The Role of Formal Methods in the Requirements Analysis of Safety-Critical Systems: a Train Set Example

A. Saeed, R. de Lemos and T. Anderson
The Role of Formal Methods in the Requirements Analysis of Safety-Critical Systems: a Train Set Example

A. Saeed, R. de Lemos and T. Anderson

Abstract

Requirements analysis plays a vital role in the development of safety-critical systems since any errors in the requirements specification will corrupt the subsequent stages of system development. Experience in safety-critical systems has shown that requirements specification errors can and do cause accidents. This paper presents a general framework for the formal specification and verification of the critical requirements in the development of safety-critical systems. The framework is based on a clear separation of the mission and critical issues during requirements analysis. Analysis of the critical issues is performed in two phases. The first phase identifies those "real world" properties relevant to the critical requirements: the physical laws or rules of operation, and the system hazards. In the second phase, the interface between the system and its environment is identified, and the behaviour required at this interface is specified. For each phase, due to its own characteristics, we propose the utilization of different formal models, respectively, a logical formalism (timed history logic) and a net formalism (predicate-transition nets). To illustrate the proposed framework an example based on a train set crossing is presented.

© 1993 University of Newcastle upon Tyne.
Printed and published by the University of Newcastle upon Tyne,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.
Bibliographical details

SAEED, Amer

[By] A. Saeed, R. de Lemos and T. Anderson

(University of Newcastle upon Tyne, Computing Science, Technical Report Series, no.429)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE.
DE LEMOS, Rogerio Sergio Neves
ANDERSON, Thomas

Abstract

Requirements analysis plays a vital role in the development of safety-critical systems since any errors in the requirements specification will corrupt the subsequent stages of system development. Experience in safety-critical systems has shown that requirements specification errors can and do cause accidents. This paper presents a general framework for the formal specification and verification of the critical requirements in the development of safety-critical systems. The framework is based on a clear separation of the mission and critical issues during requirements analysis. Analysis of the critical issues is performed in two phases. The first phase identifies those "real world" properties relevant to the critical requirements: the physical laws or rules of operation, and the system hazards. In the second phase, the interface between the system and its environment is identified, and the behaviour required at this interface is specified. For each phase, due to its own characteristics, we propose the utilization of different formal models, respectively, a logical formalism (timed history logic) and a net formalism (predicate-transition nets). To illustrate the proposed framework an example based on a train set crossing is presented.

About the author

Dr. A. Saeed is a Research Associate in the Department of Computing Science.

Mr. R. de Lemos is a Research Associate in the Department of Computing Science.

Professor T. Anderson is a Professor in the Department of Computing Science.

Suggested keywords

FORMAL MODELS REQUIREMENTS ANALYSIS SAFETY-CRITICAL SYSTEMS
SPECIFICATION VERIFICATION

Suggested classmarks (primary classmark underlined)

Dewey (18th): 001.6425 658.47
U.D.C. 681.322.06 519.718
The Role of Formal Methods in the Requirements Analysis of Safety–Critical Systems:
a Train Set Example

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

Abstract

Requirements analysis plays a vital role in the development of safety–critical systems since any errors in the requirements specification will corrupt the subsequent stages of system development. Experience in safety–critical systems has shown that requirements specification errors can and do cause accidents. This paper presents a general framework for the formal specification and verification of the critical requirements in the development of safety–critical systems. The framework is based on a clear separation of the mission and critical issues during requirements analysis. Analysis of the critical issues is performed in two phases. The first phase identifies those “real world” properties relevant to the critical requirements: the physical laws or rules of operation, and the system hazards. In the second phase, the interface between the system and its environment is identified, and the behaviour required at this interface is specified. For each phase, due to its own characteristics, we propose the utilization of different formal models, respectively, a logical formalism (timed history logic) and a net formalism (predicate–transition nets). To illustrate the proposed framework an example based on a train set crossing is presented.

Keywords: safety–critical systems, specification, verification, formal models, requirements analysis.

1. Introduction

A safety–critical system is a system for which there exists at least one failure that can be adjudged to cause a catastrophe (e.g. loss of life). The development of such systems is usually controlled by regulatory authorities, depending on the area of application, such as transportation, aerospace, energy industry, medicine, and defence systems. These regulatory boards impose certification criteria, in accordance with the safety standards established for the different sectors, which must be satisfied before the system can be put into service. As the use of computers increases in such critical applications and the level of criticality of the roles performed by the computers also increases, new methods for the development of such systems are urgently needed. One approach to improving the level of safety is to use formal specification and verification in conjunction with other methods of software development, such as testing and software fault–tolerance. Some authors consider that the utilization of formal methods is the only way, at least, in principle, that high levels of safety can be obtained for safety–critical applications [13]. The potential advantages of using formal methods, apart
from contributing to improving the understanding of the whole specification, include
unambiguity, refinement consistency, and the opportunity to check for completeness (with
respect to a key set of questions and inferences based on the information specified [7]).
However, to realize these potential benefits, a systematic development methodology is
required – to allow the analysts to focus on the realities of the system during requirements
analysis. The provision of such a methodology, together with a formal model, introduces a
clear role for automated tools.

In this paper, the approach to be followed in a framework for the software requirements
analysis for safety-critical systems, is based on a clear separation of the mission and the
critical requirements, and moreover, the subdivision of the analysis of the critical
requirements into two distinct phases. On the one hand, the mission requirements focus on
what the system is supposed to achieve in terms of function, timeliness and some dependability
requirements – namely the attributes of reliability, availability and security [10]. On the other
hand, the critical requirements focus on the elimination and control of hazards, and the
limitation of damage in the case of an accident; in other words, they address what the system
should not do [11], and are thus related to the safety attribute of dependability. Some of the
benefits of making this distinction during requirements analysis are: the resolution of potential
conflicts, detection of omissions and inconsistencies between the mission and critical issues,
the simplification of safety certification, and the ability to focus on the safety-critical issues.

As far as the analysis of the critical requirements is concerned, we recognise two phases: the
first deals with the identification of the real world properties and the system hazards, and the
second deals with the mapping of the real world state into the system [2]. In other words, we are
concerned with the laws or rules which dictate the behaviour of the system, and how these laws
or rules are perceived and handled by the system. Because of the differing characteristics of
each phase, instead of seeking a single formalism this paper discusses the approach of using,
different formalism for each phase [16]. This has the advantage of allowing us to select
formalisms in accordance with the properties that should be expressed at each phase of
development. Furthermore, the use of more than one formalism can bring benefits of
increased flexibility, which can be exploited to accommodate the differences in the demands
and characteristics of different safety-critical application areas, such as different levels of risk
which in turn demand cost-related solutions [1].

The framework presented here for the formal specification and verification of critical
requirements, aims to show that different formalisms can be integrated, exploiting the most
appropriate features of each formalism. The rest of the paper is organized as follows. The next
section describes the main components (and their interrelationship) into which a process
control safety-critical system can be divided. Section 3 presents the framework for the formal
specification and verification of the critical requirements, giving the characteristics and the
formalism to be used in each phase. In section 4, an example based on a train set crossing is discussed, in accordance with the framework of the previous section. Finally, section 5 presents some concluding remarks.

2. General Structure of Safety–Critical Systems

It is believed that the adoption of a general structure for a safety–critical system will be a useful guide to the requirements analysis, since it allows the analysis to be split into different phases. For applications referred to as process control systems, which are the primary focus of this paper, a commonly accepted structure is to partition the system into three distinct components: the **operator**, the **controller**, and the **physical process** (or plant). The **environment** is that part of the rest of the world which may affect, or be affected by, the system. Applying the approach of separating the mission and critical issues, the general structure of the system can be further decomposed into: mission and safety operators, and mission and safety controllers. This structure is shown in figure 1.

![Diagram showing the general structure of a safety-critical system](image)

**Figure 1. General Structure for Safety–Critical System**

The safety controller and the safety operator are the components which must ensure that the system does not enter a hazardous state. The interaction between the mission and the safety components is restricted, by imposing the condition that the mission components cannot directly affect the behaviour exhibited by the safety components. This situation can be realized by preventing the mission controller from issuing any control commands to the safety controller; it is up to the safety controller to monitor the operating conditions of the mission
controller and then to issue the appropriate control commands to the mission controller. Thus, the mission controller can influence the safety controller only via the observations made by the safety controller (either over the state of the physical process or mission controller). The safety controller must assume sole control of the system, that is, it must override the mission controller, whenever the state of the system ceases to be safe. Also, the safety controller endeavours to prevent the occurrence of hazardous states, and, in the worst case, if an accident becomes inevitable, must minimize the amount of damage that will be caused to the system and the environment.

Still referring to figure 1, it would be preferable to have two totally independent controllers, with separate sensors and actuators, to avoid any common failures of both the mission and safety controller. Unfortunately, for some applications this ideal structure is difficult to obtain due to limitations found in the physical process, or restrictions imposed by the design of the system.

The physical process can be classified according to the dynamics of the system [6]: continuous variable, discrete event, or a combination of these two. Continuous variable dynamic systems (CVDS) are modelled in terms of transfer functions using differential equations, and can tolerate brief state inconsistencies between the actual state of the real world and that perceived by the safety controller; examples of such systems are chemical plant and airborne systems. In contrast, discrete event dynamic systems (DEDS) are modelled by automata or discrete mathematics, but do not tolerate momentary state inconsistencies; examples of such systems are manufacturing systems and high—voltage power substations. Since DEDS are invariably man—made systems, there are no physical laws which constrain system configuration other than natural limits of material and ergonomics — rules of operation. In this paper, we are only concerned with the requirements analysis of the safety controller in discrete event systems.

3. Requirements Analysis of Safety—Critical Systems

The aim of this paper is to restrict the requirements analysis of safety—critical software to the critical issues for the system. However, it is usually impossible to maintain a complete dichotomy between the mission and the critical requirements, because it would be futile to impose critical requirements that are so stringent that the system could not satisfy its mission. Some aspects of the mission requirements must be considered in the analysis of the critical requirements, to ensure that the behaviour of the safety controller will (under fault—free circumstances) permit the satisfaction of the mission requirements as well as ensuring the absence of system hazards. Furthermore, there will also be common real world properties, arising from the laws of the physical process, that influence both mission and critical requirements.
In the following, two phases of the analysis of the critical requirements are presented in terms of their main characteristics. These two phases are called the critical requirements analysis, and the critical system analysis. The separation of the analysis of the critical requirements into two distinct phases is intended to simplify the analysis, permitting an easier understanding and reasoning about the real world properties, and how the system perceives and manipulates them. In this approach, instead of using a single formalism for both phases, a different formalism is used in each phase. This enables formalisms to be selected according to criteria appropriate each phase of the requirements analysis.

Before these two formal analysis phases can be conducted a preliminary informal phase must be performed — conceptual analysis — during which a conceptual model of the system is defined by a multidisciplinary team. The objective of the conceptual analysis is to produce an initial, informal statement of the aim and purpose of the system and to identify any potential disasters; the latter is an essential first step in determining the critical requirements.

3.1. Critical Requirements Analysis

The activities to be performed during this phase include the identification of the real world properties, in terms of physical laws and rules of operation when dealing with CVDS and DEDS respectively, and the identification of the system hazards. As a product of the real world analysis, the critical requirements specification is produced, containing the safety constraint and the safety strategies. The safety constraint is simply the negation of the hazards, whereas the safety strategies are schemes which are used to reduce (or even to eliminate) the possibility of the occurrence of hazards. The general features which are required from a formalism to be used in this phase are the following [16]: the specifications should have a conjunctive character, in the sense that new requirements can be added to the specification without the need to reconstruct the full specification, the behaviour of the system should be defined in terms of all of its possible runs (i.e. a linear sequence of events and/or states), and there is no need to explicitly specify concurrency and non-determinism. The most appropriate formalisms for this phase are logical formalisms, such as Temporal Logic [5], Real-Time Logic (RTL) [8], Real-Time Temporal Logic (RTTL) [15], and Timed History Logic (THL) [17].

We include a summary of THL, which will be used for the critical requirements analysis phase in the train set crossing example. This model allows the representation of physical laws, parallelism and timing issues in a coherent and analysable form.

Timed History Logic

There are three main concepts in this formal model: histories which are the underlying semantics of the model; relations which are constructs used to capture properties that are invariant over histories; and modes which are used to structure specifications expressed by the
model. We now present an overview of histories and relations; a more detailed description of the model is given elsewhere [18].

System History

The base time set (BT) is the set of non-negative real numbers. A time point (t) is defined as an element of BT, and a time interval (Int) as a subset of BT which represents an uninterrupted period of time. Over any interval the following functions are defined: start point s(Int) — the latest time point in BT less than or equal to all time points in Int; end point e(Int) — the earliest time point in BT greater than or equal to all time points in Int; and interval set SI(Int) — the set of all intervals contained within Int. The system lifetime (T) is an interval which represents the operational lifetime of the system.

The state vector (Sv) of a system is simply the vector of all its state variables. These are the time varying quantities which either measure or represent factors that influence the mission or safety of the system. For a system with n state variable we have: Sv = (p₁, ..., pₙ) The set of possible values of a state variable (say, pᵢ) is defined by its variable range (Vpᵢ). The state space of a system (Γ) is defined as the cross product of the variable ranges.

A history H of a system is a function of the form H: T → Γ. The set of all “possible” histories of a system is defined as the universal history set (ΓH) which is the set of all history functions. For a history H the sequence of values taken by a state variable pᵢ are denoted by the function H.pᵢ: T → Vpᵢ.

Restrictions imposed on system behaviour are captured using three sorts of relations: class relations, invariant relations and history relations.

Class relations

Class relations are used to define classes of variables with similar mathematical properties. A class relation for a variable pᵢ is a predicate built using standard relations and logical connectives, and one free function variable pᵢ. A history H satisfies a class relation Cpᵢ if and only if the substitution of H.pᵢ for pᵢ within Cpᵢ evaluates to true. We denote this by writing: H sat Cpᵢ.

Invariant relations

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 system predicate is a predicate built using standard mathematical functions, relations and logical connectives, and n free value variables p₁, ..., pₙ of types Vp₁, ..., Vpₙ. No other free variables may be used.

A tuple of values V = (x₁, ..., xₙ), where xᵢ is of type Vpᵢ, satisfies a system predicate SysPred if and only if substitution of each xᵢ for pᵢ within SysPred evaluates to true. We denote this by writing: V sat SysPred.
A system predicate SysPred is an invariant relation for a history H if and only if: 
\( <H.p_1(t), \ldots, H.p_n(t)> \) sat SysPred for all \( t \in T \). This will be abbreviated as \( H \) sat SysPred.

**History relations**

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 standard mathematical functions, relations and logical connectives, two free time variables \( T_0, T_1 \), 2n free value variables \( p_{1,0}, \ldots, p_{n,0}, p_{1,1}, \ldots, p_{n,1} \) (where \( p_{i,j} \) has type \( V_{p_i} \)), and n free function variables \( p_1, \ldots, p_n \) (where \( p_i \) is a function of class \( C_{p_i} \)). No other free variables may be used.

A history \( H \) satisfies a history predicate HistPred 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 \( T_1 \), iii) \( H.p_i(s(\text{Int})) \) for \( p_{i,0} \) for all \( i \), iv) \( H.p_i(e(\text{Int})) \) for \( p_{i,1} \) for all \( i \), and v) \( H.p_i \) for \( p_i \) for all \( i \), evaluates to true. We will denote this by writing: \( H \) sat HistPred@\( \text{Int} \).

A history predicate HistPred is a history relation for a history \( H \) if and only if \( H \) sat HistPred@\( \text{Int} \) for all \( \text{Int} \in \text{SI}(T) \). This will be abbreviated as \( H \) sat HistPred.

**3.2. Critical System Analysis**

The activities to be performed during this phase include the identification of the interface between the safety controller and the physical process, and the specification of system behaviour that must be observed at the identified interface. Also in this phase a top level organization of the system is realized in terms of the properties of the sensors and actuators of the system, and the effects of the possible failures of these sensors and actuators. This phase leads to the production of the critical system specification, which contains a description of the relationship between the actuators and sensors of the safety controller and the real world, and a description of the interrelationship between the components of the safety controller. Our aim is to construct a critical system specification that is robust to failures in the mission subsystem; that is, the behaviour expressed by the critical system specification should be unaffected by failures in the mission subsystem. The general features which are required from a formalism to be used in this phase are the following: the specification should be able to represent the architectural design of the system, modelling the interactions between the critical requirements, which implies the need to explicitly specify concurrency and non-determinism. Thus, the type of formalisms most appropriate for this phase are graphical formalisms [12] such as Petri nets, Extended State Machines (ESM) [15], and Modecharts [9]. Also, program-like formalisms that have constructs supporting non-determinism and concurrency can be employed, such as CSP and CCS.

In the following we present the semantics of Predicate—Transition nets (PrT nets) [4], a form of high-level Petri net, which will be used in the critical system analysis phase of the train set.
crossing example. Petri nets are mainly used for the modelling and analysis of discrete-event systems which are concurrent, asynchronous, and non-deterministic. The use of PrT nets, instead of (Timed) Petri nets [12], adds to the modelling power of the latter the formal treatment of individuals (i.e. the notion of token identity) and their changing properties and relations.

**Predicate—Transition Nets (PrT Nets)**

Below, we present an informal definition of PrT nets; a formal definition is given elsewhere /Genrich 81, Murata 88/.

**Definition.** Let S, T, F be finite sets. The triple N = (S, T, F) is called a directed net iff the following conditions hold:

1. \( S \cap T = \emptyset; \)
2. \( S \cup T \neq \emptyset; \)
3. \( F \subseteq (S \times T) \cup (T \times S); \)
4. domain (F) union codomain (F) = S \cup T,

For a given net N = (S, T, F) S is the set of places of N, T is the set of transitions of N, and F is the flow relation containing the arcs of N. For \( x \in S \cup T \), \( I(x) = \{ y \in S \cup T | (y, x) \in F \} \) is called the preset of \( x \), and \( O(x) = \{ y \in S \cup T | (x, y) \in F \} \) is called the postset of \( x \).

**Definition.** A PrT net consists of the following constituents:

1. a directed net (S, T, F) where,

   S is the set of predicates ("first-order" places), and

   T is the set of transitions (schemes of marking changes);

2. predicates are relations among individuals; a predicate is applied to a specific number of arguments and has the value of true or false when individuals are supplied as arguments;

3. the transitions are schemes of elementary changes of markings representing the processes carried out by the system — instances of these schemes are generated by means of consistent substitution of individual variables by symbols;

4. an arc label specifies a variable extension of a predicate to which the arc is connected. The zero-tuple indicating a no-argument predicate (i.e. an ordinary place) is denoted by the special symbol '.

**Definition.** The arity of a predicate is defined by the arity of its incident arc labels; the arity of a transition is defined by the cardinality of the set of variables occurring in the arc labels of its incoming arcs.
Definition. The graphical representation of a PrT net is obtained by representing a predicate by a circle, a transition by a box, an element of $F \cap (S \times T)$ by a directed arc from a circle to a box, and an element of $F \cap (T \times S)$ by a directed arc from a box to a circle.

3.3. The Overall Framework and Comparison with Other Approaches

In the following we explain how the specification of the critical requirements should be produced. Also, we compare our proposed approach with other approaches currently found in the literature.

The basic aim behind our framework for the requirements analysis of safety—critical systems is to subdivide the whole problem into smaller domains where the analysis of the requirements can be simplified, thereby leading to more accurate specifications. This is achieved by, first, splitting mission and critical requirements, and second, subdividing the analysis of the critical requirements into the critical requirements analysis and critical system analysis. In the analysis of a system, performing the critical system analysis may highlight some shortcomings in the safety strategy defined during the critical requirements analysis — these shortcomings would have to be addressed by modifying the safety strategy (the safety constraint cannot be modified without changing the physical process).

In this paper we have identified just two distinct phases for critical requirements analysis, but a larger number of phases could be utilised if this was desirable, depending on the type of application and its level of criticality. The choice of the type of formalism to be used, at each phase, is influenced by the issues which are most important at that phase, and the potential that certain formalisms have in representing these issues. This approach enables a fuller exploitation of the appropriate features of the different formalisms than can be achieved with a single—formalism approach.

If we adopt the approach of performing the critical requirements analysis in two phases, and representing the specifications produced at each phase in a different formalism, we must provide a means to relate the different specifications. After specifying and verifying the laws and rules which describe the real world properties, these become invariants for the second phase, against which the refinement of the system model should be verified [3]. One advantage of using net formalisms as a modelling notation is that they provide the opportunity for using simulations as a means of validating the critical system specification. After obtaining the reachability graph from the PrT net model, the same analysis in modelling faults and failures as that presented in [11] can be performed.

The basic approach suggested in this paper, of dividing the requirements analysis into distinct phases, and performing the analysis at each phase in the most appropriate formalism, has not previously been investigated. What has usually been presented is the utilization of a single formalism such as Invariants [2], Temporal Logic [5], Petri nets [12], and THL [18]. However, there are two approaches in the literature which use different formalisms for requirements
These papers are primarily concerned with the analysis of timeliness requirements; it is not their concern to establish a methodology for the analysis of the critical requirements of safety—critical systems — the issue which is central to this paper. In [15], ESMs are used to model “plant—controller processes”; from the paths of these ESMs, trajectories are obtained which can be used to provide a formal operational semantics. The specification of plant behaviour is then given by RTTL formulae over these trajectories, and verified by demonstrating that the trajectories defined by the ESMs do indeed comply with RTTL formulae. In [9], the specification of the system is realized in terms of RTL and Modecharts; Modecharts produce a decision procedure for classes of properties expressed as RTL formulas. The system properties are verified using Computation Graphs, obtained from the Modecharts, to check if the corresponding RTL formulas comply with the Modecharts.

4. The Train Set Crossing

With the aim of exemplifying the proposed framework, an example of a train set crossing was selected. An obvious advantage of using a train set instead of a real railway system is that safety strategies can be studied and implemented without endangering the travelling public. The train set crossing described below raises safety—critical issues that are similar to those found at the traditional level crossing (i.e. road—rail). Although full proofs have been constructed for the lemmas presented in this section, only sketches are provided here due to space limitations.

Conceptual Analysis

The physical process consists of two track circuits Cp and Cs, and two types of trains — primary (Tp) and secondary (Ts). The circuits are divided into sections and there are two separate crossing sections at which they intersect. Trains of type Tp travel around circuit Cp and trains of type Ts travel around circuit Cs. The longest train is shorter than the smallest section. The primary trains always take priority over the secondary trains at the crossing sections (c.f. priority of trains over road vehicles at a level crossing). Specifically, a primary train must not be made to wait for a secondary train at a crossing section. Several disasters are associated with the system, but we only consider the following potential disasters: a. trains of the same type collide; and b. trains of different type collide.

4.1. Critical Requirements Analysis

The critical requirements analysis is concerned with the identification of the system hazards, the formalization of the safety constraint and the definition of a safety strategy. We consider each potential disaster separately.

Remark. During the critical requirements analysis addition and subtraction on circuit section numbers are performed modulo the number of sections of the circuit.
4.1.1. Collision of Trains of Same Type

We construct a general model for a set of trains running on a circuit C. This model consists of two sets: the sections of C, \( S = \{0, \ldots, N_s\} \) and the trains which run on C, \( \text{Tr} = \{1, \ldots, N_t\} \); and a state variable \( \text{Ptrain} \) which represents the position of the trains on C.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Range</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>Ptrain</td>
<td>( S^{N_t} )</td>
<td>The position of each train expressed as a section number, that is the section containing the front of a train.</td>
</tr>
</tbody>
</table>

We will use \( \text{Ptrain}(i) \) to denote the state variable for the position of train \( i \), in the formulation of system and history predicates.

The hazardous states of the circuit are identified by considering the relative position of the trains on that circuit. Firstly, we make the observation that a collision can occur only if some part of two trains are in the same section. If we consider the case when the front of a train is in section \( i \), since trains are less than one section in length we can have a collision involving that train only if the front of another train is in section \( i+1 \), section \( i \) or section \( i-1 \). The hazardous states of the circuit are those states in which a train may be involved in a collision, therefore a state is hazardous if the front of one train is in the same, or adjacent, section as the front of another train. Hazardous states can be expressed by the system predicate: \( \exists x, y \in \text{Tr}: x \neq y \land \text{Ptrain}(x) \in \{\text{Ptrain}(y)-1, \text{Ptrain}(y), \text{Ptrain}(y)+1\} \).

Thus we deduce that a safety constraint of the form “for any two trains there must be at least one section between the sections containing the fronts of the trains” will, if maintained, prevent the occurrence of any hazardous state. This safety constraint (SC) is expressed as a system predicate: \( \forall x, y \in \text{Tr}: x \neq y \Rightarrow \text{Ptrain}(x) \notin \{\text{Ptrain}(y)-1, \text{Ptrain}(y), \text{Ptrain}(y)+1\} \). The universal history set \( \Gamma H \) is the set of all functions \( H: T \rightarrow S^{N_t} \); the set of safe histories \( \Gamma H \) is the subset of \( \Gamma H \) for which SC is an invariant relation: \( \Gamma H = \{H \in \Gamma H \mid H \text{ sat SC}\} \).

Circuit Safety Strategy

The circuit safety strategy is based on a reservation scheme; the basic rules of this scheme are:

- a. for any train the current section (i.e. the position of the front of the train) and the section behind the current section must always be reserved; and b. no section can be reserved by more than one train.

To formalize the rules of the reservation scheme we extend our model by the introduction of the notion of a reservation set \( R = \mathcal{P}(S) \), the set of subsets of sections, and a state variable \( \text{Rtrain} \) which captures the reservation sets of the trains.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Range</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>p2</td>
<td>Rtrain</td>
<td>( R^{N_t} )</td>
<td>The reservation sets of the trains.</td>
</tr>
</tbody>
</table>

The rules can then be formally expressed as two system predicates:

- a. \( \forall x \in \text{Tr}: \{\text{Ptrain}(x)-1, \text{Ptrain}(x)\} \subseteq \text{Rtrain}(x) \); and
b. \( \forall x, y \in \text{Tr}: x \neq y \Rightarrow \text{Rtrain}(x) \cap \text{Rtrain}(y) = \emptyset. \)

We say that a history \( H : T \rightarrow S^{Nt} \times R^{Nt} \) satisfies this safety strategy if and only if predicates \( a \) and \( b \) are invariant relations for that history.

**Lemma 4.1.**

*A history that satisfies the circuit safety strategy must be a safe history.*

**Proof:** (By contradiction.) Assume \( \exists x, y \in \text{Tr}: x \neq y \land \text{Ptrain}(x) \in \{ \text{Ptrain}(y) - 1, \text{Ptrain}(y), \text{Ptrain}(y) + 1 \}. \) Then by rule \( a \), \( \{ \text{Ptrain}(x) - 1, \text{Ptrain}(x) \} \subseteq \text{Rtrain}(x) \land \{ \text{Ptrain}(y) - 1, \text{Ptrain}(y) \} \subseteq \text{Rtrain}(y) \), hence \( \text{Rtrain}(x) \cap \text{Rtrain}(y) \neq \emptyset. \) This contradicts rule \( b \).

### 4.1.1. Collision of Trains of Different Type

We construct a model for two types of trains, running on two circuits. This model consists of: four sets: sections of \( C_p, S_p = \{ 0, ..., Nsp \} \); primary trains, \( T_p = \{ 1, ..., Ntp \} \); sections of \( C_s, S_s = \{ 0, ..., Nss \} \) and secondary trains, \( T_s = \{ 1, ..., Nts \} \); and two state variables \( \text{Ptrainp} \) and \( \text{Ptrains} \) which represent the position of trains on \( C_p \) and on \( C_s \) respectively.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Range</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( p_1 )</td>
<td>( \text{Ptrainp} )</td>
<td>( S_{p_{Np}}^{Np} ) The position of each primary train expressed as a section number.</td>
</tr>
<tr>
<td>2</td>
<td>( p_2 )</td>
<td>( \text{Ptrains} )</td>
<td>( S_{s_{Nts}}^{Nss} ) The position of each secondary train expressed as a section number.</td>
</tr>
</tbody>
</table>

The hazardous states for the collision of trains of different type are identified by considering the positions of the primary and secondary trains relative to the crossing sections. Informally, a crossing section is that part of the track which consists of the sections (one from each circuit) at which the two circuits intersect. To simplify the analysis we consider only one crossing section (CC). We can have a collision involving a primary train and a secondary train only when both trains are in CC. Since the length of a train is less than a section, a train is in CC only if the front of that train is in CC or the section that follows CC. For each circuit we introduce the notion of a danger zone as the set of sections in which the front of a train can be while that train is in CC. Hence, the danger zone on \( C_p \) is defined as: \( D_p = \{ C{C_p}, C{C_p} + 1 \} \), where \( C{C_p} \) is the number of the section of \( C_p \) that is part of CC; and the danger zone on \( C_s \) as: \( D_s = \{ C{C_s}, C{C_s} + 1 \} \), where \( C{C_s} \) is the number of the section of \( C_s \) that is part of CC. We conclude that the hazardous states for the collision of trains of different type are those in which the front of a primary train is in the danger zone \( D_p \) and the front of a secondary train is in the danger zone \( D_s \). These hazardous states can be expressed by the system predicate: \( (\exists x \in T_p: \text{Ptrainp}(x) \in D_p) \land (\exists x \in T_s: \text{Ptrains}(x) \in D_s) \).

Thus we deduce that a safety constraint of the form "either the front of no primary train is in the danger zone \( D_p \) or the front of no secondary train is in the danger zone \( D_s \)" will, if maintained, prevent the occurrence of any hazardous state. This safety constraint (SC) is expressed as a system predicate: \( (\forall x \in T_p: \text{Ptrainp}(x) \notin D_p) \lor (\forall x \in T_s: \text{Ptrains}(x) \notin D_s) \). As
before, we define the safe history set $SH$ as: $SH = \{H \in \Gamma H| H \text{ sat } SC\}$, where $\Gamma H$ is the set of all functions $H: T \rightarrow S_{SpNt} \times S_{S_{Nt}}$.

**Crossing Safety Strategy**

The crossing safety strategy is based on a modification of the reservation scheme. The two basic rules are: a. for any train the current section (i.e. the position of the train) and the section behind the current section must always be reserved; and b. section $CCp$ and section $CCs$ cannot both be reserved.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Range</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_3$</td>
<td>$Rtrainp$</td>
<td>$R_{SpNt}$</td>
<td>The reservation sets of the primary trains.</td>
</tr>
<tr>
<td>$p_4$</td>
<td>$Rtrains$</td>
<td>$R_{S_{Nt}}$</td>
<td>The reservation sets of the secondary trains.</td>
</tr>
</tbody>
</table>

The rules are formalized as two system predicates:

$a. (\forall x \in Tp: \{Ptrainp(x) - 1, Ptrainp(x)\} \subseteq Rtrainp(x)) \land (\forall x \in Ts: \{Ptrains(x) - 1, Ptrains(x)\} \subseteq Rtrains(x))$; and

$b. (\forall x \in Tp: CCp \notin Rtrainp(x)) \lor (\forall x \in Ts: CCs \notin Rtrains(x))$.

We will say that a history $H: T \rightarrow S_{SpNt} \times S_{S_{Nt}} \times R_{SpNt} \times R_{S_{Nt}}$ satisfies this safety strategy if and only if predicates $a$ and $b$ are invariant relations for that history.

**Lemma 4.2.**

A history that satisfies the crossing safety strategy must be a safe history.

**Proof:** (By contradiction.) Assume $(\exists x \in Tp: Ptrainp(x) \in Dp) \land (\exists y \in Ts: Ptrains(y) \in Ds)$. Then by rule $a$ and the definitions of $Dp$ and $Ds$, $CCp \in Rtrainp(x) \land CCs \in Rtrains(y)$. This contradicts rule $b$.

**Mission Requirement and Safety Strategy**

Although the safety strategy for the crossing section is sufficient to prevent collisions involving primary and secondary trains, it ignores the mission requirement concerning the priority of primary trains over secondary trains. More specifically, the safety strategy allows the safety controller to reserve the crossing section for a secondary train without regard to the priority that should be given to the primary trains. To resolve the conflict between the safety strategy and the priority of the primary trains, the safety strategy is modified to allow a secondary train to reserve the crossing section only when it is anticipated that it will pass through the crossing section before a primary train arrives at the crossing section. To model the priority of primary trains over secondary trains we introduce the notion of a priority zone, on circuit $Cp$ defined as: $PZ = \{CCp - ZL, ..., CCp - 1, CCp\}$. The value of $ZL (\geq 1)$ in the definition of $PZ$ depends on the anticipated relative velocities of the primary and secondary trains. A safety strategy for the crossing section which incorporates the priority of primary trains over secondary trains is defined by adding the following rule (the priority constraint) to the crossing section safety strategy: for any interval $Int$, if a secondary train $i$ has not reserved the section $CCs$ at the start.
point of \textit{Int}, and at every time point during \textit{Int} some section in the priority zone is reserved, then the secondary train \textit{i} must not reserve CCs at any time point in \textit{Int}. This rule is formalized as a history predicate:
\[ \forall x \in Ts: [ \text{CCs} \notin \text{Rtrains}(x)(T_0) \land \forall t \in [T_0, T_1]: \exists y \in \text{Tp}: \text{PZ} \land \text{Rtrainp}(y)(t) \neq 0 \rightarrow \forall t \in [T_0, T_1]: \text{CCs} \notin \text{Rtrains}(x)(t) ]. \]

We will say a history satisfies the priority constraint if the history predicate above is satisfied for any interval within the system lifetime (i.e. it is a history relation for that history).

The set of histories that satisfy the conjunction of the priority constraint and crossing safety strategy, form a subset of those that satisfy the crossing safety strategy and hence they are all safe histories.

This analysis can be applied to both crossing sections, provided the crossing sections are sufficiently far apart. That is, on the circuit \textit{Cp} they should be at least \textit{ZL}+1 sections apart (which ensures that the priority zones do not overlap) and on the circuit \textit{Cs} at least 2 sections apart (which ensures that the danger zones do not overlap).

\textbf{4.2. Critical System Analysis}

After establishing the safety strategies during the critical requirements analysis phase, the critical system analysis phase investigates how these strategies are to be implemented by the safety controller. In other words, we must investigate how the safety strategies can be mapped onto a set of sensors and actuators, and evaluate the impact of failures in these sensors and actuators on the validity of the safety strategies. In the train set, at the start of every section there is a sensor which detects the presence of a train, and an actuator which allows the safety controller to stop the train within a section.

The analysis to be performed entails modelling the relationship between the components of the safety controller (including sensors and actuators), the physical process, and the interface between them. The general approach followed in modelling the system is to maintain a clear separation between models of the physical process and the safety controller, even though they must cooperate whenever an action is to be performed. The advantage of adopting this approach is that both models can be independently developed and modified, and the state of the physical process can be seen to correspond to the sequence of control commands issued by the safety controller.

In our PrT net model of the train set circuit, we have chosen to represent explicitly each of the individual actions that make up the track circuit because this gives a clear visualization of how the physical process behaves under the safety controller. The analysis will be divided into two parts according to the safety strategies, established during the critical requirement analysis. First, we model the safety strategy which prevents the collision of trains of the same type, and second, we model the safety strategy which prevents the collision of trains of different types;
the latter includes the provision of priority of primary over secondary trains. The failure analysis of sensors and actuators will only be presented for the first safety strategy.

4.2.1. Collision of Trains of the Same Type

To illustrate modelling the behaviour of trains in a circuit, we assume that each circuit contains seven sections (N\textsubscript{s}=6) and two trains (N\textsubscript{t}=2). For simplicity, the modelling and analysis is performed for just one circuit. The model is shown in figure 2. The sections in the circuit are denoted by \(i, j \in S, S = \{0, ..., 6\}\), and the trains are denoted by \(x \in \text{Tr}, \text{Tr} = \{1, 2\}\). The predicates of the PrT net model of a train set circuit are the following:

\[
\begin{align*}
S_j(x) & \quad \text{train } x \text{ occupies this section (section } j \text{ of the circuit)}; \\
\text{SPP}(i,x) & \quad \text{status of the circuit as perceived by the controller, in terms of sections } i \text{ that are occupied by trains } x; \\
\text{RS}(i) & \quad \text{section } i \text{ will be released (i.e. not reserved);} \\
\text{FS}(j) & \quad \text{section } j \text{ is not reserved;} \\
\text{IPC}(j,x) & \quad \text{train } x \text{ has entered section } j; \\
\text{ICP}(j,x) & \quad \text{train } x \text{ is allowed to enter section } j.
\end{align*}
\]

Starting from the initial marking, as shown in figure 2, whenever a train enters a new section, the safety controller is notified by a tuple \((j,x)\) containing the section and the train number, which represents a sensor detecting the passage of a train. This information enables the safety controller to find out which section should be released (transition t2), and which trains are then allowed to move into new sections (transition t3). The latter transition represents the activity of the safety controller instructing the actuator to issue a control command to allow a train to enter a new section.

Sensor and Actuator Failures

In the following we consider what are the consequences if either a sensor or an actuator fails. Sensors can fail either by not detecting a train when it enters a section, or by detecting a "ghost train" – which occurs when a sensor detects a train in a section which is actually empty. Actuators fail by not stopping a train within a section. In this paper we only examine the first type of sensor failure.

For the analysis of sensor failures in the PrT net model we introduce a "sink transition" (its firing only consumes tokens without producing any tokens) from the predicate \(\text{IPC}(j,x)\) with the following relation \((j=1 \land x=1)\). This represents a typical case in which, although train \(1\) has moved from section \(0\) to section \(1\), the information may not be received by the safety controller. Through simulation of the PrT net, by "token playing", we can then reach a deadlock state where train \(1\) cannot move to the next section because the safety controller has not received the tuple corresponding to the passage of train \(1\) to section \(1\). This shows that
Figure 2. The PrT net Model of the Train Set Circuit

the safety constraint is satisfied even when a sensor fails to detect a train entering a section.

4.2.2. Collision of Trains of Different Type

To illustrate the modelling of trains in the crossing section of two circuits, we assume that: the two circuits cross only once, each circuit contains seven sections, and two trains run on each
circuit. To simplify the analysis we ignore the safety strategy for avoiding collisions of trains of the same type; thus we implicitly assume that each circuit has a safety controller applying the safety strategy developed in the previous subsection. The PrT net model of two circuits with a single crossing section is shown in figure 3. The circuits are denoted by \( m \in C, C = \{p, s\} \), sections of each circuit are denoted by \( j \in S, S = \{0, \ldots, 6\} \) (\( j = 6 \) is the crossing section CC on both circuits), and the trains are denoted by \( x \in Tr, Tr = \{1, 2\} \). The predicates of this model are the following:

- \( Smj(x) \) — train \( x \) occupies this section (section \( j \) of circuit \( m \));
- \( CC(x) \) — train \( x \) is in the crossing section;
- \( ICP(m,j) \) — danger zone of circuit \( m \) is reserved by a train;
- \( IPC(m,j) \) — determined by circuit \( m \) and section \( j \), either the danger zone in the circuit is released or a primary train enters the priority zone;
- \( EPT(\cdot) \) — a primary train is in the priority zone;
- \( SE(\cdot) \) — a secondary train is allowed to enter the danger zone.

The sections of the danger zone in circuit \( Cp \) are CC and Sp0, and in circuit \( Cs \) are CC and Ss0. The sections of the priority zone in circuit \( Cp \) are the danger zone and sections Sp4 and Sp5. Starting from the initial marking, as shown in figure 3, a secondary train will only be allowed to enter the danger zone if no primary train is in the priority zone. Whenever a secondary train leaves the danger zone, two tokens are produced (transition \( t3 \) which enables a primary or a secondary train to enter their respective danger zones. The two tokens in the predicate \( SE(\cdot) \) ensure that the two primary trains are allowed to be in the priority zone at the same time. Also notice that, within a certain firing sequence, a primary train may have to wait for a secondary train to vacate the danger zone of circuit \( Cs \). This occurs because it is assumed that each transition fires instantaneously, which does not accurately represent the time that the real world activities take to be performed. A more accurate model would be obtained by associating relative times with some of the transitions, representing the duration that an activity needs to be executed either in the physical process or by the safety controller.

5. Discussion and Conclusion

The analyses presented in section 4.1 concentrated on the logical properties of the train set crossing; as such, they could be performed using qualitative temporal logic [5]. However, in other type of applications, it may be necessary to introduce the dimension of time into the formal analysis. By using THL, as proposed in this paper, timing issues can be analysed by simply modifying the relations over the state variables. Work is being presently performed on the refinement of the PrT net models of the train set circuit and crossing to include the representation of time.

The difficulty of completely separating the mission from the critical requirements was emphasised in section 4.1.2. Conflicts must be resolved by finding safety strategies which are
adequate for both requirements, as in the case of the priority zone established for the primary trains.

The basic aim of this paper is to present a general framework for the specification and verification of critical requirements in the software development of safety-critical systems. The framework consists of the following steps. From the system conception, and after the
identification of the system disasters, the first phase is the enumeration of all possible hazards, and the description of the real world properties. These issues, together with the safety constraint and the safety strategies, are specified in terms of a logic formalism (which can be verified) to produce the critical requirements specification. A net model of the system under consideration is then built from the critical requirements specification; this expresses the interrelationships between the components of the safety controller. The general approach was shown to be feasible by applying it to the example of the train set crossing.

Acknowledgements

The authors would like to acknowledge the financial support of the Alvey Directorate/UK, BAe (DCSC), CAPES/Brazil, and the ESPRIT Basic Research Action PDCS.

References


