) \in \{CC(c, m) \oplus (mnst + 1), ..., CC(c, m) \oplus 1\}, \)

from 1.3.2, 1.3.3, and pssb.

1.3.5. \( \forall t_3 \in [t_1, t_2]: \text{PAlreadyReserved}(c, m)(t_3) \land \)

\( \text{PSection}(c, x)(t_3) \in \{CC(c, m) \oplus (mnst + 1), ..., CC(c, m) \oplus 1\}, \)

from 1.3.4 and pssa.

1.3.6. \( \forall t_3 \in [t_1, t_2]: \text{PStop}(c, x)(t_3) \land \)

\( \text{PSection}(c, x)(t_3) \in \{CC(c, m) \oplus (mnst + 1), ..., CC(c, m) \oplus 1\}, \)

25
from 1.3.5 and \( pssa \).

1.3.7. \( \forall t_3 \in [t_1, t_2]: PSection(c, x)(t_3) \in \{ CC(c, m) \ominus (mnst + 1), ..., CC(c, m) \oplus 1 \} \), from 1.3.6 and \( pro3 \).

This contradicts 1.3.1 confirming 1.3.

1.4. \( 1.2 \land 1.3 \Rightarrow PSection(c, x)(t_2) \notin DZ(c, m) \lor CC(c, m) \in PReserved(c, x)(t_2) \).

This contradicts 1.1 confirming 1.

Secondly, we prove that there can only be one type of train in the danger zone.

2. \( \forall m \in M: \exists c \in L: \forall x \in Tr(c): [ PSection(c, x) \notin DZ(c, m) ] \).

Assertion 2 is proven by contradiction.

2.1. Assume: \( \exists t \in T: \exists m \in M: \)

\( (\exists x \in Tr(p): PSection(p, y)(t) \in DZ(c, m) \land \exists y \in Tr(s): PSection(s, y)(t) \in DZ(c, m) ) \)

2.2. \( CC(p, m) \in PReserved(p, x)(t) \land CC(s, m) \in PReserved(s, y)(t) \), from 1.

This contradicts \( pssc \), confirming 2.

**Lemma 3.3.** A history that satisfies the controlling system safety strategy \( CSS_{2,1,1,1} \), rules of operation \( cro3 \), \( cro5 \) and \( cro7 \), and the initial conditions \( cic1 \) and \( cic2 \), must satisfy plant safety strategy \( PSS_{2,1,1} \).

\[
CSS_{2,1,1,1} \land cro3 \land cro5 \land cro7 \Rightarrow PSS_{2,1,1}.
\]

**Proof.** The proof is presented in four parts.

**A.** \( cic1 \land cic2 \Rightarrow PSS_{2,1,1} \).

From these initial conditions we infer that \( SC_{2,1} \) is not violated which implies that \( PSS_{2,1,1} \) is maintained.

**B.** \( cssa \land cro3 \land cro5 \land cro7 \Rightarrow pssa \).

Proof is conducted by examining two cases.

1. \( CSection(c, x) \notin \{ CC(c, m) \ominus (mcsf + mnst + 1), ..., CC(c, m) \oplus 1 \} \)

1.1. \( PSection(c, x) \notin CC(c, m) \ominus (mnst + 1), ..., CC(c, m) \oplus 1 \), from \( cro3 \) and 1

1.2. \( 1.1 \Rightarrow pssa \).

2. \( CSection(c, x) \in \{ CC(c, m) \ominus (mcsf + mnst + 1), ..., CC(c, m) \oplus 1 \} \)

2.1. \( PAAlreadyReserved(c, m) \Rightarrow PStop(c, x) \)

2.1.1. \( CAAlreadyReserved(c, m) \Rightarrow CStop(c, x) \), from 2 and \( cssa \).

2.1.2. \( CAAlreadyReserved(c, m) \Leftrightarrow PAAlreadyReserved(c, m) \), from \( cro7 \).

2.1.3. \( CStop(c, x) \Rightarrow PStop(c, x) \), from \( cro5 \).

2.1.4. \( 2.1.1 \land 2.1.2 \land 2.1.3 \Rightarrow 2.1 \).

2.2. \( \neg PAAlreadyReserved(c, m) \Rightarrow CC(c, m) \in PReserved(c, x) \)

2.2.1. \( \neg CAAlreadyReserved(c, m) \Rightarrow CC(c, m) \in CReserved(c, x) \), from 2 and \( cssa \).

2.2.2. \( \neg CAAlreadyReserved(c, m) \Leftrightarrow \neg PAAlreadyReserved(c, m) \), from \( cro7 \).
2.2.3. \( CC(c, m) \in CReserved(c, x) \Rightarrow CC(c, m) \in PReserved(c, x) \), from cr07.

2.2.4. \( 2.2.1 \land 2.2.2 \land 2.2.3 \Rightarrow 2.2. \)

2.3. \( 2.1 \land 2.2 \Rightarrow pssa. \)

C. \( cssb \land cro3 \land cro7 \Rightarrow psb. \)

Proof is conducted by examining two cases.

1. \( CC(c, m) \notin CReserved(c, x)(T_0) \lor \exists t \in [T_0, T_1]: CSection(c, x)(t) \notin \{CC(c, m)\oplus(mcsf + mnst + 1), ..., CC(c, m)\oplus1\}. \)

1.1. \( CC(c, m) \notin PReserved(c, x)(T_0) \lor CC(c, m) \in CReserved(c, x)(T_0), \) from cro7.

1.2. \( \exists t \in [T_0, T_1]: PSection(c, x)(t) \notin \{CC(c, m)\oplus(mnst + 1), ..., CC(c, m)\oplus1\} \lor \forall t \in [T_0, T_1]: CSection(c, x)(t) \in \{CC(c, m)\oplus(mcsf + mnst + 1), ..., CC(c, m)\oplus1\}, \) from cro3.

1.3. \( CC(c, m) \notin PReserved(c, x)(T_0) \lor \exists t \in [T_0, T_1]: PSection(c, x)(t) \notin \{CC(c, m)\oplus(mnst + 1), ..., CC(c, m)\oplus1\}, \) from 1.1 and 1.2.

1.4. \( 1.3 \Rightarrow psb. \)

2. \( CC(c, m) \in CReserved(c, x)(T_0) \land \forall t \in [T_0, T_1]: CSection(c, x)(t) \in \{CC(c, m)\oplus(mcsf + mnst + 1), ..., CC(c, m)\oplus1\}. \)

2.1. \( \forall t \in [T_0, T_1]: CC(c, m) \in CReserved(c, x)(t), \) from cssb.

2.2. \( \forall t \in [T_0, T_1]: CC(c, m) \in PReserved(c, x)(t), \) from cro7.

2.3. \( 2.2 \Rightarrow psb. \)

D. \( cssc \land cro7 \Rightarrow psc. \)

Follows directly from cro7.

Appendix B – Timed History Logic (THL)

THL is a formalism based on the time domain \((\mathbb{R}_+, \prec, 0)\) and consists of three main concepts: histories, relations and modes. Here we present an overview of histories and relations; a more detailed description of the model is given elsewhere /Saeed 90/.

For a system with \( n \) state variables we have the state vector: \( Sv = \langle p_1, ..., p_n \rangle \), the range of a variable is denoted by \( \mathbf{Vp}_i \). The state space of a system \( (\Gamma) \) is defined as the cross product of the variable ranges.

The semantics of THL are defined in terms of histories. A history \( H \) of a system is a function of the form \( H: T \rightarrow \Gamma \). The set of all “possible” histories of a system is defined as the universal history set \( \Gamma H \) (i.e. the set of all functions \( H: T \rightarrow \Gamma \)). For a history \( H \) the sequence of values taken by a state variable \( p_i \) is denoted by the function \( H.p_i: T \rightarrow \mathbf{Vp}_i \).

Invariant relations are used to express relationships over the state variables which hold at every time point within \( T \); these are formulated as system predicates.
A tuple of values \( V = (x_1, ..., x_n) \), where \( x_i \) is of type \( \mathbf{Vp}_i \), satisfies a system predicate \( P \) if and only if substitution of each \( x_i \) for \( p_i \) within \( P(x_1, ..., x_n) \) evaluates to \text{true}. This is denoted by: \( P(V) \). We will denote \( P(H,p_1(t), ..., H,p_n(t)) \) by \( H \text{ sat } P@t \). A system predicate \( P \) describes an invariant relation for a history \( H \) iff: \( \forall t \in T: H \text{ sat } P@t \).

History relations are used to express relationships over the state variables which hold during every interval included within \( T \); these are formulated as history predicates.

A history predicate is a predicate built using two free time variables \( T_0, T_f \) and \( n \) free function variables \( p_1, ..., p_n \). No other free variables may be used. A history \( H \) satisfies a history predicate \( HP \) for an interval \( \text{Int} \) if and only if the expression resulting from substituting: i) \( s(\text{Int}) \) for \( T_0 \), ii) \( e(\text{Int}) \) for \( T_f \) and iii) \( H,p_i \) for \( p_i \) for all \( i \), evaluates to \text{true}. This is denoted by: \( H \text{ sat } HP@\text{Int} \). A history predicate \( HP \) describes a history relation for a history \( H \) iff: \( \forall \text{Int} \subseteq T: H \text{ sat } HP@\text{Int} \).