# ESTABLISH A GENERIC RAILWAY ELECTRONIC INTERLOCKING SOLUTION USING SOFTWARE ENGINEERING METHODS

A Dissertation Submitted to

**School of Electrical & Information Engineering**

**Engineering & the Built Environment**

**University of the Witwatersrand**

In Fulfilment

of the Requirements for the Degree of

**MSc (Software Engineering)**

**University of the Witwatersrand**

By

**Kirti Nathoo**

**0605347p**

**August 2014**

Supervisor: Professor Rex Van Olst

## DECLARATION

I, Kirti Nathoo, hereby declare that this dissertation entitled "***Establish a generic railway interlocking solution using software engineering methods***" represents my own work and has been written by me in its entirety. This dissertation has not already been submitted for any other degree or to any other university or tertiary institution for examination. To the best of my knowledge and belief, this paper contains no material previously published or written by another person, except where due reference has been made in the text.

.

_____ _____

K. Nathoo Signed at Lenasia South

**ACKNOWLEDGEMENTS**

I would like to thank my supervisor Professor Rex Van Olst for the support, great interest and guidance in helping me complete my thesis. Thank you to the Rail Automation division at Siemens AG, for the help and use of their resources and for the knowledge that I've gained through experience. Final thank you to everyone for their support in helping me accomplish this goal.

## ABSTRACT

A research investigation has been undertaken to establish a generic software interlocking solution for electronic railway systems. The system is intended to be independent of the physical station layout and easily adaptable in any country of application. Railway signalling principles and regulated safety standards are incorporated into the system design. A literature review has been performed to investigate existing interlocking methods and to identify common aspects amongst these methods. Existing methods for the development of electronic interlocking systems are evaluated. The application of software engineering techniques to interlocking systems is also considered. Thereafter a model of the generic solution is provided. The solution is designed following an agile life cycle development process. The structure of the interlocking is based on an MVC (Model-View-Controller) architecture which provides a modular foundation upon which the system is developed. The interlocking system is modelled using Boolean interlocking functions and UML (Unified Modelling Language) statecharts. Statecharts are used to graphically represent the procedures of interlocking operations. The Boolean interlocking functions and statechart models collectively represent a proof of concept for a generic interlocking software solution. The theoretical system models are used to simulate the interlocking software in TIA (Totally Integrated Automation) Portal. The behaviour of the interlocking during element faults and safety–critical events is validated through graphical software simulations. Test cases are derived based on software engineering test techniques to validate the behaviour and completeness of the software.

The software simulations indicate that the general algorithms defined for the system model can easily be determined for a specific station layout. The model is not dependent on the physical signalling elements. The generic algorithms defined for determining the availability of the signalling element types and the standard interlocking functions are easily adaptable to a physical layout. The generic solution encompasses interlocking principles and rail safety standards which enables the interlocking to respond in a fail-safe manner during hazardous events. The incorporation of formal software engineering methods assists in guaranteeing the safety of the system as safety components are built into the system at various stages. The use of development life cycle models and design patterns supports the development of a modular and flexible system architecture. This allows new additions or amendments to easily be incorporated into the system. The application of software engineering techniques assists in developing a generic maintainable interlocking solution for railways.

# Table of Contents

# List of Figures

# List of Tables

## ACRONYMS

APN – Automation Petri Net

CENELEC – European Committee for Electrotechnical Standardization

CIS – Computer Interlocking System

ECC – Element Control Computer

FSM – Finite State Machine

GUI – Graphical User Interface

HMI – Human Machine Interface

LLD – Ladder Logic Diagram

MPM – Multiprocessor Module

MVC – Model-View-Controller

PLC – Programmable Logic Controller

POSMOR – Principles of Safe Movement on Rails

RAD – Rapid Application Development

RAMS – Reliability, Availability, Maintainability and Safety

RTO – Radio Train Order

SCADA – Supervisory Control and Data Acquisition

SICAS – Siemens Computer Aided Signalling

SIL – Safety Integrity Level

SPAD – Signal Passed At Danger

SSI – Solid State Interlocking

TCC – Traffic Control Centre

THR – Tolerable Hazard Rates

TIA – Totally Integrated Automation

UML – Unified Modelling Language

## TERMS & DEFINITIONS

**Agile method** – A software life cycle process for the development of a software based system. Agile methods follow an iterative process where each iteration consists of the complete development process of requirements engineering, design, implementation and testing. Upon completion of an iteration a functional release of software is produced.

**Authority** – Permission awarded to a licensed person to perform a specified task.

**Berth Track** – Piece of track section facing a signal.

**Boolean -** A data type which consists of either one or two values, i.e. True of False.

**Calling a route** – Setting the start signal of a route to a green indication to allow a train to proceed along the route.

**CENELEC** – European Committee for Electrotechnical Standardization. Defines safety standards that must be followed to ensure safe and reliable systems are developed. These safety standards are applicable to both software and hardware systems. System models must be evaluated to determine the required SIL to be achieved.

**Conflicting Movement –** A movement of trains that may result in an accident or collision.

**Control table – A c**ontrol table consists of a set of rules of operation which must be obeyed for the safe movement of trains in a rail yard**.** The conditions defined in the control table are derived from the station layout. Conditions include route number, required field elements per route, position of elements, etc. The conditions in the control table are used as input for the interlocking software and can also be used to validate the developed software.

**Description Language** – The use of vivid and descriptive terms to describe an event or process.

**Design Pattern –** A design pattern serves as a software template upon which the system is

developed. The pattern highlights main aspects of the system architecture and creates a reusable framework which can be applied to other systems. The general design architecture consists of functional requirements, a processing system, user interface and data communication links between objects.

**Electronic Interlocking –** Electronic interlocking systems are controlled by microprocessors which monitor and control field equipment in a yard. Fail-safe logic and safety standards are built into the interlocking to ensure the safe movement of trains. In an electronic interlocking system the field elements are monitored electronically and feedback is graphically provided to train operators.

**Fail-Safe System –** During a fault or hazardous event, a fail-safe system is either forced into a predefined state or continues to function normally. In the event a system component fails, it must fail in a silent manner and not affect the other element or system functions.

**Field Elements** – Consist of physical equipment which forms part of a railway. Elements include signals, track sections, points, etc. Signals guide train control operators as trains are driven over track sections and point sets in a yard. The interlocking controls and monitors the field equipment.

**Fixed Block** – Category of train movement wherein the physical layout is divided into fixed blocks and components. The fixed blocks are separated by signals. Trains move from one fixed block to the next along a route.

**HMI Panel** – A graphical user interface through which users can interact with and operate an automated system.

**Least Restrictive Aspect –** The least restrictive aspect is the proceed aspect which does not indicate many restrictions in train movement. Therefore the train may continue along its designated route when a proceed aspect is observed. When a signal changes from a stop (i.e. red) aspect to a less restrictive aspect (i.e. yellow or green), the signal is said to have cleared.

**Line –** A railway track along which trains travel.

**Main Route** – Route upon which trains perform principle tasks of moving passengers or freight along main lines.

**Most Restrictive Aspect** – The stop aspect of a signal is defined as the most restrictive aspect. This classification is defined in terms of train movement such that the most restrictive aspect does not permit any movements of trains.

**Moving Block** – Category of train movement which consists of the area occupied by a train and the distance in front of it. No other train is allowed to enter this block as it travels along a route.

**MVC –** A software design pattern which decomposes a system into three distinct layers, i.e. model, view and controller. The model layer stores the data required by the system and perform logic functions. The view layers graphically displays the system data to the user and the control layer processes and responds to system data events.

**Overlap –** Section of track circuit beyond the destination signal which provides an additional piece of track in the event a train does not stop before the destination signal. This is a safety measure to prevent train collisions.

**Point** – A type of track circuit which connects/merges two railway lines into one. A points set allows a train to select between two routes of travel.

**Railway –** Railways are an affordable means of transportation commonly used for the transfer of passengers and freight over long distance. A railway system consists of signalling field equipment and an electronic interlocking system. These components function collectively to ensure the safe movement of trains on a railway.

**Railway Signalling –** Signalling system used to control train traffic and ensure the safe movement of trains in a rail yard. The system ensures no collisions occur between trains and signal elements display varying aspects which inform train drivers of conditions ahead.

**Route –** A route consists of start and destination signals between sequentially connected track

circuits and possible point sets. A train may only be permitted to travel upon an available route. The conditions required for a route are defined in a control table.

**Safety-critical System –** Is a system whose failure may cause injury or loss of human life, loss or severe damage to equipment or the environment.

**Shunt Route –** Shunt movements performed by trains are considered secondary movements that are not performed on main lines. These movements include moving wagons into a goods yard, moving empty carriages into a station, backing a freight train into a siding, etc.

**Signals** – Signals are placed between track sections and display different aspect indications. These indications are used to inform train operators of conditions ahead on the railway line.

**SIL** – A SIL is determined based on the level of criticality of a system. The SIL level also represents the probability of a system to execute specified safety functions within a defined time interval.

**SIMATIC –** Siemens integrated automation system which consists of a wide range of products for most automation needs.

**Software Engineering –** Is the application of engineering to the development of software. Software engineering covers the technical aspects of developing software systems through designing, implementing, testing and modifying software. It also covers software management concerns such as scheduling, budgeting, etc.

**Software Testing** – Method of assessing the quality and functionality of a system. The software is evaluated to determine if requirements have been met.

**Solid State Interlocking** – An interlocking system based on the use of processor modules. Three modules are required for a single interlocking. Each module performs the same functions. In the event the functions differ, the module triggers an alarm.

**SPAD** – Signal passed at danger is indicated when a train driver does not stop in time and proceeds past a signal displaying a red aspect.

**Statecharts** – A modelling technique which illustrates the logical flow of processes in a system. Statecharts enable the display of multiple hierarchical processes and the different states of system components. These states are triggered by events which are generated from transitions between other states. The state of an object is globally transmitted throughout the system.

**Station** – A geographical location of signalling field elements on a railway line where crossing movements may take place.

**Track Circuit –** An electrical circuit used to detect the occupation/presence of a train on a railway line.

**Train –** A motorised locomotive coupled to wagons used for the transportation of passengers or goods.

**Validation –** A process wherein the validity of a software system is checked. The software is checked to determine if the defined requirements and specifications have been met. Validation also confirms that the developed system meets user requirements and related standards.

**Waterfall method –** A software life cycle model which follows a strictly planned development process. The waterfall model mainly focuses on requirements engineering prior to the development of software. The model follows a phased approach wherein the results obtained in each phase are compared to the specified requirements. The quality of the system is evaluated and controlled at the output of each phase.

**WinCC –** Is a SCADA and HMI SIMATIC system from Siemens and is used to monitor and control industrial processes.

# 1. **INTRODUCTION**

## *1.1. Introduction*

Railways are an efficient method of land transportation as rail vehicles are used to transport passengers and goods at high speeds over long distances [1]. This is a cost-efficient method for improving transport networks in various countries. Railway systems consist of physical field equipment and interlocking software. The interlocking monitors and controls the operation of field equipment and guarantees the safe movement of vehicles in a railway system [2]. Railways are safety-critical environments wherein systems are well maintained but not regularly updated. This can be attributed to the immense safety regulations and cost implications in upgrading the complete system. Software systems used in rail environments are required to be fail-safe and ensure the system always responds in a safe manner. Subsequently worldwide interlocking systems are governed by safety standards of operation.

An electronic interlocking system is the most advanced interlocking technique for ensuring the safe movement of trains in a railway yard. Many countries employ variations of electronic interlocking systems based on the applicable software, railway layout and system requirements in the country of application. A research investigation has been conducted to determine if many existing electronic railway interlocking systems are based on fundamental interlocking principles and regulated safety standards. Common aspects amongst these systems are identified. Based on these results, the investigation focuses on defining a generic software solution for electronic interlocking systems which can be applied to any rail layout in varying countries. The application of software engineering methods in developing this solution is also examined.

## *1.2. Research Objectives*

Two fundamental goals will be researched, i.e. the development of a generic railway interlocking solution and the application of software engineering techniques in developing this solution. Firstly, the different types of interlocking systems used for railways will be investigated to establish a background on railway interlocking. Thereafter the current prevalence of electronic railway interlocking systems will be researched. Various electronic interlocking systems implemented in railways worldwide will be investigated in order to determine common aspects amongst varying existing systems. The outcome of this investigation will be used to identify fundamental

components that form part of an electronic interlocking system. These findings will then be used to determine if a generic solution for electronic railway interlocking systems can be defined and if this solution can be independent of the physical layout of the railway. A software interlocking model will then be designed following a software engineering approach to aim to validate the specifications for a generic solution.

The application of software engineering methods during the development of a general interlocking solution will be examined. Existing software techniques previously followed whilst developing an electronic interlocking system will be investigated. These techniques will be used as a guideline to determine which methods can be applied to the interlocking solution. Relevant software methods such as lifecycle models, design and architecture patterns and modelling techniques will be examined. The relevance of these techniques in assisting with the flexibility and maintainability of the generic interlocking solution will also be considered.

### *1.3. Research Approach Methods*

A literature survey is performed to examine the various types of interlocking systems and the existing methods implemented by each. Each method is extensively examined and common components amongst each are identified. Applicable software engineering methods applied to interlocking systems are also examined. These methods include the incorporation of formal methods to ensure the safety of a system. Design patterns and life cycle processes are followed to assist with maintainable software development. Existing methods are evaluated and the advantages and disadvantages of each are deduced. Thereafter the applicable methods are selected and used to develop an interlocking solution. An interlocking model of a generic solution is presented. The model is then validated through the use of software engineering test techniques. A comprehensive analysis of the interlocking model and the software methods incorporated into the design is discussed to determine if a generic electronic software interlocking solution can be established.

### *1.4. Thesis Overview*

The research hypothesis is structured to provide an extensive background on the development of railways and railway signalling. The signalling equipment and interlocking system that form part of railway systems are explained. Railway interlocking principles and guidelines which guarantee the safe movement of trains on railways are outlined. In addition, fail-safe interlocking principles which further ensure the safety of rail systems are presented. Existing railway interlocking methods and

applicable software engineering methods are noted. These methods are evaluated in order to determine their relevance in developing a generic interlocking solution. Software engineering methods such as life cycle models, design patterns and modelling techniques are analysed. Thereafter the interlocking solution model is presented. The relevant software methods applied are explained and the software design of the model is discussed. The software design includes a physical station layout, control table, Boolean interlocking function and software simulations. The simulation of the interlocking software using Siemens TIA Portal software is described. The software is then validated through the use of test cases. An analysis of the software simulation and generic interlocking model is examined. The application of software engineering techniques during the development of the interlocking solution is evaluated. Outcomes of the research investigation are discussed. Key findings are declared and suggestions for future work are proposed.

# 2. **BACKGROUND OF RAILWAYS**

## *2.1. Introduction*

Railways are used for the efficient transportation of passengers and goods [2]. Railway signalling systems are used to control and ensure the safe movement of vehicles on railways [1]. Railway systems consist of physical field equipment and an interlocking system. Field equipment refers to signalling elements such as signals, tracks, points, etc. that form part of the physical railway layout. The interlocking monitors and controls the field equipment and the movement of trains on a railway. The interlocking also establishes the safety of a railway system [2].

The development of railways to ensure the safe movement of rail vehicles is examined. A brief background on railway signalling highlighting key developments in improving railway safety is provided. Railway systems consist of signalling equipment and an interlocking system. The functions and relationships between the signalling equipment, interlocking system and signalling methods are described. Essential railway interlocking principles and fail-safe techniques that must be followed for the safe movement of rail vehicles are provided.

## *2.2. Railways*

A railway is a means by which vehicles with flanged metal wheels travel on an arrangement of metal rails. The configuration of metal rails and signalling equipment enable the transfer of goods at high speeds and over long distances. The first railways were primarily developed for the transportation of minerals (i.e. iron ore, coal, etc) from mines across far distances to roads and ports. To this day many railways still serve this purpose [1].

The age of railway signalling began in 1830 when the first railway for the transportation of passengers by locomotives was opened. This railway was called the Liverpool and Manchester Railway. The main concern of railway vehicles was the braking distance required to stop a train in time. This proved difficult for steel trains travelling on cast iron rails where levels of friction and adhesion were significantly reduced [1]. As a result, train control methods were required to ensure a safe distance between trains was always maintained. A few existing train control methods include the: telegraph order, wooden staff, Van Schoor token, pilot working, radio train order, track warrant system and signalling methods [3].

A telegraph order consists of a telegraphic code which is used between railway policeman to report the arrival and departure of trains within designated sections. The telegraphic code is produced by a

telegraph instrument which uses a needle which when moved to the left/right spells out words letter-by-letter [1]. A physical wooden staff is also awarded to train drivers. The wooden staff represents a movement authority, i.e. the driver with the wooden staff is permitted to proceed through a given section. Similarly, the Van Schoor token system consists of a disc which represents a token. The driver with the token has the authority to proceed through a given track section. In pilot working systems, an operating employee is appointed as a pilot and is required to accompany each and every train. The pilot acts as a human token amongst train sections [3].

Order forms are completed by train drivers for radio communication across specified track sections. These forms are part of the radio train order method of controlling trains. The track warrant system is a paper based system wherein train control operators are assisted by a live visual display of the station and field elements displayed on a computer screen [3]. Following this system, visual communication systems which provide information to train drivers have been developed. The visual information informs the driver about the availability and safety of the route ahead. This system is referred to as signalling. The fundamental purpose of railway signalling is to always maintain a safe distance between trains and to ultimately regulate the movements of trains on a railway network [3].

### *2.3. Railway Signalling*

Railway signalling refers to a system of protection for vehicle traffic on railways. This is achieved by the safe separation of trains and by preventing derailments and collisions between trains. A railway signalling system provides information to drivers through the use of signals which inform them of line conditions that can be expected ahead. In turn drivers are expected to interpret the signal indications accordingly. The signalling system provides supervisory control which enables train drivers to correctly direct and route trains to their destinations, re-route trains to avoid collisions and deliver passengers timeously to their destinations. Therefore a railway signalling system is responsible for the efficient movement of trains on a railway whilst ensuring the safe transportation of passengers or goods [1].

Originally, hand signals were given by railway policemen to warn train drivers of conditions ahead. Time intervals between which signals were given were incorporated to maintain a safe separation of trains at all times. Time intervals were monitored through the use of egg timers. At this point in railway development, railway safety was solely dependent on a set of rules and procedures which had to be strictly followed by train drivers [1]. Around the1840s fixed line signals were introduced to replace hand signals shown by railway policemen [4]. The signals displayed the same

information as shown by hand signals but were on a larger scale. Signals were fixed onto a post and indications were shown using an arrangement of moveable flags or discs [1]. At night oil lamps were used to provide signal indications [4].

In 1841 semaphore signals were developed and became the first standardised mechanical signal in use in the UK. Semaphore signals consist of a signal arm which when situated at different positions indicate different signal aspects. An arm in a horizontal position indicates a stop aspect. When the arm is positioned downwards at a 45 degree angle, a caution aspect is indicated. A proceed aspect is indicated when the arm is positioned further downwards at a 90 degree angle [4]. However, these arms easily break off and freeze in position during colder months [3]. Subsequently, coloured light signals mounted onto tall posts were later introduced. These signals were still manually operated by railway policemen. The main advantage of the fixed signals was the visibility of the indications from further away [1].

In 1921, the Institute of Railway Signal Engineers (IRSE) released a standard for three-aspect colour light signals, i.e. red (stop), yellow (caution) and green (clear). This standard was then applied to all future signalling developments [1]. Another significant development was the central control of signals and points from a central location in a station. Mechanical levers were used to control the elements. Later the levers were connected together in a particular configuration such that they were physically locked, i.e. the levers could only be pulled when safe to do so. An example of the system safety included that levers for two conflicting routes could not be pulled at the same time. In this manner railway safety was incorporated into the system. The configuration and fail-safe features of the levers were known as the interlocking. This feature became the basis for all signalling henceforth. By the year 1890 full interlocking became a legal requirement for all stations. By the 1890s, almost 40 different interlocking methods had been designed and patented [4].

Later electrical circuits were developed for track sections. These electrical circuits were incorporated into the rails and were able to detect if a section of track was occupied by a railway vehicle. During this period, the use of electrical components became more common. Electrical point machines were developed to move the points forwards and backwards. Through the use of electricity a larger area of points could be controlled. Due to the extensive advancements in electricity at the time, more aspects of the interlocking were developed using circuits involving electromechanical relays [4].

With the introduction of colour light signals and advancements in electric point machines there was

no longer need for big mechanical levers to control the electric switches. Lever frames were developed with small levers which could easily be flicked to control the switches. Later, a route setting panel was developed whereby train operators could control the conditions of the route through the use of two pushbuttons. The first button was connected to the start signal and the second to the destination signal. The interlocking system would then arrange for all the points within the route to be set into the correct position [4].

By the end of the 19th century, there was a firm development in the integration of electricity and microprocessors into railways. As the electricity networks spread the need for railways increased to support urban transport. Around 1960, electronic equipment was used for the remote control of the interlocking over long distances for which cables could not reach [1]. However, the manner in which electronic components would fail could not be predicted. As a result, microprocessors were integrated into the system to alleviate this issue and guarantee the safety of the system [4]. From the 20th century onwards computers began to be incorporated with electronic systems [1].

Software developed for railway signalling systems was originally designed to operate on a specific type of processor module. These were known as SSI (Solid State Interlocking) systems [5]. SSI systems were designed to provide an interlocking which is at least as safe as the old relay techniques [4]. The software was conventionally coded in assembler language. However, due to constant advancements in microprocessors this approach was not feasible. Later the signalling software was developed using high level languages which enables the software to be independent of the physical hardware and processor modules [5].

A railway signalling system consists of signalling field equipment and an interlocking system. Signalling field elements consists of signals, track circuits, points and other physical equipment which forms part of a station layout [2]. A station layout illustrates the physical arrangement of the signalling field elements and also indicates the routes permitted between signals [6]. The interlocking monitors and controls the functions of the equipment and is responsible for the safe operation of trains on railway lines [2].

### *2.3.1. Signalling Field Elements*

The physical and operational aspects of signals, track circuits and point elements are discussed. The operational logic and fundamental functions of an interlocking system are described.

#### 2.3.1.1 Signal Elements

Signals display different aspects to notify train operators of conditions ahead [7]. These aspects convey information such as speed limit, distance to the next station and the possible state of the next signal [2]. Signals are used as a method of indicating an "authority to proceed." Train drivers are aware of the interpretation and limit of this authority [8].

Signals can be either mechanical or light signals. Mechanical signals consist of arms and indications are displayed through different positions of the arms. Light signals display different lights which are also known as aspects. Three colours are universally used as aspects for colour light signals, i.e. Red, Green and Yellow. A red indication specifies that the train driver must come to a complete stop just before the signal [4]. The driver may only pass after a special authority of movement has been obtained or the signal is cleared to a proceed aspect [9]. A green aspect indicates that the driver may proceed past the signal and a yellow indication denotes caution [4]. A yellow indication informs the train driver that he must reduce speed and proceed at caution as the next signal is displaying a red aspect [9]. Although the classifications of these aspects are universally standard, the aspects may be used in varying arrangements in different countries [4].

A three-aspect signal is displayed on a design plan as a mast connected to three circles which collectively represents the signal head. Each circle represents a lamp and bars within the circle indicate the colour. A horizontal stroke indicates a red aspect, a diagonal line indicates a yellow aspect and a vertical line indicates a green aspect. The symbol used is shown in Figure 2.1. In design plans, tracks lie horizontally. Similarly instead of being drawn upright, signals are drawn at 90 degrees parallel to the track circuit. The most restrictive aspect (i.e. red) must be placed nearest to the drives' eye level. As a result the red aspect is actually the lowest aspect, green is the highest aspect and yellow is placed between the two aspects [4].

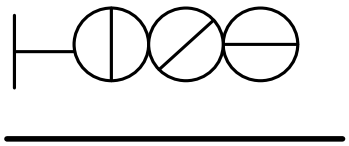

Figure 2.1: Signal design plan - Symbol 1 [4]

The symbol shown above indicates train movement from left towards the right direction. However the symbol shown in Figure 2.2 is applicable to train movement from the right heading towards the

left direction [4].

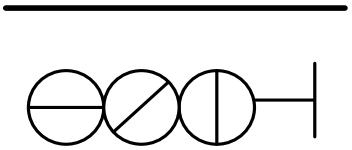

Figure 2.2: Signal design plan - Symbol 2 [4]

#### 2.3.1.2 Track Circuit Elements

Track circuits are used to detect the presence of trains across track sections. Signals are physically placed between track circuits in a station layout [7]. A Track consists of steel rails on which the train travels upon. Sleepers are used to hold the rails in position and the correct distance apart. Sleepers are arranged in a transverse pattern under the rails. A Pandoral clip is then used to hold the rail in place. The Pandoral clip is piece of steel rod bent into a particular shape and acts as a spring to keep the rail in place. The track sections are manufactured in short lengths and are joined together by a fishplate. This enables the formation of a continuous track [4].

#### 2.3.1.3 Point Elements

Points refer to an arrangement of track sections that allow junctions to be made in a track. A junction is a section wherein track lines converge or diverge. This junction is known as a point. A point's machine is required to move the points to set the required path [4]. Points can either lie in the normal or reverse positions [7].

A point consists of outside rails which are fixed and are called stock rails. There are two inside rails which can move from side to side at one end. These rails are referred to as point or switch blades. The end that can move is known as the toe of the points. The two blades are held apart by a stretcher bar. By moving the position of the point blades the train may travel straight in the current direction it was travelling. This is also referred to as the normal points position. By moving the blades into the reverse position the train may change its direction of travel and diverge onto another line. The area where one rail crosses over another is referred to as a crossing [4].

The symbol for points in a design layout is shown in Figure 2.3 and 2.4 below. A points set may lie in either one of two configurations and is dependent on the track layout. For the points symbol configuration shown in Figure 2.3, the points set would have to lie in the normal position for train movement from A to B and in reverse for movement from A to C [4].

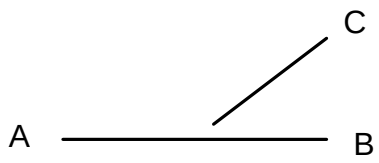


Figure 2.3: Points symbol - Configuration 1 [4]

However if the above points configuration was changed to the configuration below, the points set would have to lie in the normal position for train movement from A to C and in reverse for movement from A to B [4].

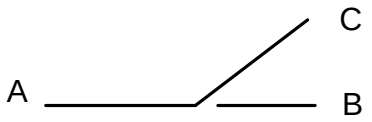


Figure 2.4: Points symbol - Configuration 2 [4]

*2.3.2. Interlocking System*

An interlocking system controls and monitors signalling elements in a railway layout. The interlocking establishes the safety of the rail system. As technology has progressed so has the type of interlocking systems and technologies used. There are four fundamental types of interlocking: mechanical, electro mechanical, relay and electronic. In a mechanical interlocking system wire pulls and mechanical levers are used to control elements. Collisions are avoided with the use of mechanical blocks which restrict the actions of elements. Electro mechanical interlocking systems consist of a combination of mechanical and relay interlocking systems. Wire pulls are replaced by relays which control the movement of elements. In relay interlocking systems, wire pulls are replaced by pushbuttons on operator panels and track sections enable electrical monitoring [10]. A relay consists of a number of electrical switches or contacts. Each contact consists of a pair of arms which acts as the switch. If the two arms are not in contact, the contact remains open and no current can flow through the switch. However, if the two arms are pushed together by an electric current, the contacts close and current flows through the switch [4].

By interconnecting the relays, the fail-safe logic required by the interlocking can be built into the system. However, railway interlocking systems consist of thousands of relays and require a great deal of space to store all the relays. In addition, the relays are physically wired to the signalling field equipment and cannot be stored far away from the station layout. This significantly limits the size

of the area and elements that can be controlled. The copper wires used to connect the relays to the field elements are also a significantly high cost factor. Subsequently, Remote Control Systems have been introduced. These systems enable the monitoring and control of a large area and large group of elements over a single pair of wires. This in conjunction with microprocessors developed into an SSI (Solid State Interlocking) system [4].

SSI systems were introduced in the late 1980s. The SSI equipment for one interlocking consists of several modules housed in a standard sized cubicle. An MPM (Multiprocessor Module) performs the interlocking functions and three of these modules are required in one cubicle. Each module has identical hardware and software. Each MPM continuously communicates with its two partners comparing operations and memory functions. All three modules are expected to operate in the same manner. In the event, the behaviour of one of the modules differs; the specified module intentionally fails and triggers an alarm warning [4].

Four microprocessors are incorporated into each module. One processor in each module processes the interlocking logic and is responsible for communication with the other processors in other modules. The second and third microprocessors manage communication with the field elements. The fourth microprocessor manages communication with the other SSI systems [4]. However, SSI systems were solely developed for specific processor modules and with constant technological advancements in microprocessors this method is no longer sustainable [5].

As a result electronic interlocking systems became the most advanced method of developing a railway interlocking system. Electronic systems can be controlled by a range of microprocessors and Programmable Logic Controllers (PLCs) [10]. These microprocessors are programmed in high level languages and are independent of the physical hardware components and the processor modules [5]. These systems electrically monitor railway functions online via a computer screen which provides instantaneous visual feedback to operators [10].

### *2.4. Interlocking Principles*

A fundamental philosophy of an interlocking system is to permit train movements with maximum safety whilst optimising and increasing the capacity of a railway line. The principles of electronic interlocking systems are designed to prevent accidents or collisions on railway lines [11]. Relevant principles regarding an interlocking system are provided. The calling and setting of routes in a station and the operation of signalling elements in route functions is explained. Thereafter, general

interlocking safety requirements are specified.

### *2.4.1. Interlocking System*

Safe train movement is guaranteed through the application of railway signalling principles and the monitoring and controlling of signalling elements. Signalling elements can be in certain defined states, i.e. a track section can either be occupied or free, a main signal can display either a red, yellow or green aspect and a point's machine can either lie in the normal or reverse position [12].

The interlocking monitors the operation of the signalling elements. For example, the interlocking monitors the occupation and clearance of track sections in order to minimize the risk arising from the loss of train detection. This ensures that a route request is issued only when safe to do so. Points and other movable infrastructure are set to and locked in the correct position by the interlocking for any movements which are authorised over them or for which they provide protection [13]. The movement of a train is limited by the current signal aspect ahead of the train [6]. Opposing signals are not operated simultaneously as this may lead to an accident or collision. Signals cannot display a proceed aspect until all points are detected in the correct positions and locked for a route. A set of points must not change its position while a train is travelling over the set. The points can only be moved after the train has cleared the junction and the associated signal is set to a stop aspect [11].

### *2.4.2. Routes*

An important safety principle for route setting states that two routes cannot share any portion of track. If two routes share a track section, they are defined as conflicting routes and cannot be set at the same time [14]. The principles regarding route functions are described in terms of route request and route setting operations.

#### 2.4.2.1 Route Request

A route request is determined by defining the starting and destination signals of a route. These signals must be unique and the specified route must be free at the time of the request. A free route indicates that the previous route request has successfully been completed. The start signal of the given route must be free and currently displaying a red aspect. A red aspect indicates that the signal is operational and not being used by another route. The track sections required by the route request must be checked for availability. The availability of the points is then checked and thereafter the current position of the points is determined. If the point is in the incorrect position, the point's

machine is instructed to move the point into the correct position [14]. A route request which requires an overlap can only be issued provided there is no conflicting route or overlap set that conflict with the overlap. Where a preferred overlap is specified and is available this overlap shall be set [13].

If the route request is cancelled, no further actions may take place. A cancellation request can only be executed up to this point prior to the locking of the route [14]. Thereafter the route is locked and the signal indications for the route are set [15].

#### 2.4.2.2 Route Setting

Prior to setting a route, all track and overlap sections along the route should be free. All points in the route should be set, free, locked and checked. All conflicting and opposing signals should be set to a red aspect. Signals within the route should be off (clear or not showing any aspect). The route should be isolated from possible conflicting routes [12].

Provided the points are set in the right direction and locked and the track sections are cleared of trains, the route is proven clear. Thereafter the route is set and the entry signal of the route is cleared to a yellow or green aspect. The colour of this aspect is determined by considering the aspects of relevant signals ahead and speed restrictions within the route. After all the requirements have been satisfied, the route is locked. The points machines are locked first, thereafter the track sections and finally the route. After the route has been called and a train begins its occupation, the route can only be cancelled once the train has come to a complete stop in the route or has completely passed through the route [8].

### *2.4.3. Signalling Elements*

Fundamental operational principles regarding key signalling elements (i.e. signals, tracks and points) are described.

#### 2.4.3.1 Signals

The interpretation for three-aspect signal indications is explained. A red aspect indicates that the following track section is occupied. A yellow aspect states that following track is clear but the next track section is occupied and a green aspect indicates that the following two track sections are clear [2]. The interlocking monitors and controls the signal aspects to ensure adequate warnings are provided for train drivers to stop at a danger signal or to reduce speed in time [13]. Signal aspects are always set to red in emergency situations [16]. Where signals are visible beyond the current

approaching signal, more restrictive aspects must be applied to limit the risk of incorrect interpretation of the signal aspect. A signal shall only clear from its most restrictive aspect (red) when the appropriate conditions are fulfilled. These conditions shall continue to be confirmed until the train has entered the route to which the signal applies. If the conditions are not proved, the signal reverts to a danger indication and displays a red aspect [13].

A start signal can only display a proceed aspect provided the route between the start and destination signals is clear. The defined overlap must be unoccupied and all points in the route must be set in the required position and locked. The start signal of any directly opposing route must be proven to be displaying a red aspect before the start signal can display a proceed aspect [8]. In addition, for a signal to obtain a proceed aspect, the next signal in the same direction of movement must continuously display a proceed aspect. If no aspect is displayed on a signal (e.g. due to a lamp failure), the preceding signal must show a red/ stop aspect [11]. In the event of a failure of a signal aspect which could lead the driver to interpret the signal as less restrictive, the aspects shall be tested. If the test indicates that the aspect is non-functional the signal aspect must be reduced to the most restrictive aspect, i.e. a red aspect and the proceeding signal aspect must be restricted to ensure the signal is not passed at danger [13].

In exceptional circumstances it is permissible for a signal aspect to be extinguished in the event no route has been set up to the signal, no train is approaching within viewing distance of the signal aspects or no confusion will be caused by an unlit signal not showing any aspect indications. If a signal needs to be replaced, it must first be set to display a red danger aspect and the associated route must be cancelled [13].

#### 2.4.3.2 Track Circuits

When a track section is unoccupied a logical 1 value is sent to the interlocking. Conversely, when a train occupies a track section a logical 0 is sent to the interlocking. This procedure is followed to prevent against wire breaks and ensure the accurate detection of trains on track circuits [2].

#### 2.4.3.3 Points

Points should not be set to normal and reverse positions simultaneously as this damages the mechanism of the point's machine controlling the points. If a point is occupied, then it is locked in its current position [7]. Points are only allowed to move provided they are free of and have not been locked. The locking of a point on a route is maintained until the train passes the point's machine

and protection is no longer required [13]. If a route has been set and reserved all the points required by that route are locked in order to guide the train along the route [16].

### *2.4.4. Principles for Safe Train Movement*

A few of the requirements defined for the safe movement of trains on a railway are summarised. Principles for the safe movement of trains refer to the prevention of safety-critical events and ensuring the interlocking responds in a safe manner. Safety-critical events include collisions between trains travelling on the same track or in the same direction [17]. In the event a safety-critical event cannot be prevented, the interlocking must then act to minimise the loss of human life and infrastructure damage by safely controlling the operation of the signalling elements. The signalling element functions are controlled by preventing the setting of routes with the affected elements [13].

The POSMOR (Principles of Safe Movement on Rails) state that railway infrastructure must be functional and in good condition such that railway vehicles can safely travel upon the tracks. Railway vehicles must conform to applicable loading specifications. Defined sections in a station must be clear, i.e. sections must be free of all obstructions, required points must be set in the correct position and the complete train must pass/leave a section. Departure and destination points in signal plans and design layouts must be clearly defined. Train drivers and control operators must be aware of the limit of authority given to a train to travel across a designated section [3]. A movement authority shall not allow conflicting train movements along a route. Whilst moving, train drivers must adhere to speed instructions and trackside indications. A train driver must always halt train movement when and where scheduled to do so [11].

## *2.5. Fail-Safe System*

A fail-safe system is defined as a control system which either responds in a safe manner by forcing a system into a predefined safe state or continues to function safely after a failure [18]. A safe state refers to the system executing an appropriate response in the event of a critical situation. The fail-safe logic of the interlocking is built into the system. This logic is extracted from the control table conditions which only allows the interlocking to set a route provided all safety requirements are met first [14]. A signalling element that fails should fail in a silent manner and should not affect the operation of other elements. In the event of an element failure, a route which requires this element should not be allowed to be set [15]. For an element to fail in a safe manner it must either perform a right or a wrong side failure. An example of a right side failure is when a signal aspect indicates red

when it should actually be showing green. This is incorrect but not a hazardous situation. An example of a wrong side failure is when a signal aspect is displaying green instead of a red indication. This could result in a critical accident [4].

The fail-safe performance of a system can be determined by performing extensive verification through simulation or model checking techniques [15]. Simulations of safety-critical events can also be performed to verify the behaviour of the software during these events. Safety-critical situations arise when the maximum number of trains on a route is exceeded, from train derailments, collisions at level crossings or collisions between trains, etc. Faulty sensors which automatically default the system into a fail-safe state can also lead to a safety-critical event. signalling elements can also cause safety hazards through the incorrect or unrequested switching of signal indications and position of point's machines. Additional hazardous scenarios include if a train does not follow the assigned route specified or if a train driver proceeds through a red signal aspect. These situations all result in a safety-critical situation wherein the interlocking must respond in a safe manner [19]. Rail safety is further discussed through the description of applicable safety standards and the integration of safety principles into the interlocking and operation of signalling elements.

### *2.5.1. Safety Standards*

A safety standard can be achieved by assigning different safety levels to different system functions based on their level of criticality. This is determined based on the overall contribution of the function to the system behaviour as defined by the European Committee for Electrotechnical Standardization (CENELEC) [20]. Developers must follow the CENELEC standards for railway applications to produce safe and reliable systems. The applicable railway applications standards are: EN 50126 (The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS) [21]), EN 50128 (Software for railway control and protection systems [22]) and EN 50129 (Safety related electronic systems for signalling [23]). All graphical system models must be verified to determine required SIL (Safety Integrity Levels) described by standards EN 50126 – EN 50129 and IEC 61508 (Functional safety of electrical/electronic/programmable electronic safety-related systems [24]). The SIL represents the probability of a system to execute specified safety functions within a specified time interval [16]. Detailed information on the specifications of the applicable CENELEC standards for railway applications mentioned above are described in Appendix A.

### *2.5.2. Fail-Safe Interlocking*

The interlocking system must satisfy safety requirements for both software and hardware. Software must be developed according to defined standards and in accordance with key fail-safe railway interlocking principles. Safety certified components can be used to satisfy the hardware requirements for the interlocking. The hardware (i.e. signalling field equipment) is also instructed to operate in a defined manner in the event of failure [2]. Key functional interlocking safety requirements applicable to the development of a generic interlocking solution have been outlined.

These requirements state that route requests received by the interlocking must first proceed through a series of checks which are used to verify the behaviour of the interlocking system. These checks ensure that no train can be directed into a route already occupied by another train [15]. To avoid a collision, two trains should never be located in the same track section [12]. Subsequently any two routes cannot share any portion of a track circuit [14]. Additional requirements state that a safe separation between trains must be ensured by providing an overlap beyond the end of a route and the destination signal [25]. A train should sequentially occupy all track sections and overlaps of its required route until it reaches the last track section, i.e. the train must completely clear the route before another train maybe allowed [12].

### *2.5.3. Fail-Safe Signalling Elements*

Signalling elements are the field elements within a station such as signals, tracks, point's machines, etc. During a safety-critical event the field elements are expected to perform in a predefined manner to limit damage to infrastructure. A few examples highlighting the expected behaviour of elements are provided, i.e. points should not be switched and should remain in their current position. Signals must be switched to a red aspect to instruct train drivers to stop and to prevent any other trains from entering the critical section [19]. For safety purposes, a dark signal in a railway yard must be considered as a signal displaying a red aspect. Train drivers must not proceed past this signal. These expected element operations are built into the interlocking to guarantee the safety of a rail system [14].

## *2.6. Railway Signalling in South Africa*

In 1858 the contract to develop the first railway in Cape Town, South Africa was awarded to the Cape Town Railway & Dock company. Two years later in 1860, the first public railway from point to Durban was developed. This railway was approximately 3 kilometres long and ran along the Bluff in Durban. Oxen were used to haul the trains. Trains used to transport iron ore along the ore

line in the Western Cape are a record distance of 7km in length. The ore line itself is 861 km's long from Sishen in the Northern Cape to Saldanha in the Western Cape. Years later in 1892, the first train from Cape Town to Johannesburg via Bloemfontein was commissioned. Later in 1963, steam trains were introduced which travelled from Cape Town to Johannesburg [3].

From the 20$^{th}$ century onwards, approximately 17 000 kms of railway had been signalled and roughly 1 600 interlocking stations were in operation. The original interlocking is known as Spoorplan and consists of mechanical relay interlocking systems [3]. This is a geographical based interlocking system and is expensive to maintain. Subsequent to this, Siemens introduced the SICAS (Siemens Computer Aided Signalling) S7 electronic interlocking system. This system has been implemented on mass transit lines and can easily be adapted for varying applications and environments. SICAS S7 is strictly based on the SIMATIC S7 PLCs (Programmable Logic Controllers) which greatly improves the efficiency and flexibility of the system. The most recent interlocking system developed by Siemens is the SICAS ECC (Element Control Computer) electronic interlocking system. This system provides a range of interface operations for electrically monitoring and controlling signalling field elements [26].

Railway companies in South Africa began to formalise around the 1840s. Many small railway companies such as Natal Railway Company, Wynberg Railway Company, etc. began to develop. In 1981 the South African Transport Services was introduced. Years later this company was privatised and in 1990 became known as Transnet. Transnet Limited is a public company which represents a large transportation network. This network consists of various divisions such as Spoornet, Petronet, Metrorail, etc. These divisions are responsible for the transportation of passengers or freight [3].

# 3. **EXISTING RAILWAY INTERLOCKING METHODS**

## *3.1. Introduction*

The transportation of passengers by railways provides a safe and affordable means of transport in developing countries. Railways enable the transportation of goods and people in a minimal amount of time. Large goods can also be effectively transported by railway vehicles using fewer resources such as time and costs. The main component of a railway system is the interlocking. The interlocking controls the main operations in a railway system and must ensure safety standards are met. Hardware safety requirements are met by using approved hardware components. The interlocking is required to institute the safety requirements of the software. These components function collectively to produce a secure system that guarantees the safe movement of railway vehicles [2]. Although the basic principles of railway signalling are universal, the way in which signalling has developed in varying countries differ. For example, the development of railway signalling in Britain differs from approaches in Europe and America. [1].

The concept of a universal computing interlocking system has been proposed by Lutovac [14]. This system is independent of the physical hardware layout of a railway yard and can be used in any application. The interlocking system would encompass all functional and safety requirements of a railway system [14]. This model suggests that a generic railway interlocking system can be developed. Common aspects amongst interlocking systems implemented worldwide can be identified. These aspects in conjunction with interlocking principles and safety regulations can be combined into a generic interlocking solution for railways. This solution can then be customised to suit varying applications in varying countries. This proposition is used as a guideline for the literature investigation.

The goal of this literature review is to identify existing methods used in developing a railway interlocking system using software engineering techniques. Current methods implemented have been researched, evaluated and analysed to ascertain which techniques shall be applied. The review is intended to effectively compare techniques and determine a suitable environment for developing a generic interlocking solution. The review is structured such that the details of existing methods for developing railway interlocking systems are first described. Interlocking components and methods of operation are provided. Thereafter applicable software engineering methods and modelling techniques used are discussed. Software engineering test techniques are suggested to validate the interlocking software. The advantages and disadvantages of these methods are evaluated to

determine the most beneficial techniques. The selected techniques are then reviewed and adapted to suit the topic under investigation.

### *3.2. Current Electronic Interlocking Systems*

An interlocking system consists of signalling functions which control the movement of trains in a rail yard. This system is referred to as interlocking because setting one group of movement locks out other movements [6].

Banci identifies the components required to develop a railway interlocking system as a station yard layout, condition table, statechart design model, model verification, software code and testing [15]. The yard layout displays the geographical layout of the elements. The condition table is based on the yard layout and describes the interlocking principles. Statecharts are used to model the elements and illustrate how these elements function collectively to form an interlocking system. Faults are injected into the model to verify the behaviour of the system. The software code is then generated and extensively tested in order to guarantee the safety of the interlocking [15].

Moller suggests physical layouts, control tables and ladder logic in specifying and modelling a railway interlocking system [27]. The physical layout illustrates the positioning of the hardware elements such as signals, points, tracks, etc. These hardware elements have attributes. For example, track circuits can be either occupied or unoccupied. Control tables are derived from the physical layout and define the behaviour of hardware elements. Control tables consist of a set of rules of operation that must be obeyed in a railway yard. The physical layout and control tables are used by software developers as input specifications for the software. The software is programmed using ladder logic. Ladder logic is a discrete time linear system. It provides a graphical representation of a logic circuit in the form of Boolean equations. Verification conditions are then designed to validate the software. These conditions are based on entries in the control table and interlocking principles [27].

Interlocking software for electronic railway systems is developed using ladder logic as proposed by Kanso [7]. The software is tested through simulations designed and monitored by signalling engineers. The layout of the railway yard and railway interlocking principles are used as input to the software. Railway yards are composed of track segments, signals and points. Routes consist of connected track segments which begin and end at signals. The routes on a railway yard are defined by a control table which is deduced from the yard layout. The interlocking implements the

conditions defined in the control table as Boolean assignments in ladder logic. Propositional logic conditions are defined in ladder logic to verify the safety properties of the system [7].

Eris states that a railway system consists of a TCC (Traffic Control Centre), interlocking system and field equipment [2]. The movement of trains in a railway is monitored in a TCC. The TCC communicates with the interlocking by sending commands such as route requests. The interlocking in turn sends the current state of the system to the TCC which is displayed in the form of a graphical layout. Field equipment refers to signals, tracks and points that are placed in a railway yard. The interlocking monitors and communicates with this equipment. The interlocking system evaluates the route request received from the TCC, determines if the route can be requested and then sets the positions and indications of the required field elements. State transition charts are used to model the functions of the field equipment. Input-output interlocking functions are deduced based on these states [2].

The conversion of the operational, functional and safety requirements of an interlocking system into a standardised CIS (Computer Interlocking System) has been presented by Lutovac [14]. The CIS monitors and authorises the movement of trains across a railway. It determines the available routes and signal indications that need to be set. The interlocking software in the CIS is separated into two groups: application driven data and general interlocking software. Application driven data refers to a station layout and control table. The control table must be derived from the station layout by a signal engineer. The data in the control table is then converted into the interlocking software through the use of Boolean interlocking functions. The interlocking software is independent of the application. Signalling principles and railway regulations are incorporated into the interlocking software. A graphical screen layout is designed to illustrate the performance of the system [14].

Minkowitz states that an interlocking is a centralised system which controls the transmission and reception of information from field elements in a rail yard [28]. This information and the state of each element are stored in memory. The interlocking is composed of configuration data and source data. The configuration data is defined for a particular collection of field elements and consists of geographic information regarding the elements and the application of interlocking principles to these elements. The source data creates a visual interpretation of this data at run time. The source data is developed from defined expressions and statements defined by signalling engineers which are used to communicate with the interlocking memory [28].

Winter suggests the development of a principle model and an interlocking model to represent an

interlocking system [6]. The principle model consists of the functional requirements of the signalling field elements and the interlocking model consists of the interlocking functions. The interlocking model identifies the key objects in the layout and the relationships between the objects in terms of functions. These functions are strictly based on the signalling objects in the layout. For a different station layout, the functions would change and have to be defined again. Safety properties are built into the principle model. However the safety variables are derived directly from the station layout and do not ensure that all possible critical events will be covered. Model checking is performed with the aid of test events based on laws of train movement. These laws are assumed and do not ensure all possible events are dealt with [6].

### *3.3. Applicable Software Engineering Methods*

Relevant software engineering methods are subdivided into: formal methods, life cycle methods, design patterns and software modelling techniques.

#### *3.3.1. Formal Methods*

Banci has considered the approach of using formal methods to develop a distributed railway interlocking system [15]. This method incorporates the use of statecharts to model the interlocking. Formal methods are used to accurately specify the interlocking principles which guarantee the safe establishment of routes for train movement. A geographic point of view based on the physical layout of elements is suggested. In this approach each element (signals, tracks, points, etc.) is modelled as an object. These objects function collectively as the interlocking. There is no central database of interlocking principles in this object-oriented approach as all element functions are modelled as separate entities [15].

#### *3.3.2. Life Cycle Methods*

Existing applications of agile and waterfall software life cycle models are considered. Jonsson analyses the use of agile practices for software development for European railways [29]. Agile methods are a life cycle model followed during the software development process. These methods follow an iterative development process. Each iteration produces a usable piece of software that can be released to the customer. Each release goes through the complete development cycle of requirements engineering, design, implementation and testing, Software is developed incrementally and can be validated during early stages of development. There is no limitation to the number of iterations. Agile methods reduce the time and complexity of producing safe systems in hazardous environments [29].

The waterfall life cycle model is a planning-driven process model for the development of software. This model focuses on extensive requirements planning ahead of project implementation. The waterfall model employs a strict phased approach in developing software. Phases are clearly defined and must be adhered to throughout the development process. In each phase, results obtained are compared to those that are actually required. This is known as verification and validation. Verification checks if the system has met its requirements and validation checks if the system has met user requirements. The waterfall process assesses and controls the quality of software produced during each phase [30].

### *3.3.3. Design Patterns*

Railways are considered real-time systems which have bounded response times. These are safety-critical systems, i.e. if the system does not perform in the expected manner during a fault, materials or human lives can be lost. A pattern is a software template which acts as a building block during the development process. A pattern refers to the software itself or to a suite of software processes. Patterns identify key aspects of a design structure and summarise previous successful engineering work which can be reused. Zalewski suggests a top-down approach of applying design patterns when designing a real-time software system [1]3. The requirements stage assists in identifying key aspects of a system. The requirements are generally expressed in terms of inputs/outputs. Zalewski assumes that in addition to the functional requirements, every real-time system includes processing and timing components. Together these components form a general architectural pattern for real-time systems. The general real-time architectural pattern consists of a processing interface, user interface, communication link and a database. This pattern can then be adapted for specific real-time applications [31].

Eriksen declares that an interlocking system requires a GUI (Graphical User Interface) through which the state of the interlocking and the system can be monitored and controlled by train operators [10]. This suggests the application of a model-view-controller architectural pattern which consists of three distinct layers. The model layer stores the system data and performs logic operations. The model layer is only aware of its layer. The signalling elements (i.e. signals, tracks, and points) are created in the model layer. The view layer is aware of the model layer and graphically displays the information stored in the model. The view layer acts as a GUI to the operators. The view layer also displays information communicated between the user and model. The final layer is the controller layer which processes and responds to events in the application. The controller can create changes in both the model and view layers if necessary [10].

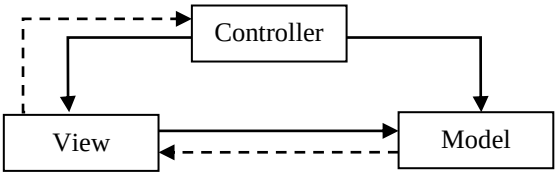


Figure 3.1: Model-view-controller architectural pattern [10]

Eriksen further suggests modifying the pattern to create a presentation layer. This layer combines the view and controller layers. The presentation layer is then responsible for displaying the information stored in the model and responding to system events [10].

### *3.3.4. Modelling Techniques*

The use of software modelling techniques in representing an interlocking system is outlined. Some of these techniques include the use of: Statecharts, Class diagrams, Automation Petri Nets, Fusion modelling and Z notation. Techniques for the verification of system models and software are also considered.

Statecharts have been used by Banci to model objects in a railway interlocking system [15]. The functionality and communication between objects are modelled as statecharts which collectively represent the interlocking. Statecharts are extensions of FSMs (Finite State Machines). FSMs are hierarchical parallel state machines which interact with each other. State machines are triggered by events. These events are generated from transitions between states and can refer to states of other machines or to global variables. Global variables are used to send state information throughout the system. UML (Unified Modeling Language) state machines and diagrams are methods of expressing statecharts [15].

Eriksen suggests that upon analysis of the railway domain, the signalling elements and their functions can be modelled to describe the operation of the interlocking [10]. Elements are modelled as objects and functions as methods in an object-oriented approach. In this approach, both elements and their functions are modelled together in a UML class diagram [10]. Class diagrams illustrate the static structure of a system by modelling the features and relationships between entities as classes. Objects which have similar features are modelled as abstract classes. Abstract classes allow code to be reused in a system [30].

Automation Petri Nets (APNs) are commonly used to model safety-critical systems such as

electronic railway interlocking. APNS are an extension of Petri Nets (PNs) [16]. A PN is a mathematical modelling tool commonly used in graph and analytic theory [32]. Yildirim uses an APN to graphically model a railway yard [16]. Railway elements such as signals, tracks and points are represented as places (Pxxx). Trains are modelled as tokens. Transitions (txx) represent attributes of elements such as the red indication of a signal or the position of a point. Events ($\chi$) refer to actions such as a route request. An interlocking table consisting of all the routes ($\chi$), required signal indications (txx) and positions of points (Pxxx) is constructed. This table is then used to develop the APN models. The APN models are then converted into a Ladder Logic Diagram (LLD) and a SCADA (Supervisory Control and Data Acquisition) interface is designed to test the software [16].

An interlocking configuration language has been developed by Minkowitz using Fusion modelling with an extension of mathematical notations [28]. This method employs a model-based formal specification technique which integrates easily with object-oriented techniques. The configuration language is composed of semantics specification and syntax specification. The syntax specification defines the structure of the system data and is used as a model on which the semantics specification is based. The model is graphically illustrated using Fusion object model notation. The semantics specification defines the interlocking functions in terms of operational processes executed on the syntax specification. The semantics are defined using Fusion schema notation. Following an object-oriented approach, a set of objects is categorised into a class and the relationship between objects is modelled as a set of tuples [28].

Zafar proposes the analysis of the safety properties of an interlocking system through the use of graph theory and Z notation [33]. This approach is specifically applied to fixed block and moving block interlocking systems. The fixed block interlocking system divides a station layout into fixed blocks and components which are separated by signals. The components within the fixed block are highly dependent on each other. In a moving block interlocking system, the area that a train occupies and the distance in front of it forms part of the moving block. No other train is allowed to enter this block. Z notation is based on standard mathematical notation used for the specification of abstract properties. This is fundamentally different to description languages [33]

Johnston states that an interlocking system must be verified against the signalling principles to ensure the system functions in a safe manner [17]. This can be achieved through model checking and the assistance of a model checking tool. Model checking is a mechanical technique which

assesses the complete environment of a system and evaluates all possible behaviours. The tool then notifies the user if the model violates a given requirement. The toolset includes a track-layout editor, control table generator and a control-table editor. The track layout and control table are inputs into the toolset. The tool then automatically generates a formal model of the interlocking design. Robinson employs the use of the NuSMV model checker which formally verifies finite state transition diagrams [17].

The interlocking software can also be verified through the use of software simulations. Software simulations can be validated using software engineering test techniques to prove the correctness of the software. The adequacy of test techniques is based on the small set of test cases designed to test the software. Dynamic test techniques such as coverage-based testing, fault-based testing and error-base testing can be implemented as these techniques require the execution of a software program. Coverage-based testing evaluates the completeness of the software wherein a number of instructions must be executed or branches followed during the execution of a program. Coverage-based testing also involves the evaluation of program variables through the execution of the software functions. Faults can also be injected into the system to determine the adequacy or performance of the system. Fault-based testing predominantly focuses on the identification of faults in the software. [30]. Test stages such as unit testing, integration testing and system testing assess software functions. In unit testing, individual system components are tested. These components are then incrementally integrated into a complete system. This procedure is then tested in the integration testing phase. A system test refers to the testing of software against the user documentation and requirements specification after the integration testing has been completed [30].

### *3.4. Evaluation of Existing Methods*

Applicable existing research methods discussed in Section 3.3 are evaluated. The examination of these methods is structured as above, i.e. first the analysis of interlocking methods and then the examination of applicable software engineering methods. The advantages and disadvantages of each approach method within are considered.

#### *3.4.1. Interlocking System*

Banci proposes that a yard layout, condition table, statechart design model, model verification, software code and testing are components required to develop a railway interlocking system. A statechart can be used to independently and collectively model the components in an interlocking system. A statechart model effectively illustrates how interlocking functions operate collectively to

form an interlocking system. However, modelling all the components can result in a large number of statecharts which increases the complexity of the model [15]. In addition, Banci suggests following a geographic approach in developing a distributed railway interlocking system. However, this system does not contain a central database or central equipment which manages the system. In this model the system depends on the geographical layout of the field elements. It also depends on the distribution of the interlocking principles to elements which model the physical field elements. Therefore the interlocking is dependent on the physical railway layout and can only be used for that specific layout. This makes the interlocking hardware dependant and inflexible for use in other station layouts with different elements [15].

Moller suggests physical railway layouts, control tables and ladder logic in specifying and modelling a railway interlocking system. The software is designed using ladder logic. Although ladder logic provides an easily understandable graphical representation, at times a large number of Boolean equations are required to accomplish simple tasks [27]. Kanso also recommends ladder logic for developing interlocking software for railways. The interlocking software is verified through simulations. Simulations are an effective technique for testing software as it mitigates the need for testing on expensive hardware and a greater level of detail can be achieved through simulations. Logic conditions are used to test the safety functions of the system. The logic conditions are similar to Boolean conditions used in ladder logic [7].

Eris proposes that a railway system mainly consists of a TCC, interlocking system and field equipment. State transition charts are used to model the functions of the field equipment and interlocking system. Although state transitions charts increase the ease of traceability in operations, an increase in inputs and states results in more difficulty in generating a state transition table of all the elements. This is especially evident in critical situations [2].

An advantage of the CIS system suggested by Lutovac is that the CIS is independent of the physical hardware layout of the station. This allows it to be used in any country of application. The software is flexible and can easily be adapted. The size of the system is determined by the control table and can therefore be optimized for any application [14]. Although signal engineers are required to complete the control table, simple and easy methods for defining a control table and screen layout are proposed. This reduces the time required for design and input of data. The software is composed of interlocking functions. The input to these functions is the data in the control table. If the control table contains a large amount of data the number of interlocking functions will increase

substantially. Nonetheless common variables can be found between functions. This increases the quality output of the system and reduces the probability of errors [14]. Railway and signalling principles are incorporated into the CIS which assists in ensuring the interlocking meets safety regulations. Although these principles are incorporated, signalling engineers are still required to design the station layout, determine the control table and validate the final system [14].

Minkowitz identifies an interlocking as a centralised control system. This system is then separated into configuration data and source data thereby providing a modular and flexible framework. Additional field elements can always be added to the configuration data and would not directly affect the source data. The source data would then just access this data from the memory and graphically display its state. This creates a significantly modular structure independent of the physical hardware. This would also enable easy amendments or additions to the system [28]. Winter also proposes the decomposition of an interlocking system into a principle model and an interlocking model. Both the functional requirements in the principle model and the functions in the interlocking model depend on the station layout and corresponding field elements. Subsequently, for each layout a different model and new functions must be determined. Therefore the model is dependent on the physical hardware and a great amount of time would be required for deducing the models for each station layout. Also, the safety variables incorporated into the system are derived from the field elements in the layout. As a result all possible failures and safety-critical scenarios are not considered [6].

### *3.4.2. Software Engineering Methods*

#### 3.4.2.1 Formal Methods

The application of distributed formal methods in railway systems provides precise methods for guaranteeing the safe movement of trains. This is achieved by accurately specifying the interlocking rules of operation. These methods provide a list of checks and actions that must be achieved in order to establish a safe functional system. Formal verification is suggested as the most significant way to guarantee the safety of a system [15]. Winter suggests the integration of formal methods into the software during the design phase. Thereafter model checking is used to determine the accuracy of the software prior to development. This is achieved by specifying a formal model of the functional requirements of a physical layout and verifying this against a formal model of the interlocking functionality. The control table is then used to determine the interlocking model [6].

#### 3.4.2.2 Life Cycle Models

Agile methods used in software development allow software to be validated at an early stage while incorporating feedback and changes into following iterations. The iterative process detects errors early during development and avoids the risk of late system integration. However, these methods have less focus on detailed planning prior to design and implementation of the software. This results in vague requirements analysis and change management. Each iteration provides a functional release of software. Feedback is incorporated into each release and the software is validated once again. This emphasises customer collaboration and a faster development process. Whilst agile practices do not offer an effective way of dealing with documentation produced during each iteration, this problem can easily be avoided by only updating documents when necessary, i.e. when notable changes are made. Given that agile methods are not commonly used in safety-critical environments such as rail automation, most agile processes can easily be adapted to suit regulated software development. Subsequently, agile methods can reduce the high costs and time required to produce safe software to meet regulations in hazardous environments [29].

By following the phases in the waterfall life cycle model the quality of software is controlled during each phase. In addition the verification and validation phases ensure the system does not divert from specified requirements. However, in the waterfall model extensive time is spent on planning prior to implementation. Requirements are specified early and cannot be changed during the development process. The waterfall model does not incorporate change. The strict phases are sometimes hard to abide by thereby creating a lengthy process of development [30].

#### 3.4.2.3 Design Patterns

Applied design patterns alleviate the lack of experience and capabilities of human knowledge. In addition, defining a generic pattern which can be adapted for real-time applications reduces the complexity of systems. However, design patterns can introduce indeterminism at the design level. Also, the use of generic or multiple patterns may result in insufficient support by software development tools. Conversely, design patterns increase the reusability of software. The patterns are functional and encompass knowledge from skilled engineers and domain experts [31].

In the model-view-controller architectural pattern, field elements are created in the model layer and are only passed to the presentation layer when required. Incorrect inputs entered by a user or received from the GUI can be treated using return values and by throwing exceptions. Although, the presentation layer must always reflect the current state of the model, i.e. the GUI needs to check for

changes in the model each time an action is invoked [10].

#### 3.4.2.4 Modelling Techniques

Statecharts illustrate a distributed point of view, i.e. signalling elements (signals, tracks and points) are modelled as separate items that jointly represent the interlocking rules. In a distributed system, elements are independent of the software. Subsequently each element and its operation can be modelled as a state machine. However, this results in a large number of state machines that interact with each other. Statecharts allow hierarchical modelling which reduces the complexity of big systems. Charts also function in parallel which allows multiple activities to be illustrated simultaneously. Global variables used ensure events are sent throughout the system and all machines have accessibility to this information [15]. Object-oriented programming allows elements to be modelled as objects and methods are used to demonstrate object functions [10]. UML is similarly based on object-oriented analysis and design methods. Class diagrams and state machines form part of UML. The main disadvantage of class diagrams is that an increase in the number of objects substantially increases the number of classes. Even so, this can be reduced by identifying common functions and attributes between objects by using abstract classes [30].

APNs are flexible in modelling safety-critical systems and ensure forbidden states are not reached. This reduces the possibility of human errors. However, the graphical and mathematical features of APNs are complex and difficult to apply. In addition, APN models can grow substantially and become very complex. To reduce complexity only one direction of travel must be considered [16].

Although Fusion notation consists of well defined semantics and provides a solid framework for formal specification, the complexity and number of Fusion models increases substantially as the system develops. The mathematical expressions are easily interpreted but are difficult to deduce. The semantics in the defined configuration language are restrictive and complex to develop further as the graphical class models are designed at a very low level [28]. Similarly, although Z notation is a recommended method for achieving required confidence levels in safety-critical systems, it provides a very mathematical modelling approach. Also at the end of development, a detailed analysis must be determined from the abstraction in order to simplify and understand the model. This increases the time and complexity in developing the system [33].

The use of model checking of an interlocking as suggested by Johnston supports the validation of the system and does not depend on signalling expertise as the foundation of the test technique. Some steps of the model checking technique can be automated provided the system design is

always modelled in the same format and the requirements can easily be determined from this format. This is applicable to railway interlocking systems as the requirements are determined from a control table which is always deduced from a station layout. As the size of the system design increases, the efficiency and time taken to validate the system increases rapidly and may not produce a result at all. This problem can be mitigated by improving the run-time and memory usage of the process. Although the output of the model checking tool reveals real errors in the control table, it also produces unexpected and unusual behaviour for trains when the model is simplified for run time purposes. The simplification of the model may also result in the loss of credibility regarding safety standards. The model checking approach is very different in comparison to the software simulation method which actually evaluates realistic scenarios [17]. However, to avoid this additional complexity statechart models can be validated directly against the interlocking principles and through the development and simulation of the interlocking software. The software simulation in turn is based on the statechart models and tests the fundamental interlocking principles. The software also notifies the user when the system does not perform as expected [17].

The incorporation of test techniques in validating software simulations accurately assesses the performance of the system. Software simulations enable the imitation of actual environment conditions. Coverage-based techniques can be used to assess the complete performance of the system and the behaviour of the system under specified conditions. Fault-based testing can also be used to inject faults into the system and to identify possible limitations. By applying test techniques the software functions can both independently and collectively be tested through the use of unit and integration test stages. Unit testing permits bottom-up testing wherein low-level components are first tested. After these components have been integrated with higher level components the new subsystem is tested again. This ensures that all software components and functions are extensively tested [30].

### *3.5. Application of Selected Methods in Developing an Interlocking Solution*

Based on the literature analysis of existing interlocking methods, relevant methods have been selected. The application of these interlocking and software engineering methods in developing a generic interlocking solution is discussed.

From the above literature review it has been deduced that an electronic railway interlocking consists of a physical layout, control table, system models, interlocking software and validation techniques. The physical layout represents the geographical topology of the signalling elements on a railway

yard. Signalling elements refer to signals, tracks and points which assist in controlling the movement of trains. Based on railway interlocking principles a simple physical layout will be designed. Control table conditions will be deduced from the physical layout. Models of the system will be designed to demonstrate the operation and communication between field equipment and the interlocking software. System components and functions will be modelled using statecharts.

Boolean notation will be used to develop interlocking functions. These functions will represent the interlocking principles and define the behaviour of the interlocking. Additional Boolean logic equations will be derived to test the software. Simulation of the interlocking software is the preferred method of validating the software as common realistic railway failures and scenarios can be simulated. The interlocking software will be simulated using Siemens SIMATIC TIA (Totally Integrated Automation) Portal software. The software will be validated by introducing faults into the system to determine if the interlocking responds in a fail-safe manner. The simulated software will be validated using the software engineering test techniques defined in Van Vliet [30]. Coverage-based and fault-based test techniques will be used to assess the interlocking functions and the complete interlocking software [30].

Formal methods will be used to validate the system and to ensure that the interlocking software meets regulated safety requirements. Agile methods will be followed when developing the interlocking software. Agile practices offer an iterative development model and enable software to be validated during early stages of development. This is beneficial when developing safety-critical systems [29]. An MVC architectural pattern will be followed as it accurately decomposes the functionality of the interlocking system into three layers. The field elements will be created and monitored in the model layer. The interlocking will manage the elements in the controller layer and element attributes and the movement of the trains will graphically be displayed to the operator in the view layer.

Based on the proposition of a universal computer interlocking system by Lutovac [14], a generic interlocking solution will be developed for rail automation systems which can be applied in varying countries of application. This contributes to the notion of a standardised railway interlocking system. The system is expected to be independent of the physical yard layout and will encompass railway operation principles to ensure the system functions in a fail-safe manner.

### *3.6. Conclusion*

A railway electronic interlocking system consists of a physical yard layout, control table, system models, interlocking software and verification techniques. Formal methods are incorporated to guarantee the safe operation of the railway vehicles across railways. These methods specify the interlocking rules of operation. Software engineering methods are integrated to improve the development process. Agile life cycle practices are used to reduce time and costs in producing safe software in safety-critical environments. This is in comparison to the waterfall model which increases development time due to strict phases of development. The model-view-controller architectural pattern is used to separate and remove interdependency of elements. Many mathematical modelling techniques are used to illustrate the functions and relationships amongst signalling elements. UML modelling techniques are also used to represent an interlocking system. UML provides an object-oriented approach allowing elements to be modelled as objects in a distributed model. The interlocking model is verified through the application of software engineering test techniques such as coverage and fault-based test techniques.

The interlocking components are integrated to develop a generic interlocking solution for railways. The interlocking encompasses safety guidelines and railway rules of operation. This solution can easily be customised for varying applications as it would be independent of the hardware layout. The generic solution is expected to improve the development process of safe software in safety-critical environments such as rail automation.

# 4. **INTERLOCKING SOLUTION MODEL**

## *4.1. Introduction*

Based on the notion of a universal computing interlocking system suggested by Lutovac [14] and assisting findings from the literature investigation, a generic software interlocking model is proposed. The model is expected to encompass both railway principles and fail-safe logic to ensure the interlocking performs in a safe manner. These components are built into the system and forms part of the foundation upon which the interlocking software is developed.

Firstly the functional requirements and assumptions of the interlocking model are specified. Thereafter the application of relevant software engineering techniques to the interlocking model is described. These techniques include the application of an MVC (Model-View-Controller) design architecture pattern and the incorporation of agile lifecycle methods during development. The proposed software interlocking model consists of a physical layout, control table and interlocking software. A rudimentary physical layout has been designed. The procedure of deducing control table conditions from the corresponding layout is described. The interlocking software is decomposed into Boolean logic functions and statechart algorithms. These aspects form the foundation of the interlocking model upon which the software is developed. The methods used to develop and simulate the interlocking software in SIMATIC TIA Portal are provided.

## *4.2. Interlocking Model Specifications*

The functional requirements and specifications to be considered when developing the interlocking solution model are outlined. The software interlocking model is expected to accurately simulate interlocking functions during the movement of a railway vehicle on a basic physical station layout. The interlocking model must be independent of the signalling elements present in the layout and must be able to correctly determine the availability of signalling element types. The software is expected to respond in a safe manner in the event of element failures or safety-critical events. Interlocking functions must not be permitted if all functional and safety requirements have not been met. The interlocking software must adhere to railway signalling principles and conform to CENELEC safety standards for railways [16].

Regarding the functional performance of the software, certain interlocking attributes are assumed in order to simplify the design of the interlocking solution. These assumptions include that a train can only occupy one track section at a particular time along a route. The speed and length of a train are

not taken into consideration. For simplicity purposes, only one route function at a time is executed and evaluated. Only main route movements are considered, i.e. Shunt routes for moving wagons into a goods yard, loading wagons, etc. is not considered. The final assumption is that train drivers behave well and trains do not speed. Subsequently collisions and hazardous events relating to speeding trains are not considered.

The interlocking software follows the three-Aspect railway signalling system, i.e. Red (Stop), Yellow (Caution) and Green (Proceed). The moving block interlocking system is incorporated into the software. This system defines that the area occupied by a train and the distance in front of it move together along a route.

### *4.3. Software Engineering Methods*

Software engineering methods have been incorporated to follow an efficient software development process. The application of an MVC design pattern and the integration of agile methods in developing the solution are described.

#### *4.3.1. Model-View Controller Pattern*

The interlocking solution is designed as an interactive system wherein computational elements are separated from elements which handle input/output data. Subsequently, an MVC architecture pattern has been followed which decomposes the system into three layers – Model, View and Controller. The MVC architecture is illustrated in Figure 4.1.

The system model is defined in the Model layer and does not communicate with the other layers. The Model layer stores the data required by the interlocking such as the current state of elements in the station layout. Logic operations such as the execution of a route request and setting a route are also performed in this layer. The GUI forms part of the View layer which allows users to interact with the software. The GUI displays the current state of the system and enables users to set and control interlocking operations. The interlocking operations are configured using screens, switches and button operations provided in TIA Portal. The GUI also provides feedback from communication between a user and the model. The View layer displays the information stored in the model. The final layer is the Controller layer which processes and responds to system events. The Controller layer has permission to implement changes in both the Model and View layers. The Controller layer handles input actions received from the GUI. An example of this action is a request to change the availability of an element.

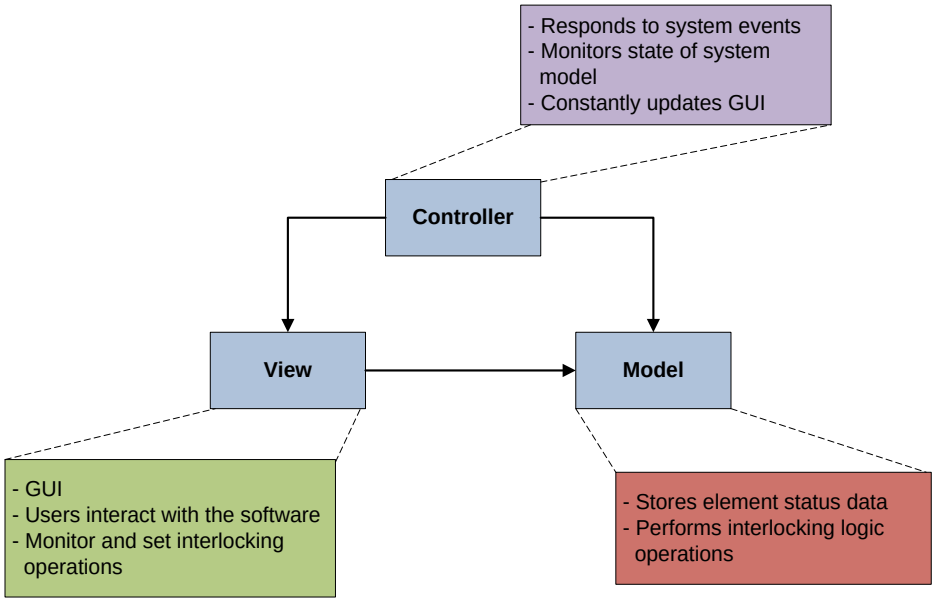


Figure 4.1: MVC software architecture

Due to the necessary communication between the View and Controller layers, a Presentation layer is defined which combines the two layers as shown in Figure 4.2. This combination ensures that the current state of the system is always displayed as the GUI now constantly checks for changes in the model after each action. The Presentation layer is now responsible for both displaying and updating the data stored in the model.

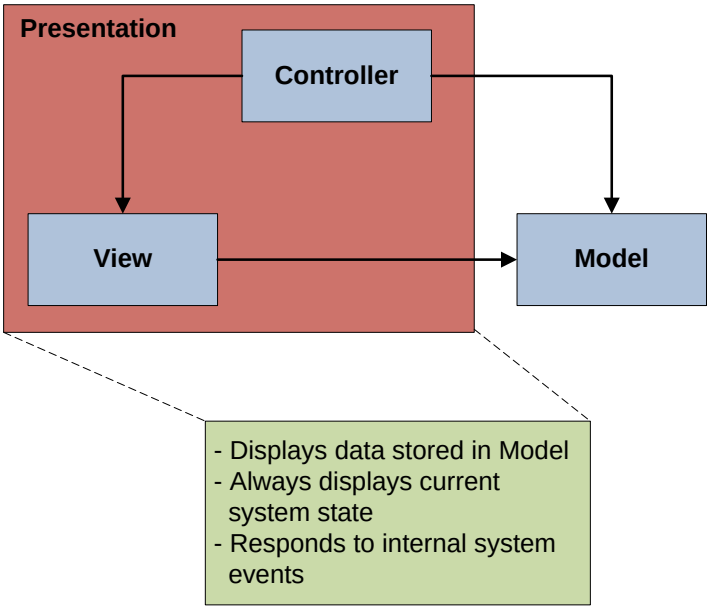


Figure 4.2: Modified MVC architecture

The MVC pattern with the adaptation of the Presentation layer provides a flexible software architecture. The architecture provides a base design upon which elements and system actions are easily dealt with. The Presentation layer ensures the GUI is constantly updated by regularly checking the state of the Model. This mitigates the problem of users not receiving feedback after configuring the interlocking. The MVC pattern provides freedom for future growth of the software solution. Even if the model size increases, the performance of the system is not inhibited.

### *4.3.2. Agile Lifecycle Methods*

An agile life cycle process has been followed during the development of the interlocking solution. Agile methods consist of small and incremental development cycles which are applicable when developing a software interlocking solution [30]. These methods allow for the incremental progression of software by first ensuring that the main interlocking functions are correctly developed first. Essential aspects of differing agile methods such as prototyping, incremental development and RAD (Rapid Application Development) are integrated to assist in developing the interlocking software.

Prototyping refers to the development of a working model of a software system with focus on certain aspects of the system. This focus allows for essential requirements to be determined [30]. For example, the software simulation requires a user interface in order to interact with the software. Prototyping methods are used to determine what vital information is required to be provided to the user to interact with the software. Incremental development methods provide a framework wherein software functionality is developed in small increments. These methods first focus on the essential features and avoid the development of excessive functionality [30]. This is beneficial when developing the interlocking software as key functions will be developed first with additional functionality built onto this foundation.

RAD involves aspects of prototyping, incremental development and the reuse of systems. RAD employs the concept of a time box wherein certain functionality must be achieved within a given time frame. If the functionality is not achieved it is removed [30]. This technique is adapted to the interlocking software solution in the form of iterations. The software system is hierarchically grouped into sections of functionality. Each group is referred to as an iteration and consists of a set of functions. These functions must be developed in the defined order to follow an incremental development model. However, in this model the time box is not applied. Also, if the set of functions in an iteration are not completed, the following iteration cannot be developed.

Following the fundamental aspects of agile methods such as incremental development, developing a working model and the grouping of system functions, the interlocking software has been decomposed into primary and secondary functions as described in Section 4.5.2. These functions are used as a guideline for dividing the software into iterations of development. Each development produces a functional release of software with key operations. The development cycle consists of – requirements specification, design, implementation and validation stages as shown in Figure 4.3.

The requirements specification phase identifies the main methods required to develop an interlocking system. Specific primary and secondary functions are categorised as requirement goals per iteration. These goals are further used to model the interlocking software during the design process. The outcome of an iteration further contributes to the development of the system model. During the implementation phase, the software functions are developed. The final stage is validation wherein the software is validated against the requirement goals.

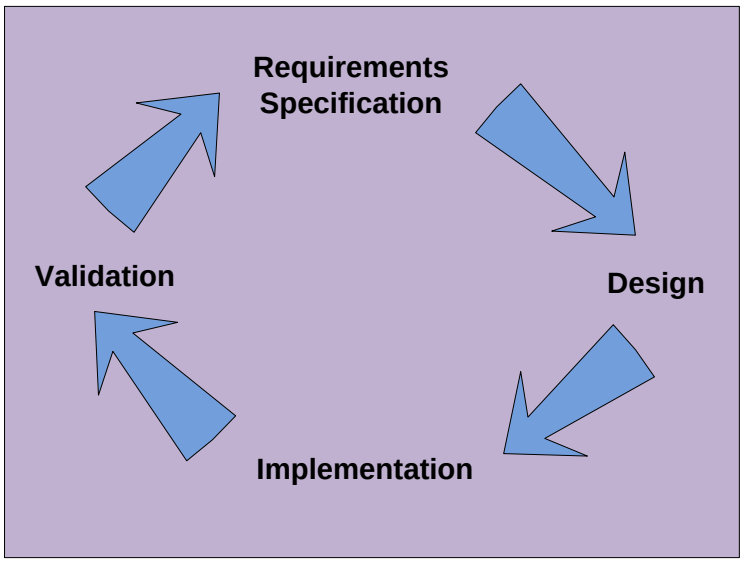


Figure 4.3: Agile life cycle model

Four iterations are defined. The primary and secondary functions to be developed in each iteration are illustrated in Tables 4.1 – 4.4. The Priority column denotes the level of functional importance of a function. A level 1 function has the highest priority. Priority functions are base functions upon which following functions may depend. The Function category column specifies whether a function is a primary or secondary method and the Function column specifies the method to be implemented. Primary functions are principal operations performed by the interlocking. Secondary functions are operations relating to the availability of signalling elements.

#### 4.3.2.1 Iteration 1

Firstly, methods to determine the availability of the signals, tracks and points are developed. Next, the physical layout is modelled to geographically demonstrate the layout of all the signalling elements. Thereafter the Route Request and Set Route functions are implemented for each route in the layout as indicated in Table 4.1. The main deliverables in this release include the development of methods to control and set the availability of signalling elements for each route in the control table, graphically depicting the availability of elements and the successfully execution of Route request and Set route functions for all routes in the layout.

Table 4.1: Iteration 1

| **ITERATION 1** | | |
|---|---|---|
| **Priority** | **Function Category** | **Function** |
| 1 | Secondary | Determine the current state of a signal |
| | Secondary | Determine the availability of a track section |
| | Secondary | Determine the availability of a point |
| 2 | | Physical Layout |
| 3 | Primary | Route Request and Set Route functions |

#### 4.3.2.2 Iteration 2

Iteration 2 solely consists of the Route call function as indicated in Table 4.2. This function allows a route to be called prior to a train travelling upon the route. A route can only be called after it has been set and reserved. The main deliverable in this software release is to call a route after it has been set and demonstrate the correct change in element indications. The Route call function must be developed for all routes listed in the control table.

Table 4.2: Iteration 2

| **ITERATION 2** | | |
|---|---|---|
| **Priority** | **Function Category** | **Function** |
| 1 | Primary | Route Call |

#### 4.3.2.3 Iteration 3

Iteration 3 consists of the Train occupation function as indicated in Table 4.3. This function can only be performed provided the route has been reserved (Iteration 1) and called (Iteration 2). The

main deliverable for iteration 3 is to accurately perform the train occupation function after a route has been called, as well as to ensure the route is cleared correctly after the train has completed the route.

Table 4.3: Iteration 3

| ITERATION 3 | | |
|---|---|---|
| **Priority** | **Function Category** | **Function** |
| 1 | Primary | Train Occupation |

#### 4.3.2.4 Iteration 4

The final iteration encompasses the fail-safe behaviour of the interlocking. In this iteration the behaviour of the interlocking during faults and safety-critical events is evaluated. This is performed by triggering element faults and simulating possible safety-critical events. A fault is assigned to each element in the physical layout. Safety-critical events are defined and can be triggered during the setting or calling of a route and during train occupation. Table 4.4 lists the required functions for iteration 4. The expected deliverable is that the interlocking responds in a fail-safe manner when an element fault is triggered, i.e. the corresponding route request is not executed and all elements revert to the default occupied state. In addition, during any safety-critical event, the interlocking must force the signal indications to red and ensure the points remain in their current position.

Table 4.4: Iteration 4

| ITERATION 4 | | |
|---|---|---|
| **Priority** | **Function Category** | **Function** |
| 1 | Primary | Element faults |
| 2 | Primary | Safety-critical events (Set Route) |
| 3 | Primary | Safety-critical events (Call Route) |
| 4 | Primary | Safety-critical events (Train Occupation) |

The above iterations illustrate the hierarchical grouping of interlocking functions through the categorisation of primary and secondary functions. The incremental development of these iterations contributes to the development of a complete functional software interlocking model. The software

model is then tested through the use of software simulations. These methods correspond to the key agile lifecycle characteristics discussed.

### *4.4. Software Interlocking Design*

In order to develop the interlocking model, a basic physical layout of a railway station has been designed. The layout encompasses all the fundamental signalling elements arranged into a simple configuration. The conditions which allow for movement of railway vehicles in the layout are stored in a control table.

#### *4.4.1. Physical Layout*

A basic physical layout has been designed using Microsoft Office Visio software. The layout illustrates the geographical arrangement of signalling field elements in a rail yard. Station layouts suggested by Moller [27], Kanso [7], Eris [2] and Mustafa [34] have been analysed and modified to develop a basic physical layout illustrated in Figure 4.4. The field elements in the layout consist of signals, track circuits and points. The elements are positioned according to railway interlocking principles and allow for bi-directional travel.

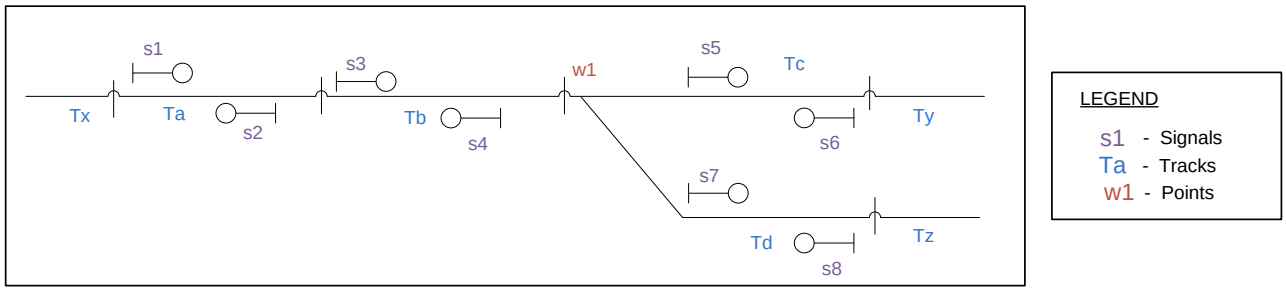


Figure 4.4: Physical station layout

#### *4.4.2. Control Table*

Conditions specified in control tables designed by Moller [27] and Lutovac [14] have been analysed to determine which conditions are of key importance. Based on the elements in the physical layout, a control table has been designed which defines:

- Route number (Number of each route in ascending order.)
- Signalling elements (All the signalling elements shown in the physical layout are listed in the table below. Signals are denoted by S, tracks by T and points by W. If an element is

required for a particular route, the element is assigned a binary value of 1. A 0 value indicates that the element is not required for the specified route.)

- Overlap (Track circuit required as an overlap for a route to protect a train in the event it cannot stop at the destination signal in time.)
- Direction (Direction in which a point must lie for a given route such that a train can travel safely across the point, i.e. normal or reverse.)

| Route no | s1 | s2 | s3 | s4 | s5 | s6 | s7 | s8 | Ta | Tb | Tc | Td | w1 | Overlap | Direction |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | Tc | N/A |
| R 2 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | Tx | N/A |
| R 3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | Ty | Normal |
| R 4 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | Ta | Normal |
| R 5 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | Tz | Reverse |
| R 6 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | Ta | Reverse |

Figure 4.5: Control table conditions

The conditions defined in the control table and the physical layout is used as input specifications for the interlocking software.

### *4.5. Interlocking Software*

The interlocking software is composed of: Interlocking functions in the form of Boolean equations, System models in the form of statecharts and Simulation of the interlocking software.

#### *4.5.1. Interlocking Functions*

The interlocking software is composed of interlocking functions. The input to these functions is extracted from the route conditions defined in the control table. The interlocking functions suggested by Lutovac [14], and the input-output state functions proposed by Eris [2] have been researched and analysed. The key components of these functions such as the use of Boolean notation, the state of elements and the definition of a general equation have been integrated to develop a general interlocking solution for railways.

The interlocking functions have primarily been developed as Boolean logic equations. A Boolean equation is defined for each route as listed in the last column of the control table illustrated in Figure 4.6. Only elements with a binary value of 1 are required for a designated route.

| Route no | s1 | s2 | s3 | s4 | s5 | s6 | s7 | s8 | Ta | Tb | Tc | Td | s1 | Overlap | Boolean Equation |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | Tc | R1 = s1 x s3 x Ta x Tb |
| R 2 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | Tx | R2 = s2 x s4 x Ta x Tb |
| R 3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | Ty | R3 = s3 x s5 x Tb x Tc x w1 |
| R 4 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | Ta | R4 = s4 x s6 x Tb x Tc x w1 |
| R 5 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | Tz | R5 = s3 x s7 x Tb x Td x w1 |
| R 6 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | Ta | R6 = s4 x s8 x Tb x Td x w1 |

Figure 4.6: Boolean interlocking functions

Based on the Boolean equations determined for each route, a general Boolean equation is defined for all routes in a given station, i.e.

$R_i = \Pi^{k}_{j=1} s_j \times \Pi^{l}_{j=a} T_j \times \Pi^{m}_{j=1} w_j$, $i=1 \ldots n$, where $n$ is the total number of routes

$\Pi^{k}_{j=1} s_j$ , where $k=i+2$ is the total number of signals required for route $i$.

$\Pi^{l}_{j=a} T_j$ , where $l$ is the total number of tracks required for route $i$.

$\Pi^{m}_{j=1} w_j$ , where $m$ is the total number of points required for route $i$.

An example is provided below which indicates how the Boolean equation for Route 4 can be determined from using the general Boolean equations.

Firstly, the conditions for Route 4 are: $i=4$, $l=c$, $m=1$ and $k=6$.

Subsequently,

$R_4 = \Pi^{6}_{j=1} s_j \times \Pi^{c}_{j=a} T_j \times \Pi^{1}_{j=1} w_j$

$R_4 = s_1 \times s_2 \times s_3 \times s_4 \times s_5 \times s_6 \times T_a \times T_b \times T_c \times w_1$

$R_4 = 0 \times 0 \times 0 \times 1 \times 0 \times 1 \times 0 \times 1 \times 1 \times 1$, Based on the values from the control table.

$R_4 = s_4 \times s_6 \times T_b \times T_c \times w_1$

This is the same result as the Boolean equation defined in the last column of Figure 4.6. Also note, that it has been observed that routes with even numbers require even numbered signals. Similarly for odd numbered routes, odd numbered signals are required.

### *4.5.2. Statecharts*

Statecharts are used to model the operation of the interlocking system. ArgoUML is a graphical modeling tool used to design the statecharts. The tool assists with the layout of the states and highlights transitions amongst them. Primary and secondary operations are modelled to demonstrate

the functional behavior of the system. Primary operations refer to the execution of a route request, calling of a route, occupation of a train, etc. Secondary operations consist of the sub-functions involved in primary operations, i.e. the procedure for determining the availability and current state of signalling elements. The complete set of statechart models has been included in Appendix B.

#### 4.5.2.1 Primary Functions

The interlocking functions are developed as generic algorithms designed to work on any railway layout. The following is a description of the statecharts developed for the primary operations:

- Route Request.

  A route request determines if a route can be set for a train to travel in a station. The request first determines the start and destination signals of a route. Thereafter all possible paths that connect these signals are identified. The required steps are illustrated in Figure 4.7.

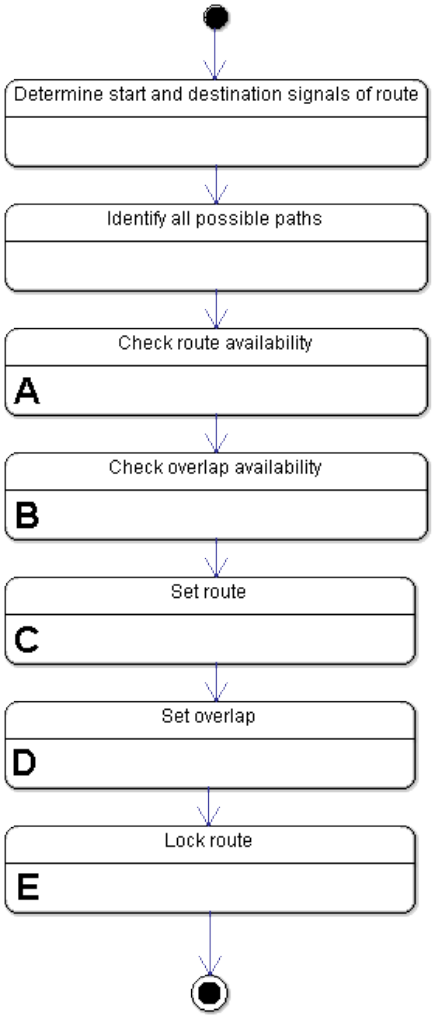


Figure 4.7: Route request statechart

The route request function is decomposed into the following sub-functions:

- Route Availability

  To check the availability of each path a series of checks must be performed as shown in the route request sub – diagram statechart in Figure B.2, Appendix B. Firstly the interlocking must ensure no conflicting routes to this request has been set. If there is a conflicting route, the route becomes unavailable and the following route must similarly be checked. Provided the route is available, the availability of the signalling elements along the route must be checked.

  First, the current state and availability of the signals are determined. Provided the signals are currently displaying a red aspect and not being used for another request, the checks may continue. Second, the required track sections are checked to determine their availability. Given that the track sections are unoccupied, the availability of the required points is assessed. If an element is available and will be used for the route a binary 1 value is sent to the interlocking for that element. This is applicable to all elements. If all the required elements are free and available, the route is considered available. However, in the event any of the elements are occupied and cannot be used the availability of another route/ path must be checked.

- Overlap Availability

  To determine the availability of an overlap track section, conflicting routes/ overlaps that have previously been set are first checked. If the required overlap is available, the next sub-function is executed, i.e. Set Route function. However, if the required overlap is not free the availability of another track section to be used as an overlap must be determined. The statechart depicting the Overlap Availability sub-function is illustrated in Figure B.2, Appendix B.

- Set Route

  In order to set a route, a series of steps must be followed. Firstly, the required start and destination signals of the route must be selected. Thereafter the instruction to set the route must be sent to the interlocking. The current state of occupancy of the track sections are checked again and then sequentially changed to a yellow indication from the destination signal backwards to the start signal. Next, the aspect of the start

signal clears to yellow. Yellow indicates that the state of the elements have been set and reserved. Upon setting the route, the option to cancel the route is still available. If a cancellation command is selected the sub-function returns to the second state in the statechart depicted in Figure B.2, Appendix B, i.e. prior to the command to set the route has been issued. If the route is not cancelled the system progresses to the next Set Overlap sub-function.

- Set Overlap

  This sub-function follows a similar procedure to the Set Route sub-function. The start and destination signal for overlap must first be selected. Thereafter the instruction to set the overlap must be issued to the interlocking. The overlap track section is checked to determine if it's free and is then set to a yellow indication as shown in Figure B.2, Appendix B. An option to cancel the overlap is provided and if selected will return the sub-function to the state prior to the set overlap command being issued. If the overlap is not cancelled the system continues to the Lock Route sub-function.

- Lock Route

  This sub-function follows a sequence of steps in order to lock the field elements as illustrated in Figure B.2, Appendix B. First the points are locked, thereafter the track sections and finally the entire route is locked. After the route has been locked, the route settings cannot be altered.

These sub-functions must be executed in the correct order to ensure the Route Request command is issued and the required route is set.

- Route Call.

After a Route Request is sent to the interlocking, the route maybe called. The Route Call process is shown in Figure B.3, Appendix B. A Route Call command must be issued from the start signal. After the command has been issued, the track sections sequentially turn green from the destination signal back towards the start signal. The green indication states that the required route has been called and cannot be used elsewhere. Thereafter, the overlap turns yellow which indicates that it has been reserved for the required route. Following these actions, the start signal clears to green which indicates that the route is ready for occupation.

The route call can still be cancelled at this point. In the event the route is cancelled the function reverts back to the first state before the Route Call command is issued.

- Train Occupation.

  This function describes the procedure as a train occupies each track section along a route. The procedure is illustrated in Figure B.4, Appendix B. Provided the start signal is displaying a green aspect, the route can be utilised. The train begins by occupying the first track section after the start signal. As the train occupies this track section, it turns red to indicate that it's occupied. The start signal then turns red which specifies that the following route is currently being used. The train continues along the route following the same procedure until it stops at the last track section before the destination signal. Provided the train stops at this signal, the reserved overlap, the route and required elements are cleared and set to available.

- Safety-critical Events.
  In the event of a safety-critical situation, the interlocking is expected to respond in a fail-safe manner, i.e. the state of the signaling elements is reset to a 0 value which indicates that they are occupied. This ensures the elements or route are not used for another route request. This procedure forces the signals to display a red aspect, the points to remain in the current position and the track sections are set to occupied as defined in Figure B.5, Appendix B.

#### 4.5.2.2 Secondary Functions

The secondary functions have been designed to determine the current state and availability of the signalling elements. The following is a list of statecharts designed for the secondary functions:

- Determine the current state of a signal.

  A signal can only be in either a red, yellow of green state. A statechart illustrating the possible states of a signal is illustrated in Figure B.6, Appendix B. Possible reasons for the signal indications are provided.

- Determine the availability of a track section.

  The current state of a track section is assessed to determine if it is free or occupied. If the track section is free, a logical 1 is sent to the interlocking. If track section is selected, it is then set to occupied and a logical 0 is sent to the interlocking. In the event the track section

is already occupied, the availability of another track must be determined. This procedure is modelled in Figure B.7, Appendix B.

- Determine the availability of a point.

  The current availability of a point is checked by determining if it's available or occupied. If the point is free a logical 1 is sent to the interlocking. Thereafter, the current position of the point is detected. If the point is currently in the correction position, it can automatically be used. If not, an instruction is sent to the interlocking to change it into the correct required position as illustrated in Figure B.8, Appendix B. If the point is currently occupied, a logical 0 is sent to the interlocking and the availability of another point is checked.

#### 4.5.2.3 Software Simulation Statecharts

Prior to the software simulation, statecharts for each route in the control table have been designed. These charts are derived from the general statecharts defined in the above functions. For each route, the generic requirements have been replaced by the actual required element. For example, in the Route Request statechart for Route 1, the availability of signal S1 is checked first. Thereafter the availability of elements Ta, S3, Tb and Tc are checked. Route request, Route call and Train occupation statecharts for all the routes are illustrated in Figures B.9 – B.14, Appendix B.

### *4.6. Interlocking Software Simulation*

The main focus in simulating the software is to test the design model and general statechart algorithms defined for the interlocking. SIMATIC TIA (Totally Integrated Automation) Portal has been used to simulate the interlocking software. TIA Portal is a software application which simulates the operational procedures in automation systems. The application combines the features of an engineering framework into the development of software projects [35]. TIA Portal does not generate any code and has been strictly used to model and simulate the proposed interlocking software solution. TIA Portal incorporates the use of tags for data communication. A tag is a memory location in the software which is updated by a change in state. The state value is centrally stored and can be reused throughout the software [36]. TIA Portal generates the software as a sequence of screens through which users interact with the application. A detailed description of TIA Portal, the features and advantages and the use of tag operations are provided in Appendix C. The screen structure of the software project has been included in Figure D.1, Appendix D.

### *4.6.1. Route List*

The physical layout and control table suggested above in Section 4.4 have been used to develop the interlocking software. Firstly, the control table was analysed to determine the number of routes and the elements required for each route. Based on this information, a screen titled Route List was developed. This screen lists all the valid routes within the layout and is shown in Figure D.2, Appendix D.

### *4.6.2. Route Element Screens*

Thereafter a screen has been designed for each route. For example, screen Route 1 (s1_s3) contains all the signalling elements required for the route, i.e. elements Ta, Tb, Tc, s1 and s3 as shown in Figure 4.8 below.

The availability of these elements can be configured on this screen through the use of a Switch object and tag operations. The Switch is connected to a tag which triggers a change in state for the associated element. The Switch is used to set the element state which is either free or occupied. A Boolean tag is associated with each element and can either consist of logical 1 or 0 values. A logical 1 indicates that the element is free and a 0 value states that the element is occupied. The Switch sets the logical value of the tag by changing between free and occupied states. The default state of all tags is set to 0. This is a fail-safe technique which assumes the element is occupied if not previously specified.

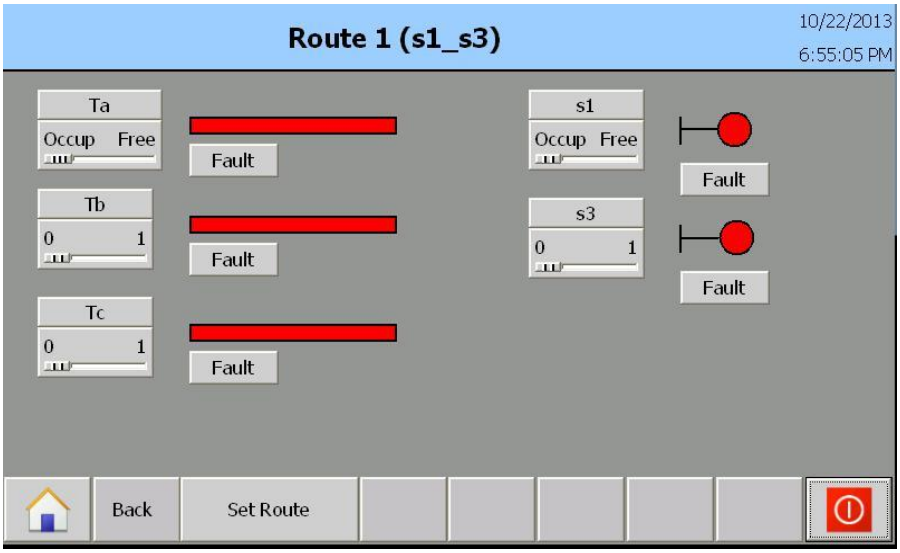


Figure 4.8: Route 1 (s1_s3) element screen

A fault button is provided next to each element to assess the element's behaviour in the event of a

fault. In addition, the behaviour of corresponding elements is also assessed when a fault is triggered. The fault button represents a group of faults that may occur for signals, tracks and points. Examples of possible element faults include: physical input/output peripheral faults, incorrect feedback current received from an element, signal filaments that have been blown, unable to detect current position of point set, set of points does not complete throwing, etc. [37].

The element fault button is connected to the same tag as the element. In the event of a fault, the state of the element is forced to an occupied state and subsequently the tag value is reset to 0. This also triggers a route configuration tag. In the event of an element fault, the route configuration tag value is set to 1 and a warning message is displayed on the screen to the user. This message notifies the user that the required route cannot be configured. Each screen has an option to Set Route which transitions the software to the Set Route screen described in Section 4.6.3. The above method has been applied to each screen, i.e. a screen has been created for each route in the control table. These screens are illustrated in Figures D.3 – D.8, Appendix D.

### *4.6.3. Set Route*

Upon selecting a route and defining the state of elements, the next step is to set the required route. The physical layout in Section 4.4.1 is replicated in this screen by using the graphical elements provided by TIA Portal. After the route request has been issued, the required route is set and all element indications change to yellow. This indicates that the route is reserved. The Set Route screen is illustrated below and in Figure D.9, Appendix D.

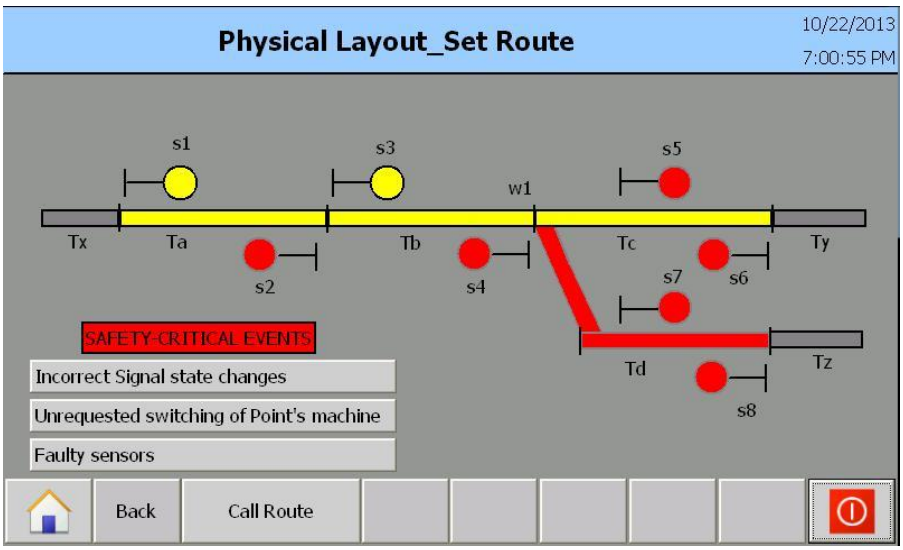


Figure 4.9: Set route screen

The Set route screen also consists of possible safety-critical events that may occur while the route is being set. Key events such as: incorrect signal state changes, unrequested switching of points and faulty sensors have been included. The purpose of these events is to evaluate the expected behaviour of the interlocking during a safety-critical event.

### *4.6.4. Call Route*

Following the setting of a route, the route must be called by the interlocking. Similar to the Set Route screen the physical layout has been replicated as shown in Figure D.10, Appendix D. When the call route command is issued the required elements change to a green indication. This indicates that the route is ready for train occupation. Possible safety-critical events have also been included as these may occur whilst a route is being called. These events are the same as those included in the Set Route screen.

### *4.6.5. Train Occupation*

Provided the route has been set and called, the train is allowed to proceed along the track. The train occupation procedure demonstrates the change in state after a train passes each element on a route, i.e. the element state is reset. For example, after a track section has been passed the track section changes from occupied (red) to free (grey) as shown in Figure D.11, Appendix D. Similarly, a signal aspect will transition from green (proceed) back to red (stop). The element state changes are sent back to the interlocking such that the element can be used for another route request if necessary.

The Train Occupation screen also provides a list of possible safety-critical events that may occur during the occupation process. Four fundamental events are included: train derailment, signal passed at danger, train detection operating incorrectly and maximum number of trains exceeded on a route. Additional safety-critical events are described in Section 2.5 Fail-Safe System. These events are included to assess the behaviour of the interlocking software in the event of a hazardous incident.

# 5. **TEST CASES & RESULTS**

The behaviour of the interlocking software developed in TIA Portal has been verified through the use of test cases. A set of test cases has been designed based on coverage-based and fault-based test techniques. Coverage-based test techniques evaluate the overall completeness of the software [30]. This technique is applied by evaluating whether the key interlocking function developed in each iteration is accurately executed in the software. The accuracy of the function is determined by comparing the expected and actual outcomes. Fault-based test techniques are incorporated into the software testing by injecting faults into the software and assessing the behaviour of the interlocking in response to these faults [30].

Test cases are designed to determine if the performance of the interlocking functions and functions to determine the availability of signalling elements executes correctly as illustrated in the UML statechart models. The adherence of the interlocking system to signalling principles and safety standards will be verified through test simulations. Safety-critical scenarios are simulated to evaluate the performance of the interlocking in hazardous situations. Test settings are configured to identify limitations of the system and possible aspects that have not been considered.

Seven test cases have been designed based on the requirement goals defined for each iteration whilst developing the interlocking software. Each case is designed to evaluate a specified interlocking function. In order to evaluate the performance of the functions, each function is applied to each route listed in the control table, i.e. Route 1 – Route 6. The application of these functions to Route 3 is used as an example for the test cases defined below. Route 3 is one of the possible routes listed in the control table and includes all the signalling elements (i.e. signals, tracks and points). The signalling elements can either be in one of the following colour states: red (occupied), yellow (reserved), green (clear) and grey (free). The state of these elements is controlled in TIA Portal using binary logic, tag operations and a Switch object as explained in Section 4.6.2.

The seven test cases have been applied to all six routes listed in the control table. These results are tabulated and included in Appendix E. The Element Configurations column illustrates whether the elements are set to free (logical 1) or occupied (logical 0). Based on this configuration, a route function assigned to each case is selected. For example, case 1 evaluates the Set Route function. The Expected State column lists the expected element indications after this function has been executed. The Actual State column illustrates the actual element indication observed.

### *5.1. Case 1: Set a route*

This test case is designed to test the functions developed in iteration 1 to request and set a route. Firstly the signalling elements for Route 3 are configured in TIA Portal to define the availability of each element. The required elements – s3, s5, Tb, Tc, Ty and w1 – are set to free, i.e. a logical 1 is sent to the interlocking for each element. This element configuration is maintained for testing functions in Case 2 and Case 3. Thereafter, the "Set Route" button in the menu bar is selected. Based on this configuration, when the Set Route function is selected the element indications are expected to change from grey (free) to yellow (reserved) as shown in Figure 5.1.

Table 5.1: Route 3 – Set route test case

| Element Configurations | Expected State | Actual State |
|---|---|---|
| s3 = 1 (Free) | s3 = Yellow | s3 = Yellow |
| s5 = 1 (Free) | s5 = Yellow | s5 = Yellow |
| Tb = 1 (Free) | Tb = Yellow | Tb = Yellow |
| Tc = 1 (Free) | Tc = Yellow | Tc = Yellow |
| Ty = 1 (Free) | Ty = Yellow | Ty = Yellow |
| w1 = 1 (Free) | w1 = Yellow | w1 = Yellow |
| | Route 3 is set. | Route 3 is set. |

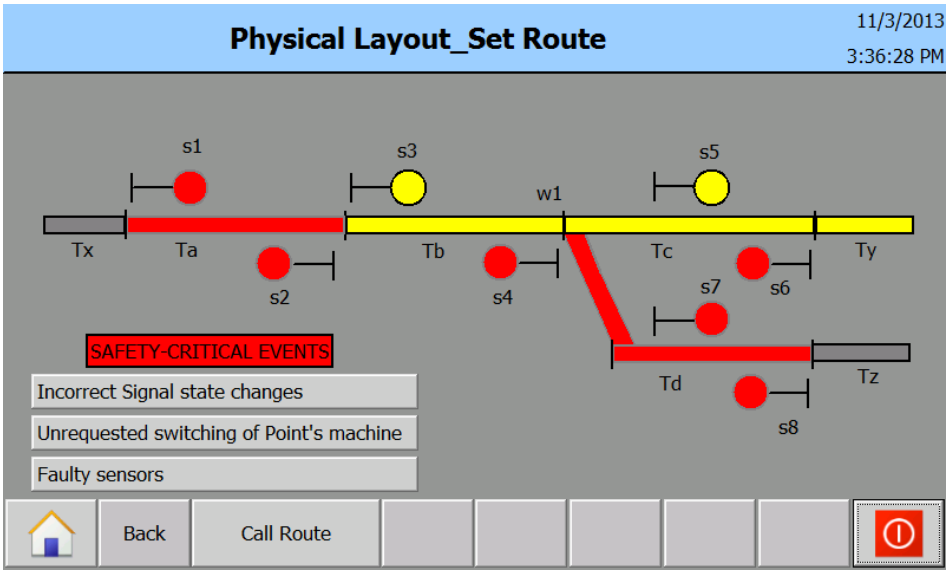


Figure 5.1: Route 3 - Set route function

*5.2. Case 2: Call a route*

Upon setting the route, the "Call Route" button in the menu bar must be selected next. This evaluates the call route function developed in iteration 2. After the function has been executed, the element indications are expected to change from yellow (reserved) to green (proceed) as shown in Table 5.2. This indicates that the route is ready for occupation by a train.

Table 5.2: Route 3 – Call Route test case parameters

| **Expected State** | **Actual State** |
|---|---|
| s3 = Green | s3 = Green |
| s5 = Green | s5 = Green |
| Tb = Green | Tb = Green |
| Tc = Green | Tc = Green |
| Ty = Green | Ty = Green |
| w1 = Green | w1 = Green |
| Route 3 is called. | Route 3 is called. |

*5.3. Case 3: Train Occupation*

Following the setting and calling of Route 3, the "Train Occupation" button in the menu bar must be selected. This test is designed to evaluate the occupation of a train on a route which is the main function implemented in iteration 3. After the train occupation function has been executed, the element indications are expected to change from green (proceed) back to grey (free) as illustrated in Table 5.3. This indicates that the train has successfully completed the route and the elements are now free and available for use by another route.

Table 5.3: Route 3 – Train Occupation test case parameters

| Expected State | Actual State |
|---|---|
| s3 = Grey | s3 = Grey |
| s5 = Grey | s5 = Grey |
| Tb = Grey | Tb = Grey |
| Tc = Grey | Tc = Grey |
| Ty = Grey | Ty = Grey |
| w1 = Grey | w1 = Grey |
| Route 3 is cleared. | Route 3 is cleared. |

The following cases (4 – 7) have been designed to assess the behaviour of the interlocking in the event of element faults and safety-critical events. In the event of element faults or safety-critical events, all the elements are forced to display a red indication which states that it is occupied (logical 0) and cannot be used.

*5.4. Case 4: Route cannot be configured*

Test case 4 is designed to evaluate the behaviour of the interlocking. For this test, the availability of some of the elements is set to occupied (logical 0). The interlocking is expected to not allow the route to be set as all the elements must be free in order to set a route. If the element is free (logical 1), the corresponding element indication should change to yellow. Conversely, if the element is occupied (logical 0) a red indication should be displayed. A combination of red and yellow element indications should not permit the setting of a route (i.e. the first interlocking function to be executed) as demonstrated in Table 5.4.

Table 5.4: Route configuration parameters

| . **Element Configurations** | **Expected State** | **Actual State** |
|---|---|---|
| s3 = 1 (Free) | s3 = Yellow | s3 = Yellow |
| s5 = 0 (Occupied) | s5 = Red | s5 = Red |
| Tb = 1 (Free) | Tb = Yellow | Tb = Yellow |
| Tc = 0 (Occupied) | Tc = Red | Tc = Red |
| Ty = 1 (Free) | Ty = Yellow | Ty = Yellow |
| w1 = 1 (Free) | w1 = Yellow | w1 = Yellow |
| | Route 3 cannot be set. | Route 3 is not set. |

*5.5. Case 5: Initiate an element fault*

This test case is developed to assess the behaviour of the signalling elements and the interlocking in the event of an element fault. The same procedure defined to set the availability of the elements for Route 3 in Case 1 above must be followed, i.e. the availability of all the elements must be set to free (logical 1). This can be seen in the first column under Element Configurations in Table 5.5. Thereafter an element fault must be triggered for element Tb by selecting the element fault button. The state of the elements after the fault has been triggered is shown in the Element State column. Any element fault should force all the elements to an occupied (logical 0) state and the corresponding route should not be set. The user must be notified of this action by an error message displayed as shown in Figure 5.2.

Table 5.5: Route configuration parameters

| Element Configurations | Element State | Expected State | Actual State |
|---|---|---|---|
| s3 = 1 (Free) | s3 = 0 (Occupied) | s3 = Red | s3 = Red |
| s5 = 1 (Free) | s5 = 0 (Occupied) | s5 = Red | s5 = Red |
| Tb = 1 (Free) | Tb = 0 (Occupied) | Tb = Red | Tb = Red |
| Tc = 1 (Free) | Tc = 0 (Occupied) | Tc = Red | Tc = Red |
| Ty = 1 (Free) | Ty = 0 (Occupied) | Ty = Red | Ty = Red |
| w1 = 1 (Free) | w1 = 0 (Occupied) | w1 = Red | w1 = Red |
| | | Route request is not issued. Route 3 cannot be set. | Cannot issue route request. Route 3 is not set. |

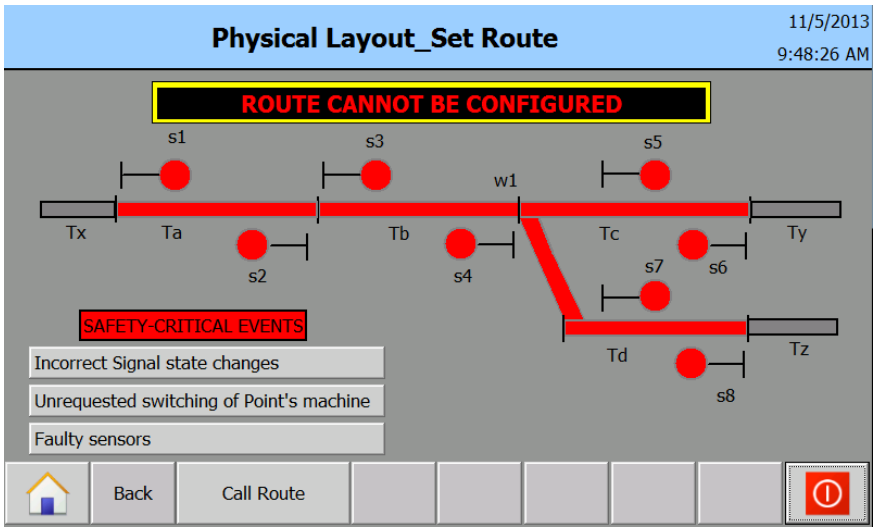


Figure 5.2: Unable to set route warning

*5.6. Case 6: Set a route and initiate a safety-critical event*

The response of the interlocking during a safety-critical event is examined. The same procedure for configuring and setting Route 3 defined in Case 1 must be followed. However, after selecting the "Set Route" function one of the safety-critical event buttons listed must be selected. During a safety-critical event, the signalling elements are expected to be forced to occupied and display a red indication as defined in Table 5.6.

Table 5.6: Set route – Safety-critical event

| Expected State | Actual State |
|---|---|
| s3 = Red | s3 = Red |
| s5 = Red | s5 = Red |
| Tb = Red | Tb = Red |
| Tc = Red | Tc = Red |
| Ty = Red | Ty = Red |
| w1 = Red | w1 = Red |
| Element indications must change to red. Route must be cancelled. | All elements turn red. Route 3 is cancelled. |

*5.7. Case 7: Call a route and initiate a safety-critical event*

Similar to the above case, this test case is designed to assess the response of the interlocking in a hazardous situation. The steps defined for configuring the elements and calling Route 3 in Case 2 must be followed. After the route has been called, one of the safety-critical event buttons listed must be selected. This is expected to force all the signalling element indications to red and the route can no longer be used as indicated in Table 5.7.

Table 5.7: Call route – Safety-critical event

| Expected Action | Actual Action |
|---|---|
| s3 = Red | s3 = Red |
| s5 = Red | s5 = Red |
| Tb = Red | Tb = Red |
| Tc = Red | Tc = Red |
| Ty = Red | Ty = Red |
| w1 = Red | w1 = Red |
| Element indications turn red. Route must be cancelled. | All elements in layout turn red. Route 3 is cancelled. |

### *5.8. Case 8: Train occupation with a safety-critical event initiated*

Test case 8 is intended to determine the performance of the interlocking when a safety-critical event occurs after the occupation of a train on a route. After the train occupation procedure described in Case 3 has been completed, one of the defined safety-critical event buttons must be selected. Similar to cases 6 and 7 above, the signalling elements are forced to occupied and to display red indications as illustrated in Table 5.8 below. The elements and route have not been freed and cannot be used elsewhere.

Table 5.8: Train occupation - Safety-critical event

| **Expected Action** | **Actual Action** |
|---|---|
| s3 = Red | s3 = Red |
| s5 = Red | s5 = Red |
| Tb = Red | Tb = Red |
| Tc = Red | Tc = Red |
| Ty = Red | Ty = Red |
| w1 = Red | w1 = Red |
| Element indications turn from grey to red. Elements are set to occupied. | All signalling elements turn red. |

The complete set of test cases verifies the key interlocking functions developed and validates the correct behaviour of the interlocking during faults and hazardous events. The functional requirements of the software are verified by following the simulations alongside the processes illustrated in the UML statecharts. This task verifies that the software has fulfilled the specified design requirements.

# 6. INTERPRETATION OF SIMULATION RESULTS

## *6.1. Analysis of Software Simulation Results*

The software simulation has been decomposed into seven test cases designed to verify key interlocking functions. The set of test cases has been applied to all six routes developed in the software. The main interlocking functions for requesting, setting and calling a route have been successfully evaluated by simulating these functions and comparing the output to the processes defined in the UML statecharts. The expected outcomes have been obtained.

A negative test case wherein the availability of certain signalling elements is set to unavailable has been executed. This test case checks if the interlocking responds in a correct manner. In this case, the interlocking did not allow the route to set which is the correct behaviour expected. This validates that the interlocking responds in a fail-safe manner by not permitting the route request due to the unavailability of all the required elements. The occupation of a train in a station is accurately simulated. The train moves in a sequential manner along the route as expected. Upon completion of the route, the route is cleared and the availability of the elements is set back to free as stipulated by interlocking principles.

Element faults have been introduced into the system to assess the fail-safe behaviour of the system. As expected in the event of an element fault for a requested route, all the elements required by the route are forced to red (i.e. occupied) and the route is not allowed to be set. This corresponds to the safety standards defined by CENELEC which states that the interlocking must respond in a safe manner to mitigate the level of risk and probability of injury to people [20]. The response of the interlocking system during safety-critical events has been comprehensively verified by simulating different types of safety-critical events. These events are triggered during the calling, setting and occupation of routes. The signalling elements perform as expected during a safety-critical event, i.e. signals and track elements are set to red (occupied) and points remain in the current position. This behaviour indicates that irrespective of the type of safety-critical event, the interlocking responds in a fail-safe manner by forcing all element states to red (i.e. occupied) such that the route and corresponding elements cannot be used for another request. This conforms to the functional safety specifications required by railway interlocking systems.

## *6.2. Generic Interlocking Solution Model*

A general software solution for developing an electronic interlocking system for railways has been

demonstrated. This solution encompasses fundamental interlocking functions and key components that form part of an interlocking system. The solution is independent of the physical layout of a railway. Standard Boolean functions are presented which can be determined for any station layout. The parameters for these functions are determined from a control table which in turn is derived from a station layout. Standard statechart models demonstrate the generic processes performed in an interlocking system. These are general processes which must be followed for the safe execution of interlocking operations. These are fundamental operations performed by all electronic interlocking systems irrespective of the railway system.

### *6.2.1. Flexibility*

The software components of the interlocking model are flexible and can easily be adapted to suit any station layout in any country of application. The main signalling elements (i.e. signals, tracks and points) are always used. These elements follow similar operational principles which can easily be adapted for additional field elements if required. These principles also ensure the safe behaviour of hardware elements under safety-critical conditions. The software components are based on standard railway interlocking principles and are independent of the physical hardware (i.e. signalling elements). The number of hardware components and the size of the system can easily be estimated from the specifications in the control table. This optimises the size of the interlocking system based on the application and mitigates the use of extra redundant hardware.

### *6.2.2. Expert Signalling Knowledge*

The generic solution encompasses fundamental interlocking principles and defined safety standards. This guarantees the safe behaviour of the system in the event of an element fault or during a safety-critical event. The general knowledge held by experienced signalling engineers is incorporated into the system. This permits the solution to be independent of expert signalling knowledge and limits the requirement of skilled signalling engineers to develop the software. Engineers are capable of developing the software and can simultaneously learn the railway interlocking principles through execution. However, adequate signalling knowledge is still required to interpret the station layout, control tables and to design the GUI. Although certain components of the interlocking system still require the knowledge of experienced signal engineers, the electronic system is fundamentally responsible for monitoring and controlling the movement of trains on a railway.

### *6.2.3. Quality*

The quality output of the system is substantially increased as the interlocking principles and safety

standards are built into the system. This ensures the interlocking system meets the required specifications. Formal methods also guarantee the safety of a system. Formal methods are used to define a set of criteria which is validated during the testing phase. Fail-safe behaviour has been incorporated into the system. This ensures that train control operators cannot request interlocking functions or alter the state of elements if the interlocking does not permit it. This certifies that the system has built in logic functions which protect the movement of trains on a railway even against human error.

### *6.3. Review of Applied Software Engineering Methods*

The application of software engineering techniques have greatly enhanced and supported the development of a generic interlocking solution. The application of formal methods in assuring the safety of an interlocking is explained. The benefit in integrating different agile method features and the incorporation of an MVC design pattern is examined. The use of software engineering test techniques in validating the software is discussed.

#### *6.3.1. Formal Methods*

Formal methods assist in guaranteeing that an interlocking system exhibits required fail-safe behaviour characteristics. These methods ensure that the software meets high quality standards especially in environments where faults cannot be tolerated.

Formal methods have extensively been incorporated into the software interlocking model by: identifying reusable functions in the system, developing a Boolean model, incorporating UML statechart models and the implementation of test cases. Key components and functions of a railway system are first identified. Similar functions are then grouped together. This technique enables the development of distinct objects with characteristic functions. These objects are then reusable throughout the system. For example, key signalling elements include signals, tracks and points. Each element is composed of a distinct group of functions. These functions are standardized and can be applied to any element of the same type, i.e. all signals (s1, s2, s3, etc) consist of three states. A mathematical model based on Boolean logic equations has been designed. These equations are derived from interlocking principles which control the operation of an interlocking system. Therefore, these equations are an interpretation of the interlocking rules which guarantee the safe establishment of routes for train movement. Statecharts are used to illustrate the sequence of checks and actions to be followed in an interlocking system. Statecharts for the main interlocking functions and for functions performed by signalling elements have been developed. This modelling technique

highlights the various subsystems of an interlocking system. Statecharts have also been used to verify the behaviour of the interlocking by validating the outcome of the software simulations. Test cases have been implemented to evaluate the performance of the interlocking. Expected outcomes are defined and the interlocking behaviour is assessed in the event of failure and related safety-critical situations.

### *6.3.2. Agile Methods and MVC Architecture*

The combination of agile life cycle methods and MVC architecture provides a maintainable foundation upon which the software has been developed. This structure enables the incremental development of interlocking software. Key interlocking functions have been identified, categorised and grouped into iterations of development. This modular structure increases the reliability of the system. The interlocking functions are easier to test and faults can easily be identified and traced. The cyclical development model enabled the interlocking software to be developed in stages. The main operations have been developed first and thereafter functions have been added onto the foundation. This development methodology increases the maintainability of the system. Improvements and amendments can still easily be incorporated into the system. Each iteration follows a quick and efficient development process of requirements specification, design, implementation and validation. The incorporation of design patterns defines a reusable architecture composed of a user interface, processing interface and communication links. This architecture supports the generic interlocking model and can easily be adapted for other real-time applications.

### *6.3.3. Software Engineering Test Techniques*

The use of a set of test cases based on applicable software engineering test techniques assists in validating the complete interlocking software. The test cases are designed to iteratively assess each interlocking function after it has been developed. Each test case is intended to test a key interlocking function. Provided the function is successfully executed and the expected output is observed, the test case is passed. The injection of faults into the software is incorporated into one of the test cases. The remaining test cases are designed to evaluate the behaviour of the software functions during a safety-critical event. The complete set of test cases collectively evaluates the overall performance of the interlocking software.

# 7. **CONCLUSION**

## *7.1. Introduction*

A conclusive analysis of the research investigation is outlined in terms of key outcomes in developing a generic interlocking solution model and the application of software engineering techniques during the development process. Significant conclusions on the software interlocking solution are presented. Recommendations for further work and additional enhancements to the software model are discussed.

## *7.2. Research Outcomes*

The research investigation has been structured into two fundamental components, i.e. a generic software interlocking solution for railways and the application of software engineering techniques in developing this solution. The findings of these research objectives are discussed accordingly.

### *7.2.1. Generic Interlocking System*

The various types of railway interlocking systems have been identified. These include: mechanical, electro mechanical, relay and electronic. Electronic railway interlocking systems are the most advanced systems and are still based on mechanical blocks and relay logic implemented in earlier systems. Electronic interlocking systems are controlled by microprocessors which electrically monitor and control signalling field equipment in a station layout. These systems are implemented using PLCs which assist in guaranteeing the fail-safe behaviour of a system. An electronic interlocking system also provides train control operators with visual feedback of train movement in a rail yard. This feedback is beneficial in ensuring the safe movement of trains and as a result many countries employ these systems. Electronic interlocking systems decrease the number of train collisions and rail accidents as the fail-safe logic and safety standards are programmed into the controllers. These systems are composed of both safety certified hardware and software.

Common aspects amongst varying electronic interlocking systems include: a physical layout, control table, design models, interlocking software and validation techniques. These fundamental components collectively represent an interlocking system. These aspects have been used to design a generic interlocking solution. A simple railway layout encompassing key signalling elements has been designed. From this layout, conditions for train movement are derived and recorded in a control table. The control table conditions for each route in the layout are interpreted as Boolean interlocking functions. These functions are generalised to establish a general Boolean equation for

determining which signalling elements are required for each route. Generic algorithms in the form of statecharts have been defined for key interlocking functions and can be applied to any route configuration. Standard methods have been designed to determine the availability of signalling element types and the execution of route functions such as the calling and setting of routes, etc. The integration of these components has collectively been modelled as a generic software interlocking model. A software simulation of the interlocking model has been executed. The integration of general functions and the results of the software simulation confirm that the model is independent of the physical layout of a railway. This satisfies a key specification defined for establishing a generic solution for railways.

### *7.2.2. Software Engineering Methods*

Some of the software techniques previously followed when developing an interlocking solution include formal methods and life cycle processes such as agile and waterfall models. UML modelling methods and MVC design patterns have also been applied to the architecture of the interlocking software.

Formal methods have extensively been incorporated into the development of the software interlocking system. Many variations of these methods are implemented to assist in ensuring the safety of the interlocking system. Formal methods have been incorporated into the system at the design level by identifying reusable components within the system, at the implementation level by developing a Boolean model and the formalisation of generic statechart models and at the validation level through the execution of test cases. Agile life cycle methods have been followed which ensures the software is validated during early stages of development. This is essential in safety-critical system as this guarantees that safe software is produced during the primary stages of development. Most agile processes can easily be adapted to suit regulated software development. The incorporation of incremental development and RAD methods enables the identification of key components with the main focus on core functionality being developed first. This is beneficial in critical railway systems as key safety features are incorporated into the core interlocking functions.

Following this method the software foundation has been formed and includes primary route functions such as the setting and calling of routes. Thereafter secondary functions such as determining the availability of signalling elements are built onto this foundation. The main functions that remain fixed form part of the foundation of a generic interlocking solution as these functions contain the fundamental operational and safety principles. This software model can then

be standardised and regulated by railway authorities as a base model upon which further railway software functions can be developed. UML statecharts have been used to develop standard system models which can be reused and applied to varying station layouts. The use of statecharts to depict the flow of processes enables the system to easily be maintained and updated if required. An MVC software design pattern has been followed to structure the interlocking software. This pattern decomposes the system functions into distinct layers. This permits changes and new features to easily be included into the interlocking system.

By applying software engineering methods in establishing a generic software solution which is independent of a station layout, development costs for electronic interlocking systems can greatly be reduced. In addition, this software model incorporates railway interlocking principles, signalling knowledge and safety standards into the design. This shortens the time required in developing a certified safe system. Due to the standardised structure and layout of the model, changes to the system can easily be incorporated. This modular structure allows additions to easily be integrated into the system design.

### *7.3. Conclusion*

Electronic interlocking systems guarantee the safe movement of trains by monitoring and controlling the physical signalling elements and interlocking operations. Mechanical, electromechanical, relay and electronic are the different types of interlocking systems currently in use. However, electronic interlocking systems are the most advanced as they incorporate the use of microprocessors and railway interlocking principles and safety standards are integrated into the system.

The fundamental components of an electronic interlocking system have been identified and incorporated into the development of a generic software interlocking solution for railways. The generic solution consists of a physical layout, control table, Boolean interlocking functions, statechart system models and the simulation of the interlocking software. The control table consists of the interlocking principles and specifies the conditions required by the interlocking. Thereafter a general interlocking function can be used to determine the required conditions for each route in a station layout. General algorithms modelled using UML statecharts can be followed to determine the process of interlocking functions in calling a route, setting a route, occupying a track section, etc. General algorithms are also defined to determine the availability of signalling elements for a

given route. The interlocking software simulation assesses the interlocking functions by evaluating the behaviour of the system in requesting and calling routes and in the event of system faults and safety-critical events.

The generic interlocking solution proved to be flexible as the system is independent of the physical station layout. The use of generic algorithms and functions allows the solution to easily be customised for any physical layout in most countries of application. However, in this customisation design additional consideration must be made to include country specific railway regulations. The solution also incorporates standardised interlocking principles required by all rail systems. The safety standards and fail-safe techniques are integrated into the system design which significantly reduces the probability of rail accidents. The use of software engineering methods such as formal methods, life cycle processes and architecture design patterns reduces the complexity in designing the software. These techniques enable the development of a modular system that is easily maintainable and reliable as all functional and safety requirements are incorporated into the system at various stages. The application of software engineering methods significantly assists in developing a sustainable generic solution for electronic railway systems.

### *7.4. Recommendations for future work*

Methods to adapt the generic interlocking software to specific station layouts or countries of application can be considered. In particular, consideration should be made for local railway application in South Africa. These customisation methods would define an efficient manner in which the general solution can easily be customised whilst still optimising the performance of the software system. Key components or parameters of the interlocking software can be identified that would be required to change for given applications. These variables can be configured into a design pattern which would serve as a template when customising any layout. To assess the efficiency and accuracy of the design pattern, example station layouts can be designed and these patterns can be applied to the layouts. Further investigation can be conducted to determine which software engineering techniques can be used to assist with the customisation methods. These techniques can further be assessed to determine if they would assist in maintaining the interlocking software and reducing the time and high costs in developing safety-critical software.

The local railway authorities and regulators such as the Railway Safety Regulator of South Africa must be investigated to determine if additional safety standards are required in addition to the

international CENELEC railway safety standards. The Railway Safety Regulator of South Africa is responsible for the creation of a safe railway environment wherein safe rail operations are performed in a guided regulatory framework. This is achieved through support, monitoring and enforcement of strict railway guidelines [40]. The examination of local railway safety regulations for specific countries of application must be included in the customisation model for adapting a generic interlocking software solution to specific station layouts and countries.

Methods to detect the non-availability of a route due to natural disasters such as landslides, earthquakes or other common disasters can be considered. The interlocking system can be linked to SANSN (South African National Seismological Network) for accurate information regarding the location and severity of a disaster [41]. This information can then be used to modify the behaviour of associated signalling elements within a given proximity of the disaster in a safe manner. A fault can be triggered on the element or the element can be forced into a fault state, i.e. in the case of a signal a red aspect can be displayed to warn the driver that he must not proceed past this signal. By triggering a fault on any of the elements within a given route, the interlocking automatically detects an alternate route upon which the train can proceed. This fail-safe feature has already been incorporated into the interlocking system and merely methods to accurately monitor the occurrence of natural disasters needs to be incorporated.

## REFERENCES


[1] Clark S. *A history of railway signalling*. Lloyd's Register Rail. London, UK, pp 1-15.

[2] Eris O, Mutlu I. *Design of signal control structures using formal methods for railway interlocking systems*. IEEE, December 2010, pp 1-2.

[3] Niekerk H V. RH Consulting. *Basic principles of signalling*. Siemens. Presentation Part 1, Reference number: 2045.102.001 REV 00, July 2012

[4] Kerr D, Rowbotham T. *Introduction to railway signalling*. Institute of Railway Signal Engineers. London, 2009.

[5] Mitchell I. *Managing obsolescence of electronic equipment in signalling*. IRSE International Technical Committee, November 2010.

[6] Winter K. *Model checking railway interlocking systems*. Software verification research centre. Australian Computer Society, Inc, Queensland, Australia, 2001.

[7] Kanso K, Moller F, Setzer A. *Automated verification of signalling principles in railway interlocking systems*. ScienceDirect, 2009, pp 1-5.

[8] Howker T. *Signalling Principles*. March 2011 – PRESENTATION, pp 1-30.

[9] Niekerk H V. RH Consulting. *Basic Principles of Signalling*. Siemens. Presentation Part 3, Reference number: 2045.102.001 REV 00, July 2012, pp 9-10.

[10] Eriksen L E, Perdersen N. *Simulation of relay interlocking systems*. Lyngby, June 2007, pp 3-42.

[11] Niekerk H V. RH Consulting. *Basic Principles of Signalling*. Siemens. Presentation Part 2, Reference number: 2045.102.001 REV 00, July 2012.

[12] Mirabadi A, Yazdi M B. *Automated generation and verification of railway interlocking control tables using FSM and NUSMLV*. Iran University of Science and Technology, School of Railway Engineering. Iran, Tehran, Narmak. 2009, pp 1-6.

[13] Woolford P. *Interlocking* Principles. Railway Group Standard, GK/RT0060, Issue 4, June 2003, pp 9-23.

[14] Lutovac D, Lutovac T. *Towards an universal computer interlocking system*. Facta Universitatis, Vol 11, No.1, 1998.

[15] Banci M, Fantechi A, Gnesi S. *The role of formal methods in developing a distributed railway interlocking system*. http://fmt.isti.cnr.it/WEBPAPER/FORMS_04.pdf, Pp 2.

[16] Yildirim U, Durmus M S, Soylemez M T. *Fail-safe signalization and interlocking design for a railway yard: An automation petri net approach*. Control Engineering Department,

Istanbul Technical University, September 2010, pp 1-2.

[17] Winter K, Johnston W. Robinson P. Strooper P. Van den Berg L. *Tool Support for checking railway interlocking designs. School of Information Technology and Electrical Engineering.* University of Queensland, Australian Computer Society, Inc, 2005, pp 4.

[18] Min Y,Zhou Y,Li Z, Ye C. *A fail-safe microprocessor-based system for interlocking on railways*. Annual Reliability and Maintainability Symposium Beijing, 1994, pp 1-3.

[19] Mewes K. *Domain-specific Modelling of Railway Control Systems with Integrated Verification and Validation.* Presented in Division 3 (Mathematics & Computer Science), University of Bremen. August 2009, pp 3-40.

[20] Verma M R,Chandra V. *The design and development of a fail-safe interlocking system using microprocessors for Indian railways*. IEEE, 1989, pp 1.

[21] Cenelec, Standards Development - Technical Bodies - EN50126-1:2012, http://www.cenelec.eu/dyn/www/f?p=104:110:2720947312294832::::FSP_PROJECT,FSP_LANG_ID:21752,25, Last accessed: 16 January 2014.

[22] Cenelec, Standards Development - Technical Bodies - EN50128:2011, http://www.cenelec.eu/dyn/www/f?p=104:110:2587428452934187::::FSP_PROJECT,FSP_LANG_ID:20508,25, Last accessed: 16 January 2014.

[23] Cenelec, Standards Development - Technical Bodies - EN50129:2003, http://www.cenelec.eu/dyn/www/f?p=104:110:3046470527007452::::FSP_PROJECT,FSP_LANG_ID:13550,25, Last accessed: 16 January 2014.

[24] Cenelec, Standards Development - Technical Bodies - EN61508-2:2010, http://www.cenelec.eu/dyn/www/f?p=104:110:3069298360366647::::FSP_PROJECT,FSP_LANG_ID:22088,25, Last accessed: 16 January 2014.

[25] Tombs D, Robinson N, Nikandros G. *Signalling control table generation and verification*. RTSA. Conference on Railway Engineering, Wollongong, November 2002

[26] Siemens, Mobility, Rail Automation, Electronic Interlockings, http://www.mobility.siemens.com/mobility/global/en/urban-mobility/rail-solutions/rail-automation/electronic-interlockings/pages/electronic-interlockings.aspx, Last accessed: 25 November 2013.

[27] Moller F, Kanso K, Setzer A. *Specifying railway interlocking systems*. Department of Computer Science, Swansea University, May 2010.

[28] Minkowitz C, Atkiss J. *An object oriented formal specification of a configuration language for interlockings*. 3rd Northern formal methods workshop, ALSTOM Signalling

Ltd, Manchester, United Kingdom, 1998.

[29] Jonsson H, Larsson S, Punnekkat S. *Agile practices in regulated railway software development*. IEEE, 2012.

[30] Van Vliet H V. *Software Engineering Principles and Practice*. John Wiley & Sons Ltd, England, Third edition, 2008, pp 50-450.

[31] Zalewski J. *Real-time software design patterns*. http://www.dsi.fceia.unr.edu.ar/downloads/informatica/info_III/realtimepatterns.pdf.

[32] Guo J, Huang Z, Liu M. *Research on the railway safety critical system with petri nets*. IEEE, 2006.

[33] Zafar N A, Khan S A, Araki K. *Towards the safety properties of moving block railway interlocking system*. International journal of innovative computing, Information and Control. Volume 8, Number 8, August 2012.

[34] Durmus M S, Yildirim U, Soylemez M T. *Signalization and interlocking design for a railway yard: A supervisory control approach by enabling arcs*. Symposium on Intelligent and Manufacturing Systems, September 2010, pp 1-5.

[35] Siemens, Industry Sector – Industry Automation Division – Press Release. http://www.siemens.com/press/pool/de/pressemitteilungen/2010/industry_automation/IIA2010112500e.pdf, Last accessed 7 October 2013.

[36] Siemens, SIMATIC TIA Portal Step 7 Basic Manual. http://stest1.etnetera.cz/ad/current/content/data_files/automatizacni_systemy/mikrosystemy/simatic_s71200/manualy/gsg_step7-basic-v10-5_2009-12_en.pdf, Last accessed 7 October 2013.

[37] Siemens. *Data engineering guideline for Siemens interlocking system.* Transportation Systems, Release 7.9.1, May 2007, pp 57-63.

[38] Nordland O. A critical look at the CENELEC railway application standards. SINTEF Telecom and Informatics. Trondheim, Norway. http://home.broadpark.no/~onordla/~SINTEF/tekster/A_critical_look_at_rail_standards.htm, Last accessed 21 October 2013.

[39] Siemens, TIA Portal Engineering Framework. http://www.industry.siemens.com/topics/global/en/tia-portal/tia-portal-framework/Pages/default.aspx, Last accessed 7 October 2013.

[40] Railway Safety Regulator, About Us. http://www.rsr.org.za/about-us/, Last accessed 6 August 2014.

[41] Council of Geoscience, South African National Seismological Network (SANSN). http://www.geoscience.org.za/index.php?option=com_content&view=article&id=1598:south-african-national-seismological-network&catid=138:seismology-unit-activities&Itemid=615, Last accessed 6 August 2014.

[42] National Instruments. Why LabVIEW. http://www.ni.com/labview/why/, Last accessed 6 August 2014.

## APPENDIX A: CENELEC STANDARDS

CENELEC (European Committee for Electrotechnical Standardization) have defined safety standards such as EN 50126 and EN 50129 for railway interlocking systems. These standards describe general safety requirements which must be followed [2]. Safety standard EN 50128 must be followed during the software development process which specifies that validation must be performed at each stage in the software development life cycle [19]. Railway interlocking systems are categorised as a SIL (Safety Integrity Level) 4 system based on the EN 50126 and IEC 61508 standards [12].

The following is a list of the key CENELEC standards for safety in railway applications:

- **EN 50126**: Railway applications – The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS) [21].

  EN 50126 provides baseline information on RAMS (Reliability, Availability, Maintainability and Safety), RAMS engineering and the connection between RAMS and Quality of Service. The standard defines a process for identifying key elements which influences the railway RAMS and explains the procedure to achieve RAMS requirements. Example of a RAMS specification, railways parameters, guidelines for following the RAMS process, etc is provided. The risk, fail-safe and safety integrity issues associated with railway systems are described [38].

- **EN 50128:** Railway applications – Communications, signalling and processing systems – Software for railway control and protection systems [22].

  This standard manages the software components of railway systems related to EN 50126 and EN 50129 [38]. The main principles include a top-down software development model, modular software, validation at each stage in the development process and adequate documentation [19]. EN 50128 identifies requirements for the life cycle process and necessary documentation (i.e. input and output documents) required for the project. The standard also describes software requirements for the complete software life cycle to produce software which meets railway safety standards. Requirements include: specification, design, implementation, testing, validation, maintenance, etc [38]. EN 50128 states that validation is performed by executing both functional tests which test the normal behaviour of the system and stress tests which tests the behaviour of the system in a safety-

critical situation. The different SIL levels are explained and stipulate that semi-formal methods such as state-machines can be used to specify and design the software [19]. The different SIL levels are explained and additional information on software techniques and methods required to be followed are provided [38].

The standard defines four critical levels where 0 is a non safety-critical level and 4 is the highest safety-critical level. These levels are defined after an assessment of the level of risk, probability of death or injuries of people, environmental disasters, damage or loss of property, etc. In railway environments, the failure of controllers can cause train derailment or collisions which can result in the loss of human life. Subsequently, the development of software systems for railways is assigned a safety-critical level of 4 [19].

- **EN 50129** – Railway applications – Communication, signalling and processing systems – Safety related electronic systems for signalling [23].

  EN 50126 and EN 50128 refer to the complete railway system whereas EN 50129 refers to a sector specific interpretation of the generic EN 61508 standard. En 50129 specifies conditions required for a safe railway system, subsystem or equipment. A Safety Case document is required wherein proof of adequate quality management, safety management, functional and technical safety is documented. The application and use of SIL levels are described in terms to THR (Tolerable Hazard Rates). Requirements regarding correct functional procedures, effects of element failures and other safety related events are provided. Techniques on identifying possible failure modes of hardware are defined and methods for avoiding and controlling systems faults are described [38].

- **EN 61508** – Functional safety of electrical/electronic/programmable electronic safety-related systems [24].

  EN 61508 provides general functional requirements for electrical/electronic/programmable electronic safety-related systems. EN 61508 defines standard software requirements for electrical/electronic/programmable electronic safety-related systems. These requirements must be fulfilled in order for the system to conform to the standard. Examples of determining the correct SIL levels for electronic systems are described [38].

## APPENDIX B: STATECHART MODELS

Siemens TIA (Totally Integrated Automation) Portal is a framework tool which integrates the engineering environment into a software project. TIA Portal assists developers in producing efficient software systems for control and automation applications [35].

National Instruments LabVIEW provides a visual programming environment which incorporates software engineering methods for the development of measurement or control applications. LabVIEW eliminates the need for physical hardware by simulating hardware functionality [42]. LabVIEW enables developers to use state machines as the software architecture upon which applications are built. These state machines are complex modules with strict parameters. A great deal of investigation and understanding is required to apply the statechart models illustrated in Appendix B to the existing state machines modules provided in LabVIEW. However, these state machine modules have proven limiting in behaviour for simulating the interlocking functions. Subsequently, Siemens TIA Portal is selected as the preferred tool to simulate the interlocking functions as it provides a user-friendly interface and simple tag-based functions. TIA Portal permits the reuse and does not limit the use of project components and the SCADA software components enable a simple graphical design and layout of complex components.

To simulate the interlocking functions in TIA Portal, the interlocking operations are decomposed into primary and secondary functions. Primary functions consist of fundamental operations performed by the interlocking and secondary functions refer to operations concerning the availability of signalling elements.

# 1. **PRIMARY FUNCTIONS**

## *1.1. Route Request*

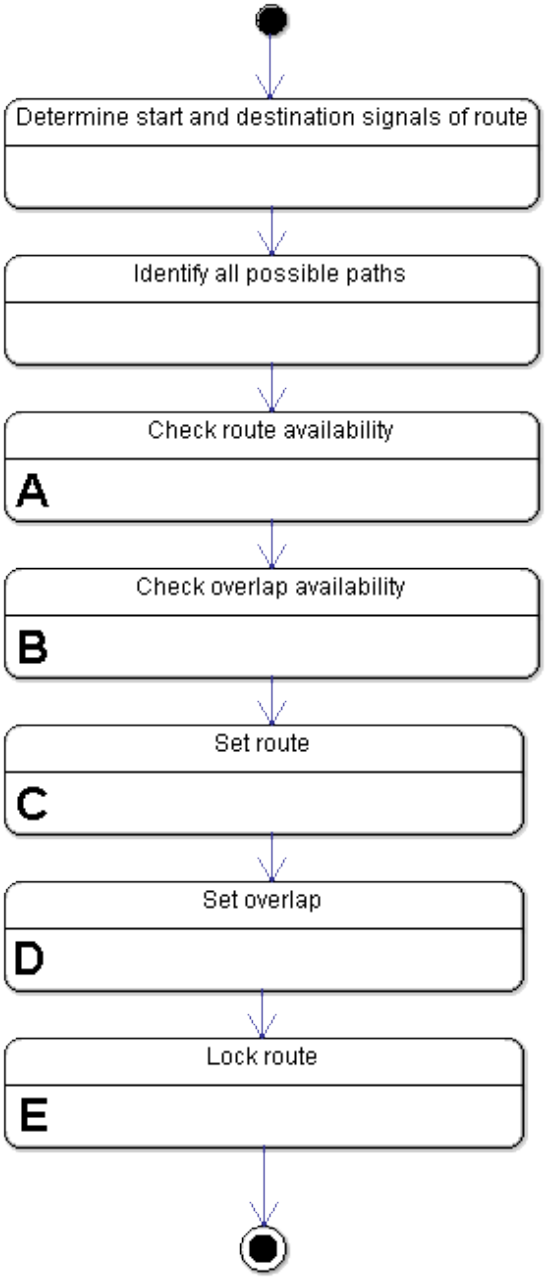


Figure B.1: Route request – main diagram

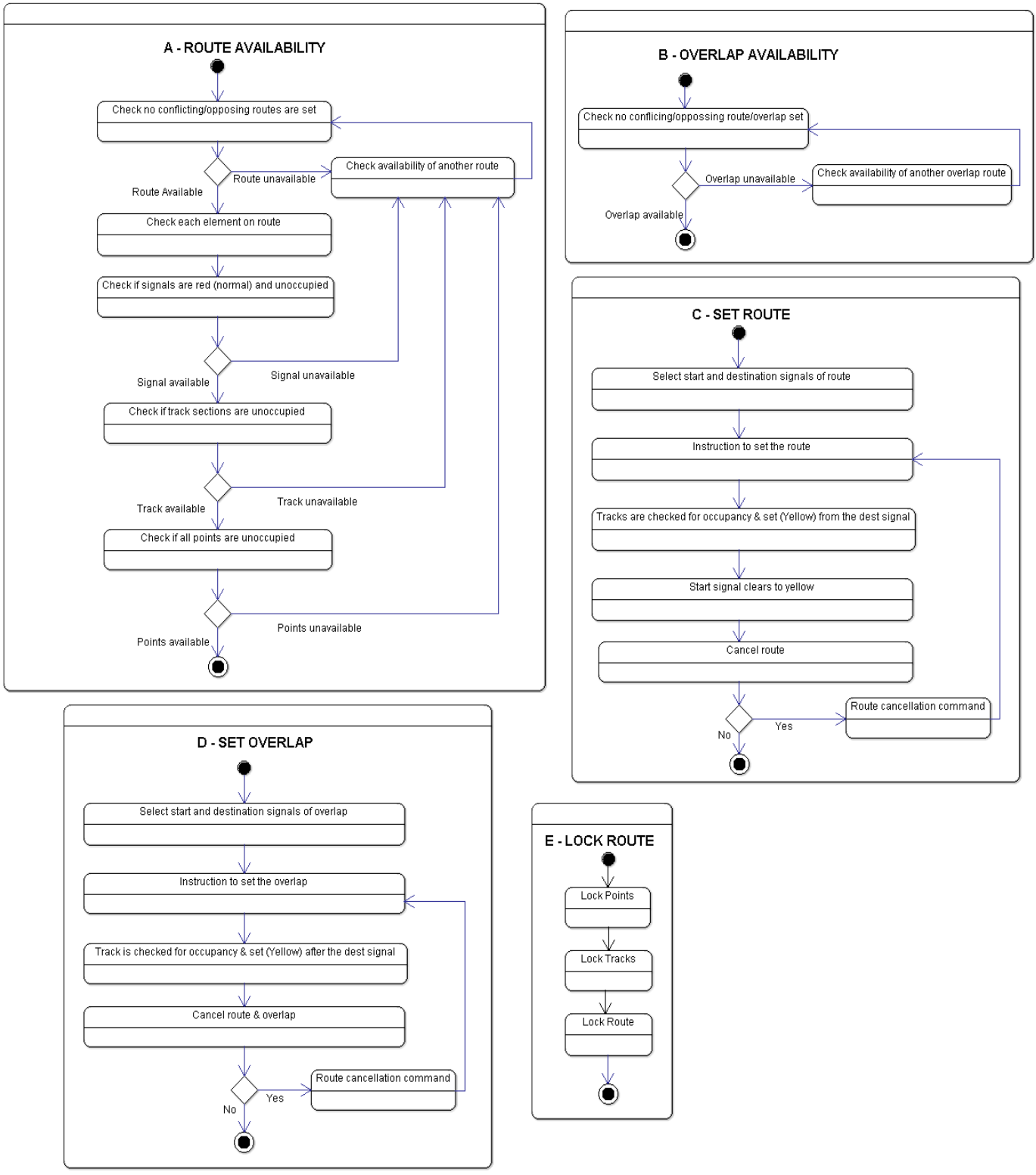


Figure B.2: Route request - sub-diagrams (Diagrams A–E)

*1.2. Route Call*

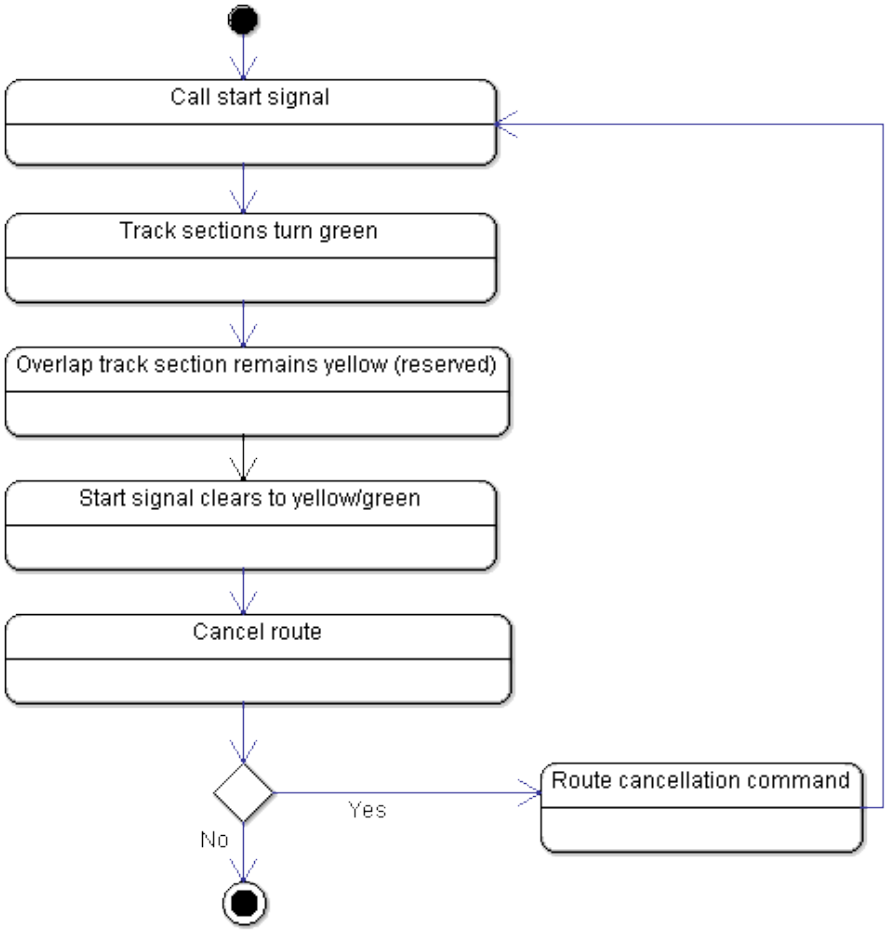


Figure B.3: Route call procedure

### *1.3. Train Occupation*

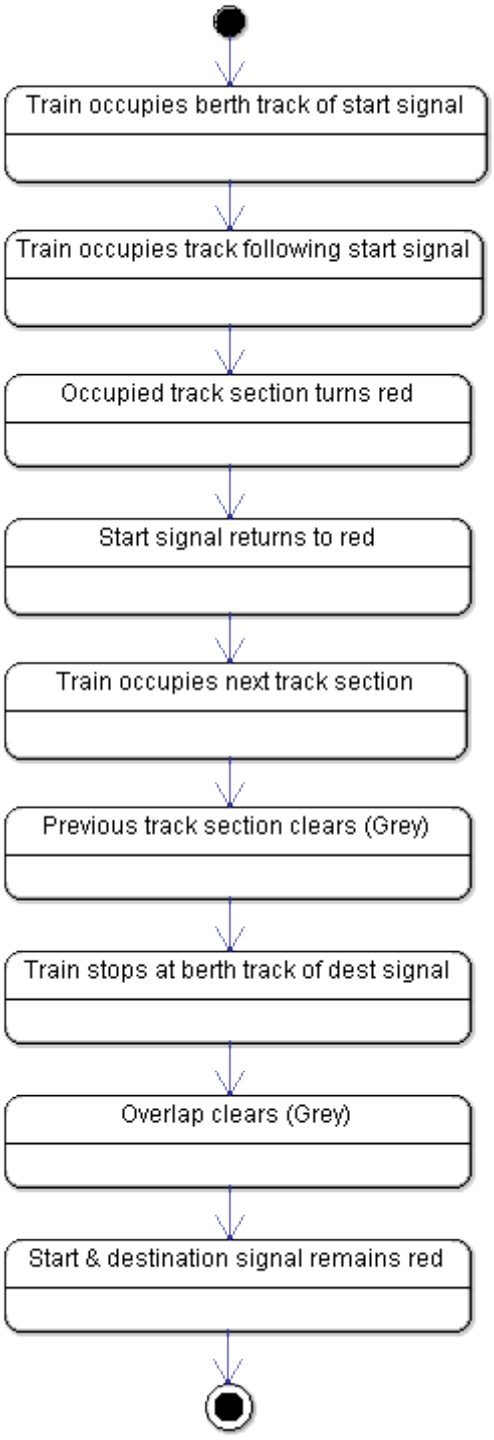


Figure B.4: Train occupation process

### *1.4. Safety-Critical Events*

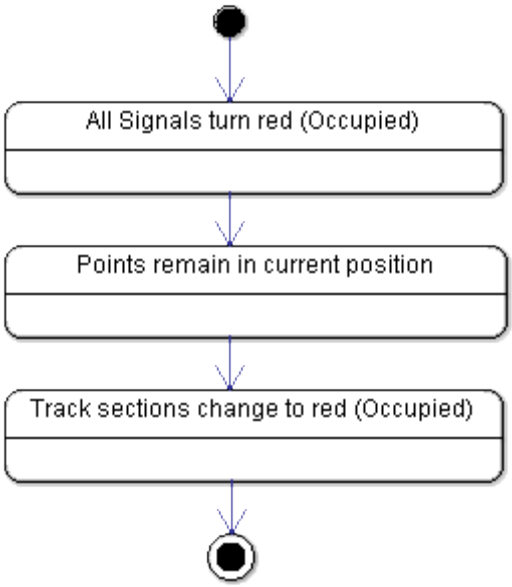


Figure B.5: Safety-critical event procedure

## 2. **SECONDARY FUNCTIONS**

### *2.1. Determine the current state of a signal*

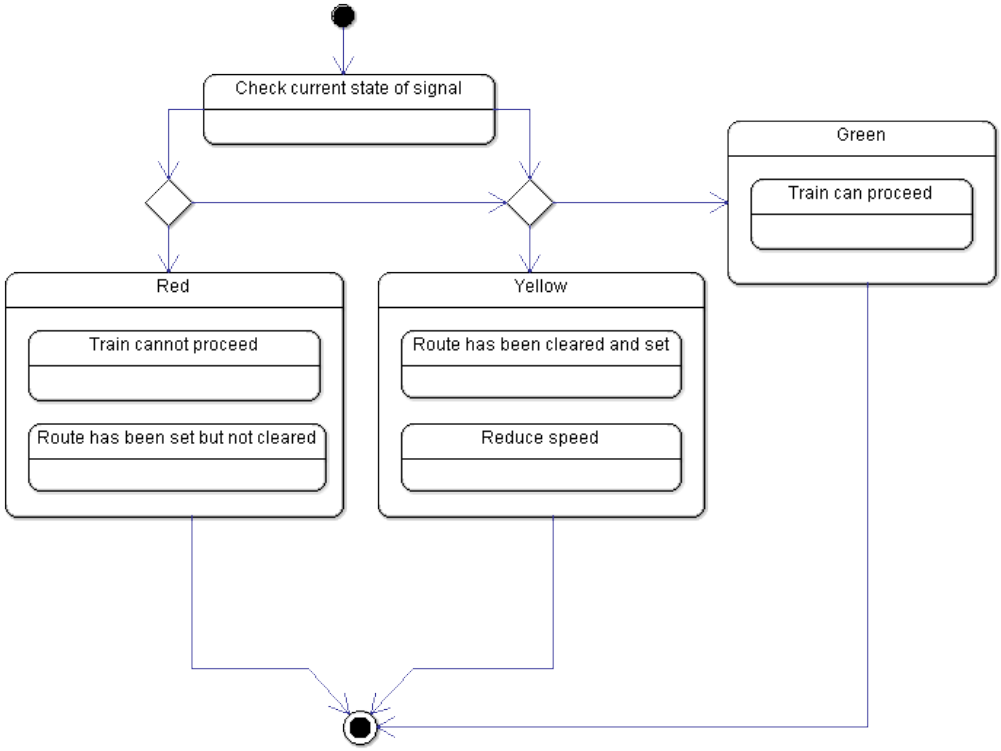


Figure B.6: Identifying current signal states

*2.2. Determine the availability of a track section*

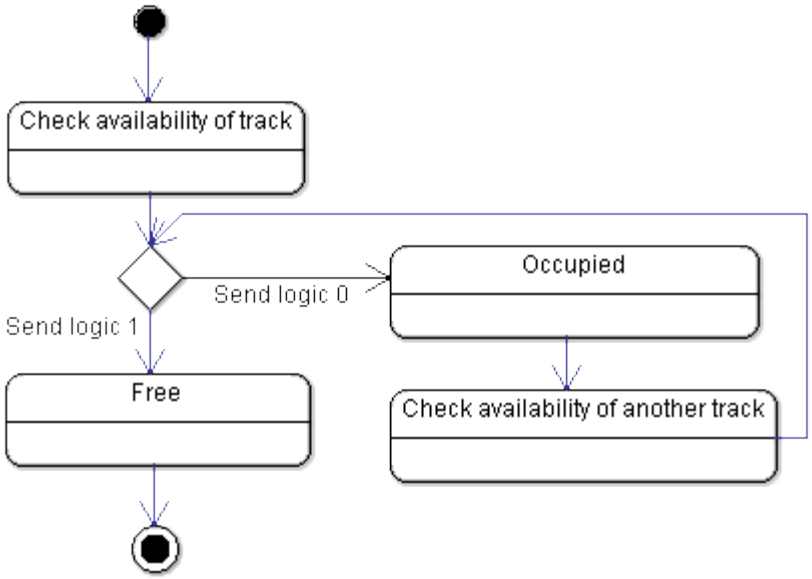


Figure B.7: Procedure to determine availability of track section

*2.3. Determine the availability of a point*

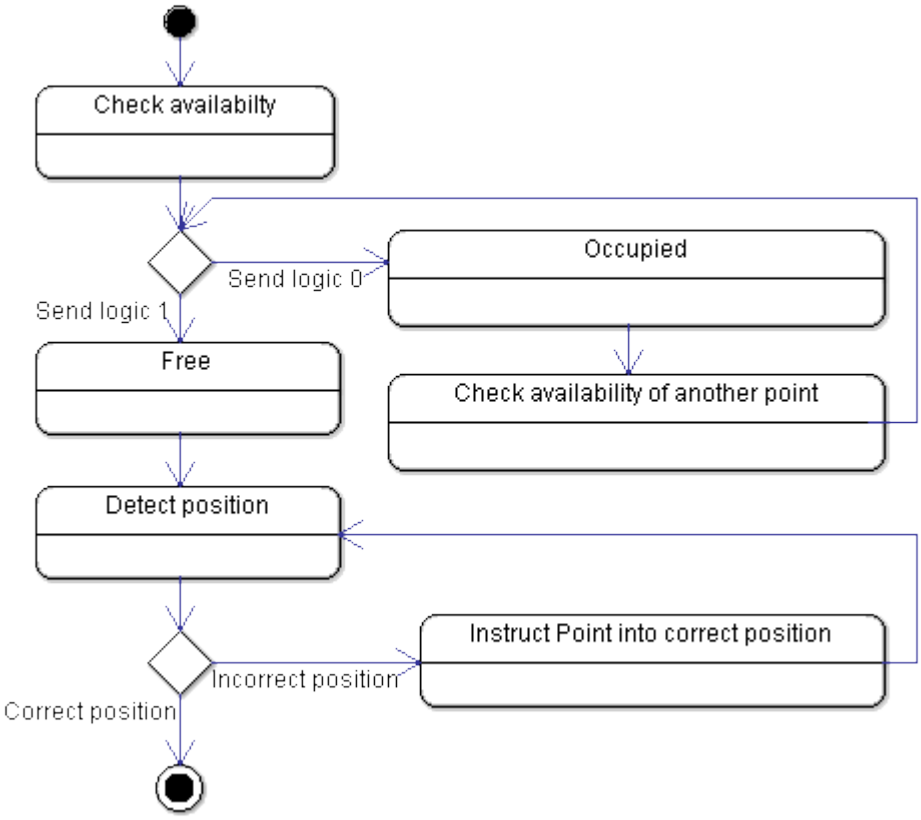


Figure B.8: Process to determine availability of point's machine

# 3. **SOFTWARE SIMULATION**

## *3.1. Route 1*

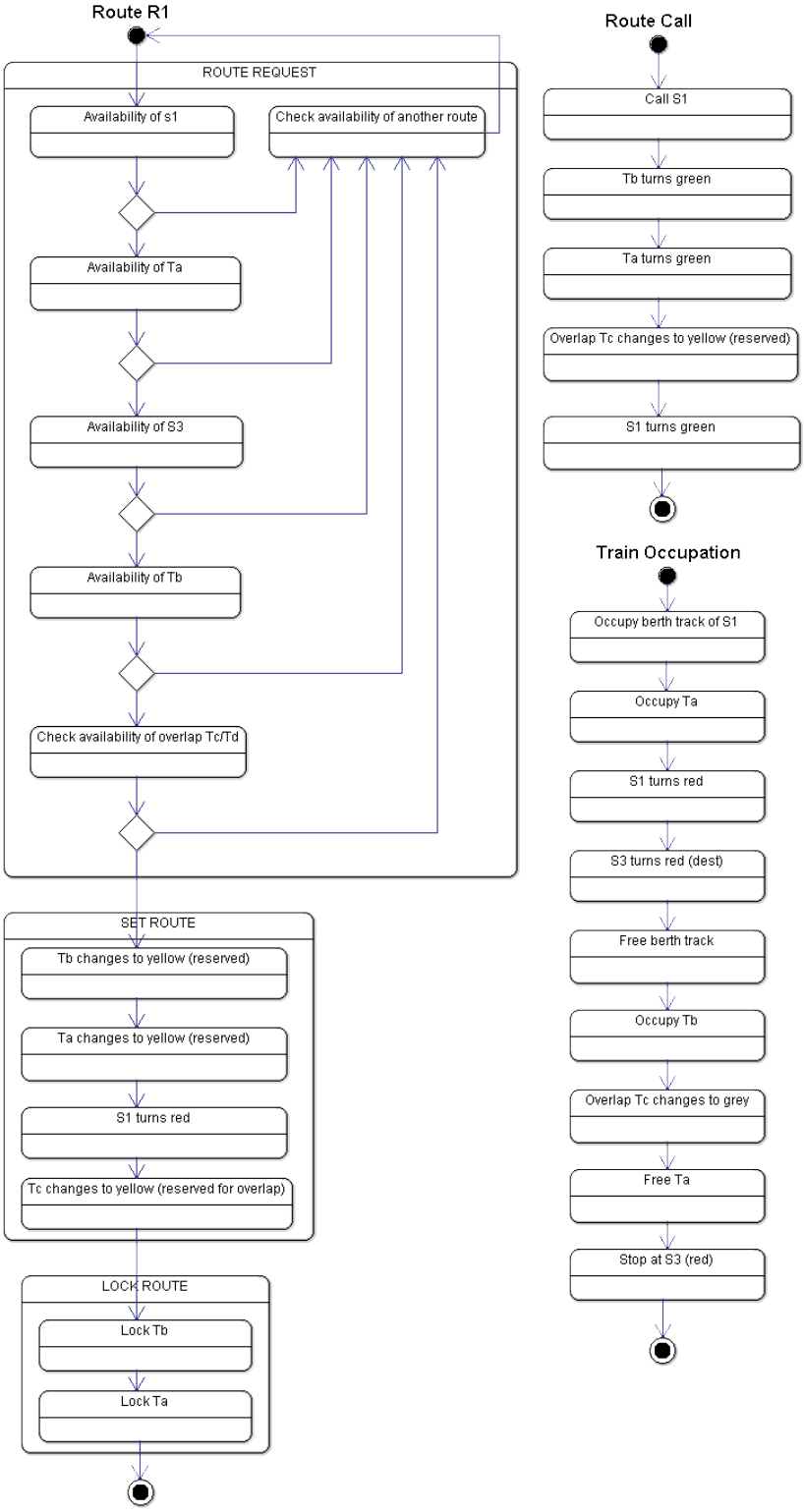


Figure B.9: Route 1 statechart diagram

### *3.2. Route 2*

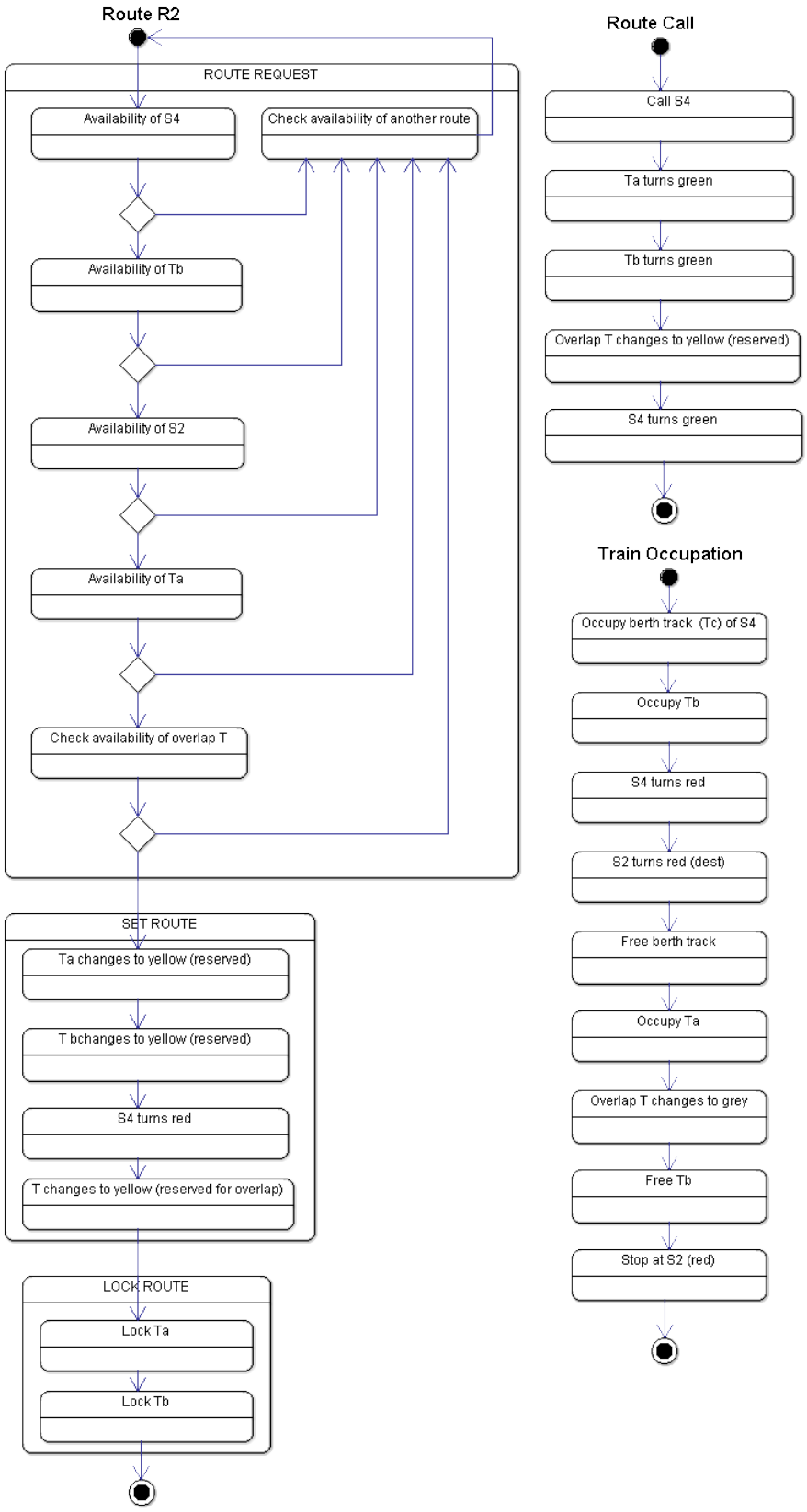


Figure B.10: Route 2 procedure

### *3.3. Route 3*

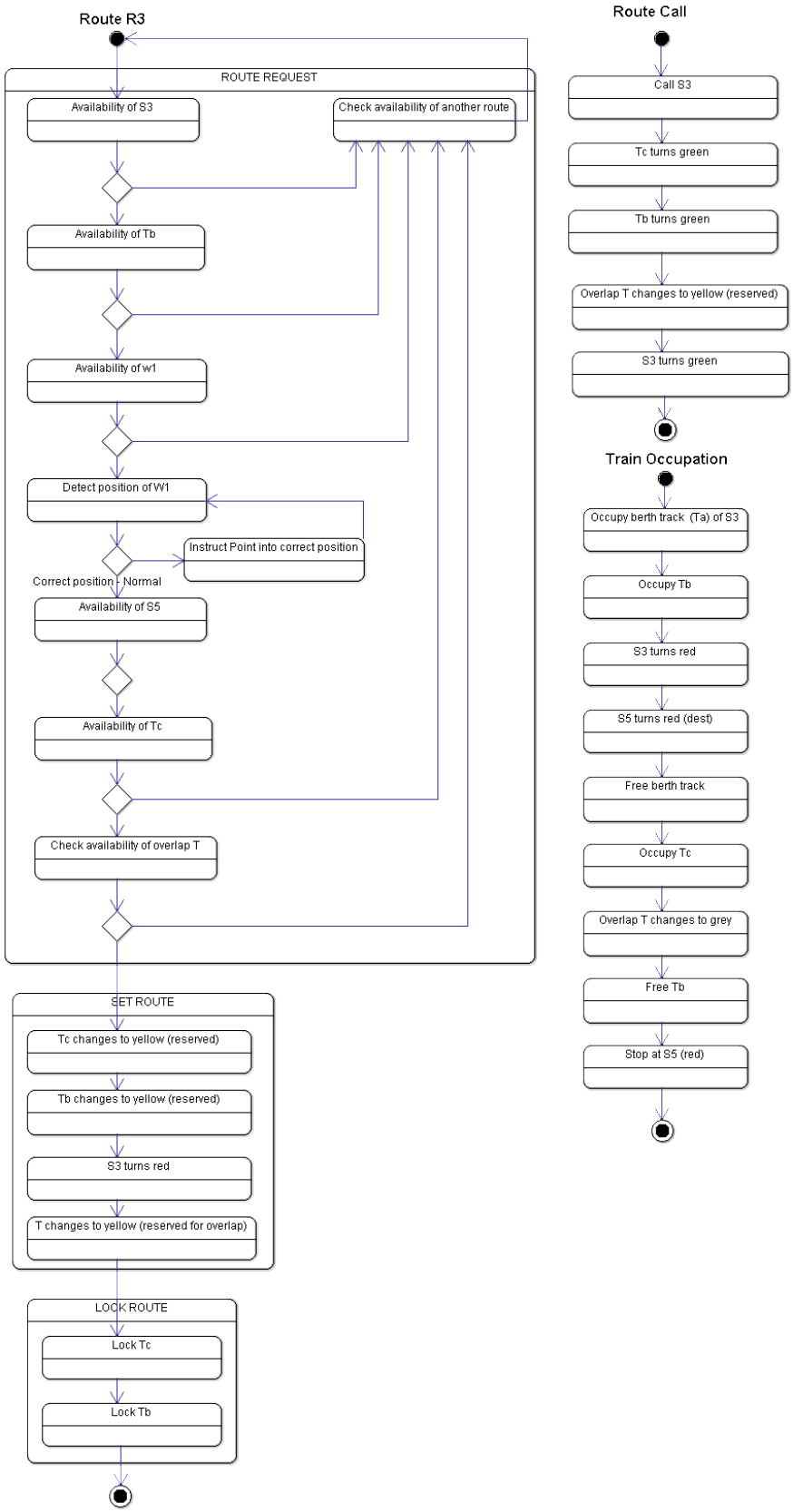


Figure B.11: Route 3 statechart

### *3.4. Route 4*

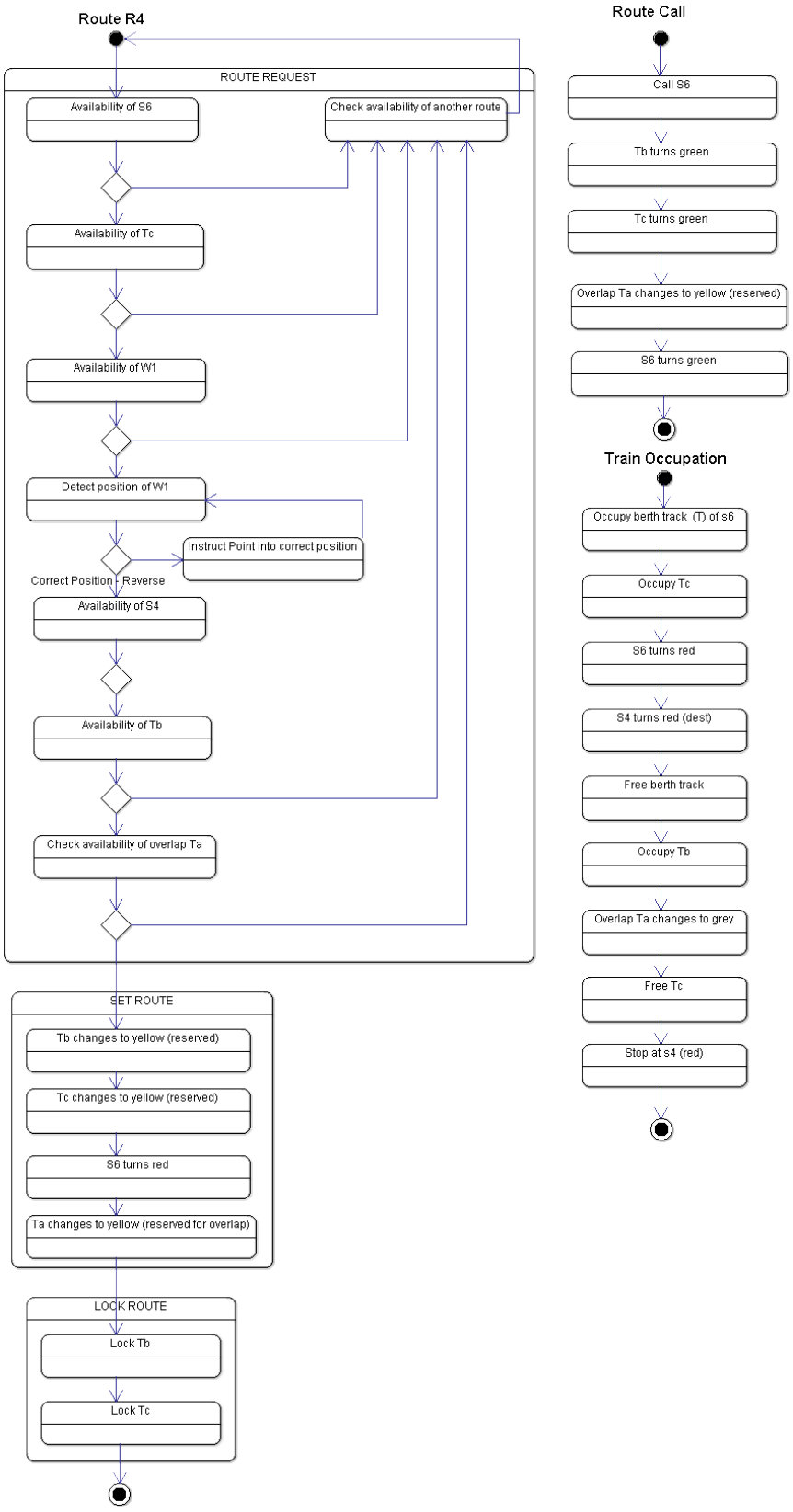


Figure B.12: Route 4 method

### *3.5. Route 5*

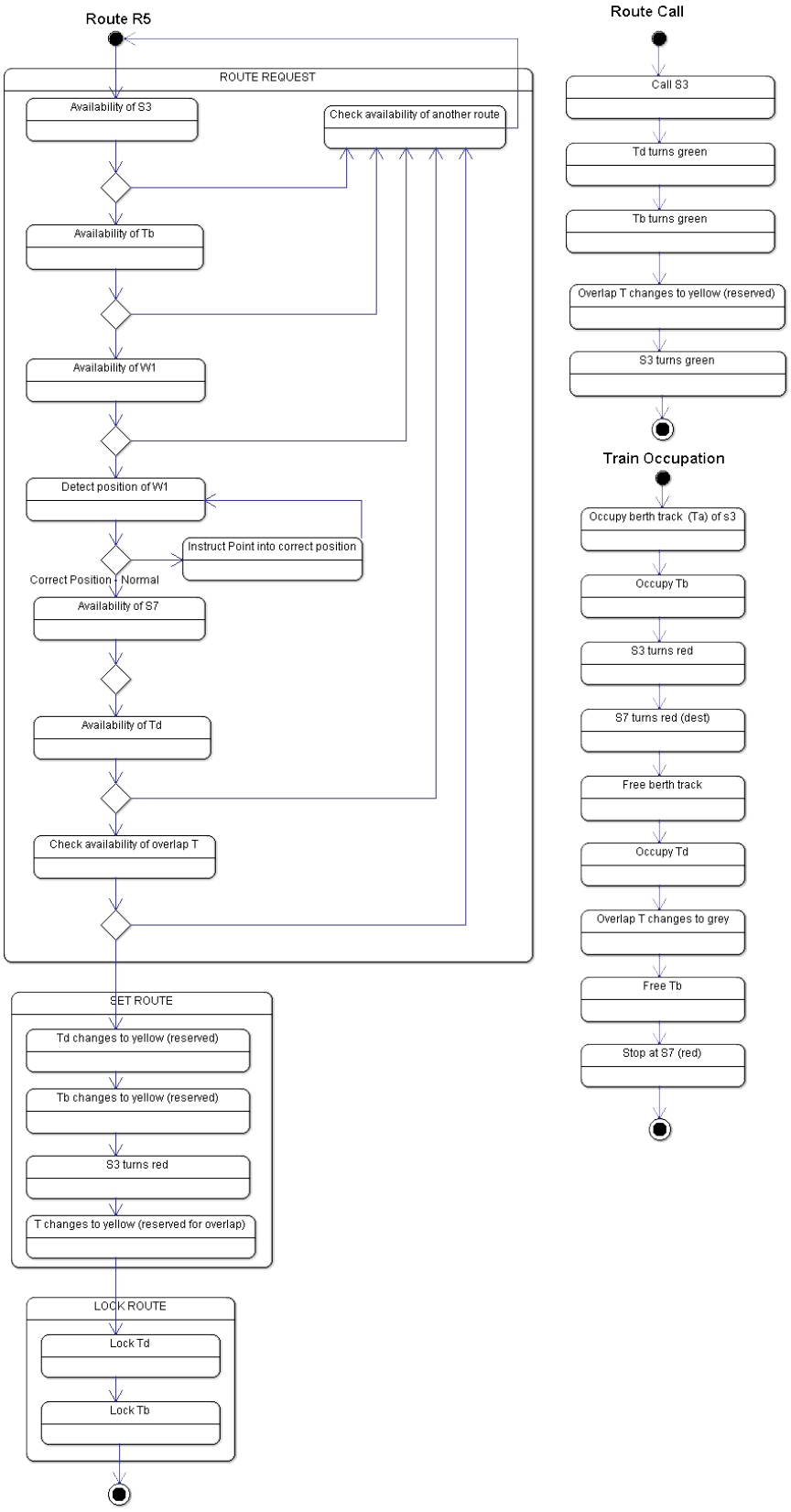


Figure B.13: Route 5 statechart process

### *3.6. Route 6*

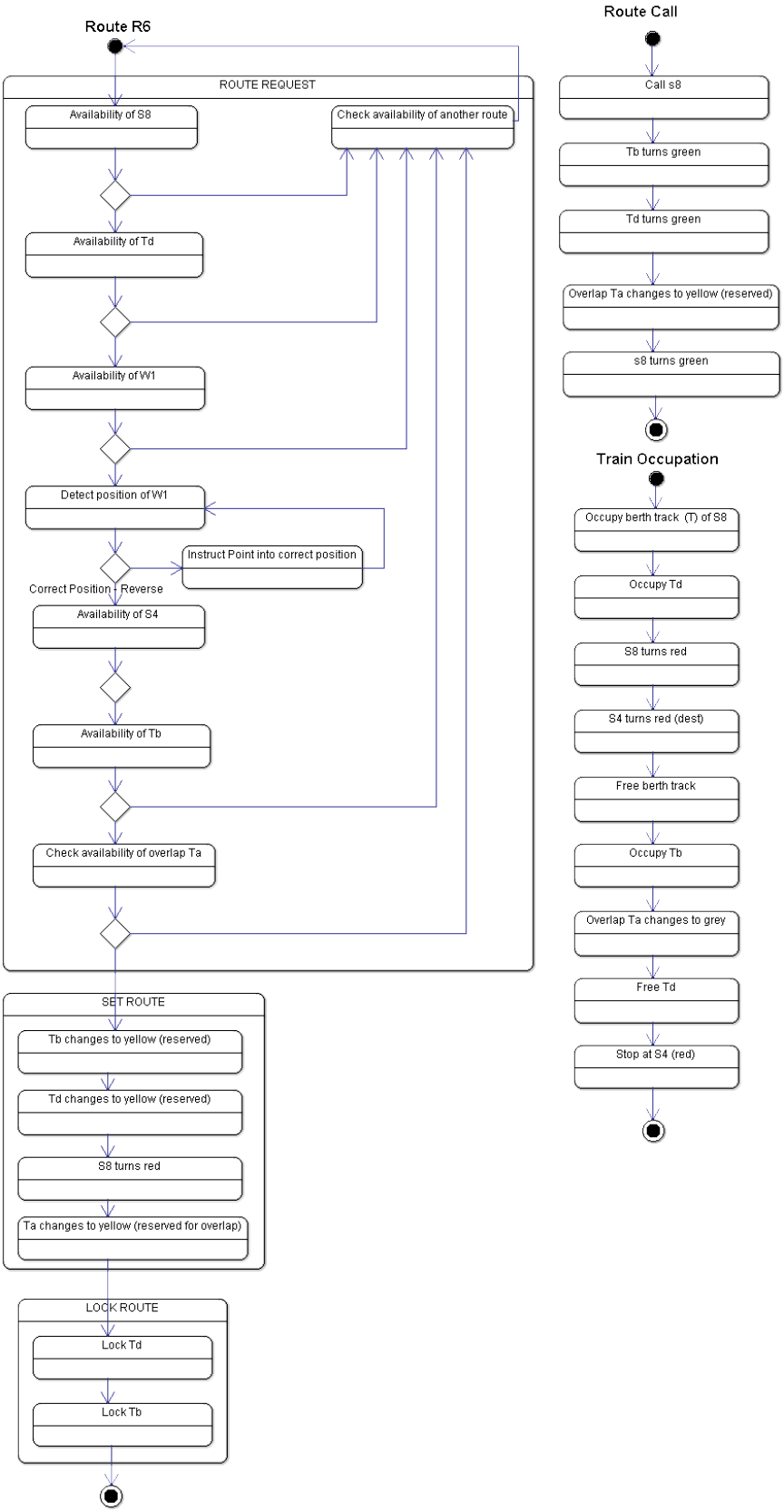


Figure B.14: Route 6 procedure

# APPENDIX C: SIEMENS TIA PORTAL AUTOMATION SOFTWARE

## 1. TIA PORTAL SOFTWARE

TIA (Totally Integrated Automation) Portal is a software engineering framework developed by Siemens. This framework modernises the automation technology industry by integrating the engineering environment into a software project. TIA Portal software enables developers to efficiently design, develop, commission and test automation software. TIA Portal software provides an extremely user-friendly interface which consists of simple functions and allows complete data transparency throughout the application [35]. The software is divided into controller software and HMI (Human Machine Interface) software. The controller software enables developers to program and test all SIMATIC PLCs (Programmable Logic Controllers). The HMI software consists of a family of WinCC software. WinCC is SCADA (Supervisory Control and Data Acquisition) software used to program HMI devices. An HMI device provides a graphical interface for interacting with automated systems. TIA Portal software enables visual displays, reporting, fault logging and user administration functions. Network configured devices also offer remote operation and diagnostics [35].

TIA Portal uses tags as communication between modules. A tag is variable or an address location in the software that is updated by a change in state. Tags are assigned data types and the state of a tag changes based on the module operation it is connected to. Tags are defined internally and are available to modules throughout the project. Software instructions operate predominantly using tag operations [36].

## 2. FEATURES AND ADVANTAGES

TIA Portal provides a uniform look and feel for all editors. The flexible window arrangement provides easy access to software tools. The software offers a flexible screen layout with a clear project tree display which assists with easy graphical configuration. TheTIA key features of TIA Portal include creating, storing and reusing project components such as HMI screens, tags, program blocks and configured modules, etc [39]. This makes the software easy to maintain [35]. TIA Portal also supports fast error localisation and analysis [39].

## 3. LAYOUT AND OPERATION

Figure C.1 below illustrates the layout of the TIA Portal software. The left panel displays the Project tree and provides a list of all the Screens developed in the project. The tags are stored

internally and are displayed in the left panel below the Screen list. Screens that are currently open can easily be accessed in the bottom panel of the application. The editor is located in the center. Graphical elements can easily be located in the right panel and dragged across to the editor. The appearance, layout and position of each graphical element can be configured in the bottom panel as indicated under the Properties tab. TIA Portal also provides accurate error localisation and diagnostic methods.

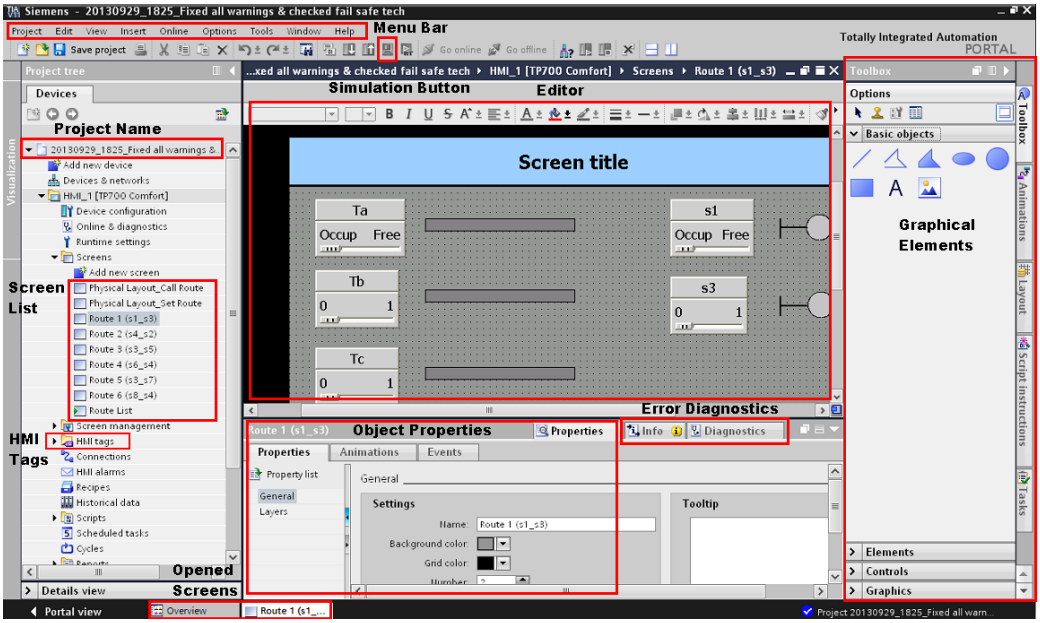


Figure C.1: TIA Portal Layout

To run the software, the Simulation button must be selected which launches the application as a series of screens. Each screen is composed of buttons which are required to interact with the project. Each screen is also updated with the current date and time as indicated in Figure C.2. Based on the selected screen the screen title appears at the top. The screen titled *Route 1 (s1_s3)* is provided as an example. Each screen is composed of button operations, graphical elements and tags. The buttons are configured to trigger transitions from one screen to another. The graphical elements consist of basic design objects provided by TIA Portal. The appearance and structure of these objects are configured especially for modelling a railway system.

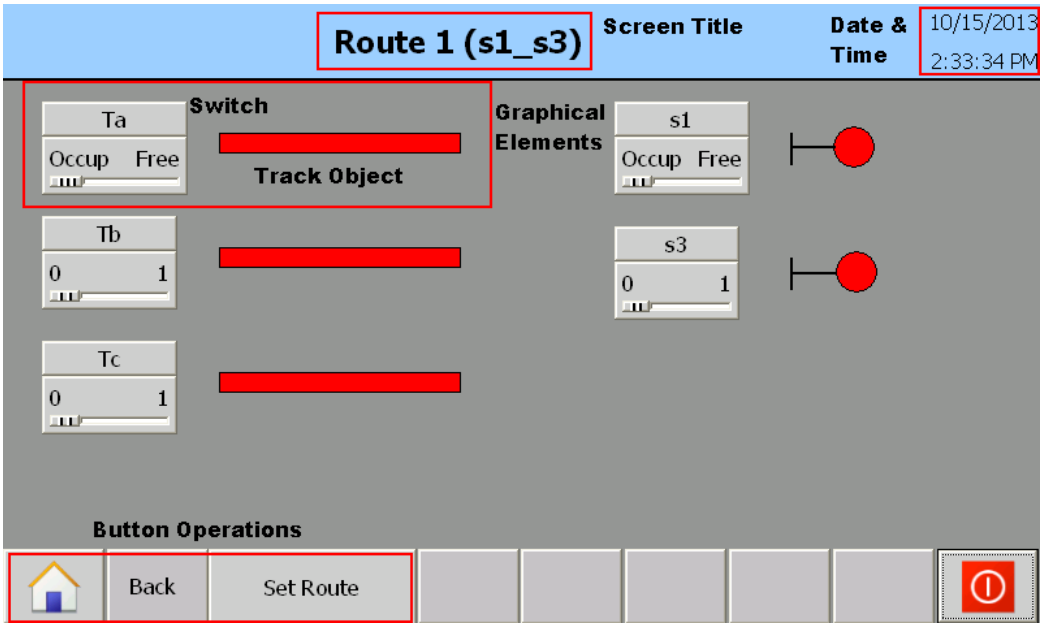


Figure C.2: Route 1 (s1_s3) Screen functions

Each graphical element is connected to a tag with a Boolean data type, i.e. the tag can either be in a 1 or 0 state. Once the state of the tag has been changed, the current state is stored and available for use throughout the application. However, through configuration the state of the tag can be reset at any point in the application. The tags are stored in a Tag table wherein the name and data type of each tag must be specified.

Figure C.3 shows an example of using a Switch object to change the indication of a graphical object through the use of tags. A Boolean tag is connected to the Switch object and the same tag is connected to a graphical track object. The Switch sets the state of the tag and is configured to change between a 1 and 0 states. The track object is configured to display a defined indication based on the current state of the tag. When the tag has a Boolean value of 0, the track object is set to appear red as shown in Case A in Figure C.3. This indicates that the track section is occupied.

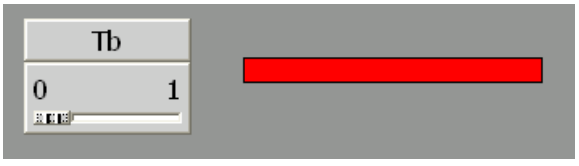


Figure C.3: Switch example – Case A

Conversely when the Switch is changed from a 0 to 1 value, the yrack object is set to appear grey as shown in CASE B in Figure C.4. Setting the Switch back to 0, resets the value of the tag and changes the track object indication back to red.

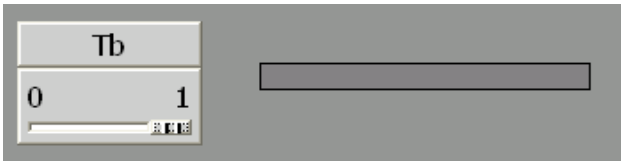


Figure C.4: Switch example – Case B

## APPENDIX D: TIA PORTAL SOFTWARE SIMULATION

The software model has been developed and simulated in TIA Portal to assess the functionality of the interlocking model.

### 1. SCREEN ARCHITECTURE

The architecture depicted in Figure D.1 below illustrates the layout and transition between screens in TIA Portal. The Route List screen provides a list of all the possible routes. After selecting a route, the user transitions from the Route screen to the Set route, Call route and Train occupation screens in the defined order. All screens are equipped with 'Back' and 'Home' buttons. The Back button returns to the previous screen and the Home button returns to the Route List screen.

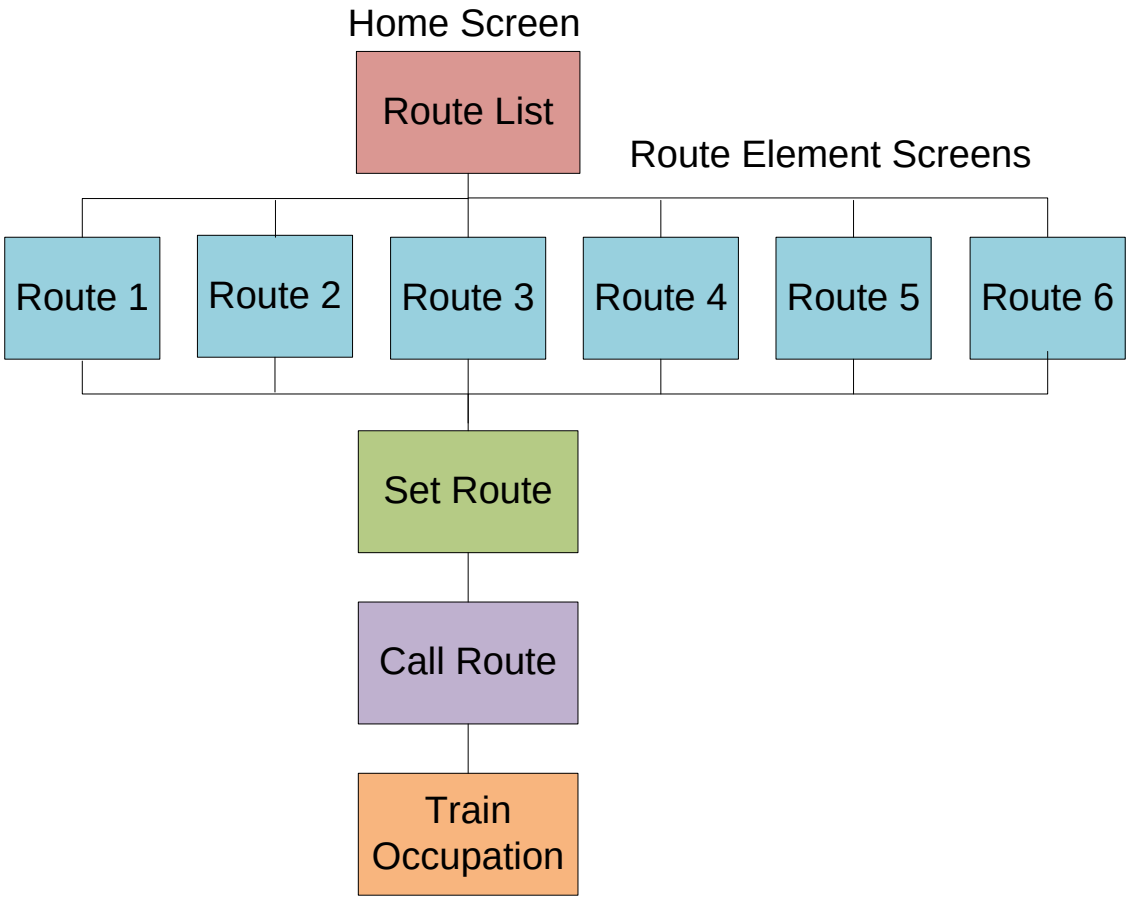


Figure D.1: TIA Portal screen architecture

## 2. **SCREENS**

### *2.1. Route List*

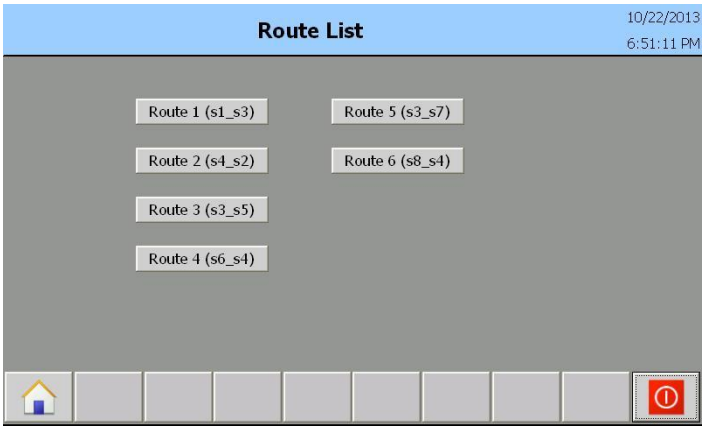


Figure D.2: Route list screen

### *2.2. Route Element Screens*

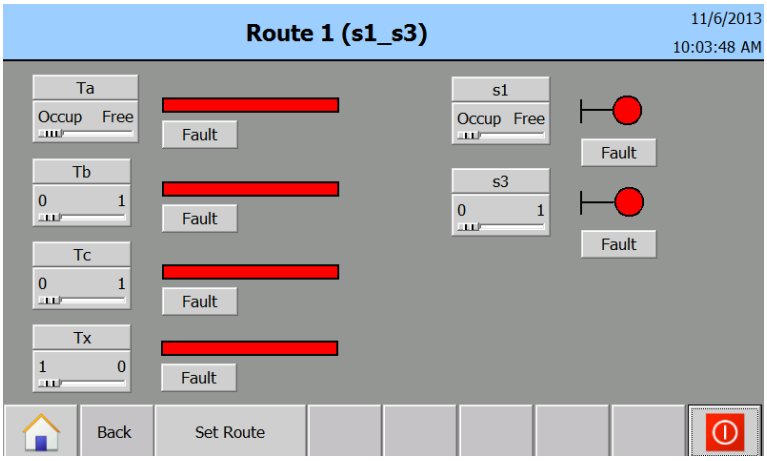


Figure D.3: Route 1 screen

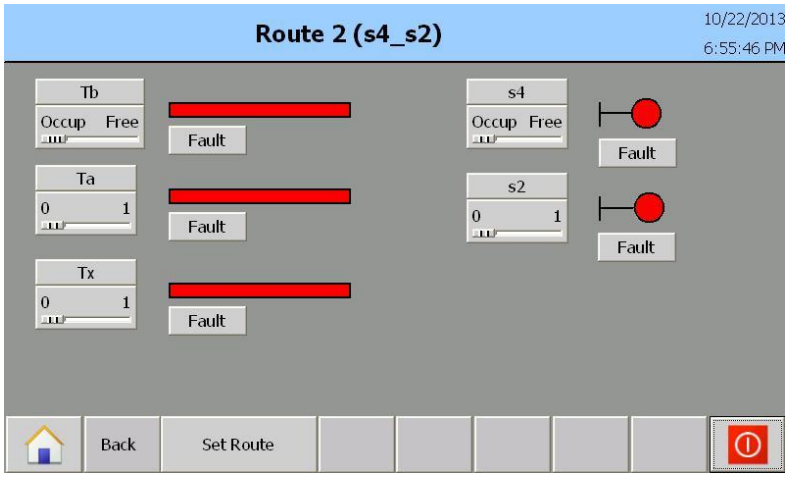


Figure D.4: Route 2 screen

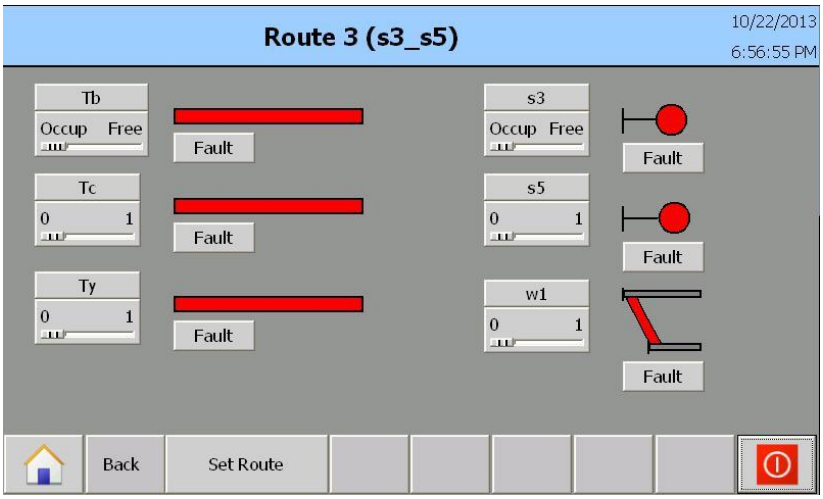


Figure D.5: Route 3 screen

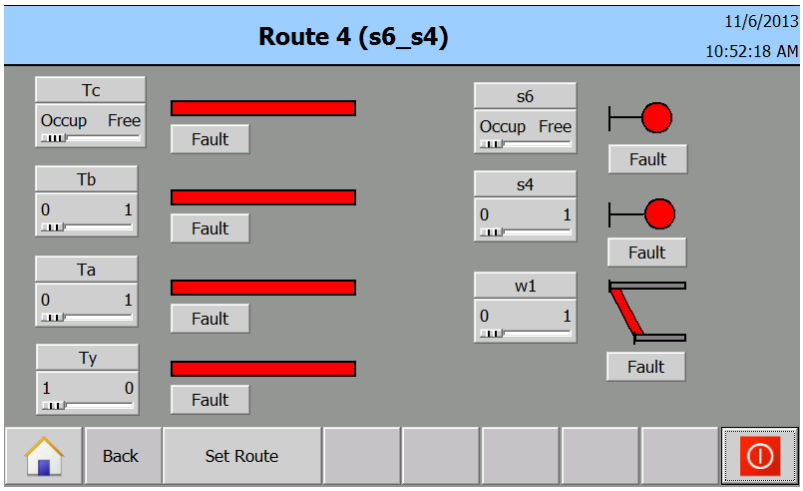


Figure D.6: Route 4 screen

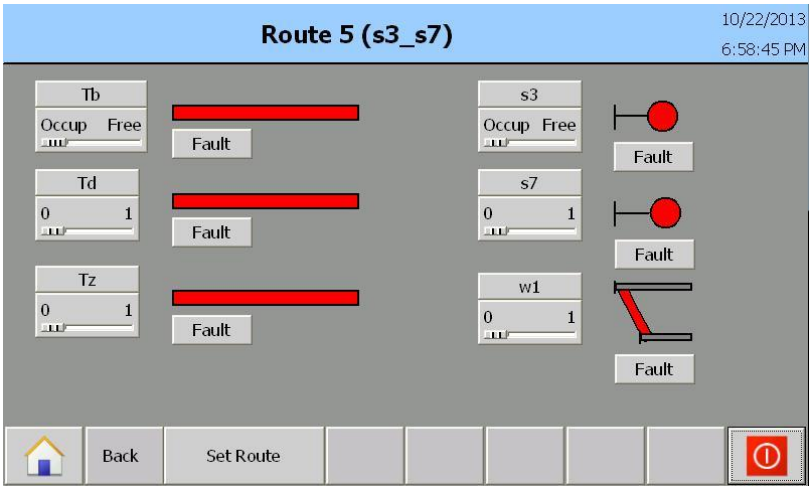


Figure D.7: Route 5 screen

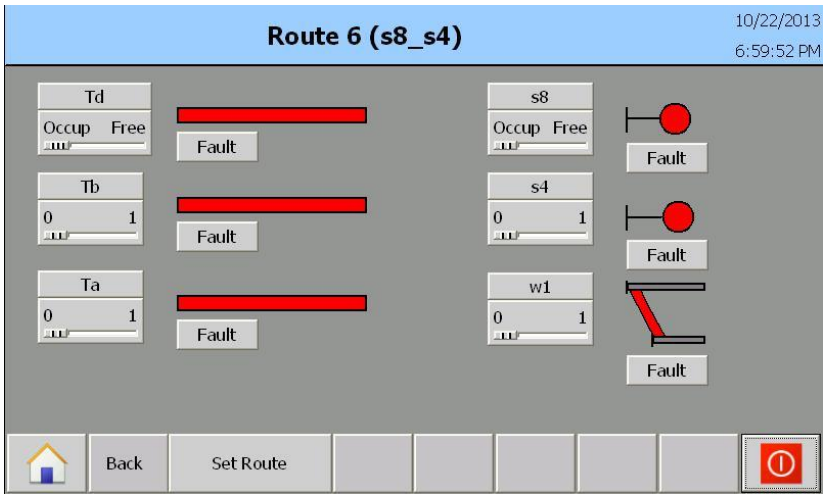


Figure D.8: Route 6 screen

*2.3. Set Route Screen*

As an example, the elements for Route 1 have previously been configured and the route has been set.

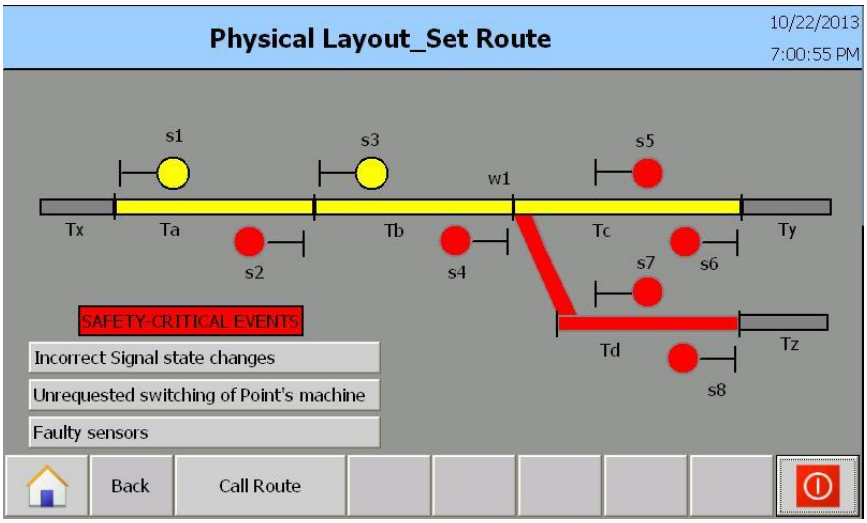


Figure D.9: Physical Layout_Set route screen

*2.4. Call Route Screen*

Similar to the Set Route Screen above, the element indications displayed in calling Route 1 are shown.

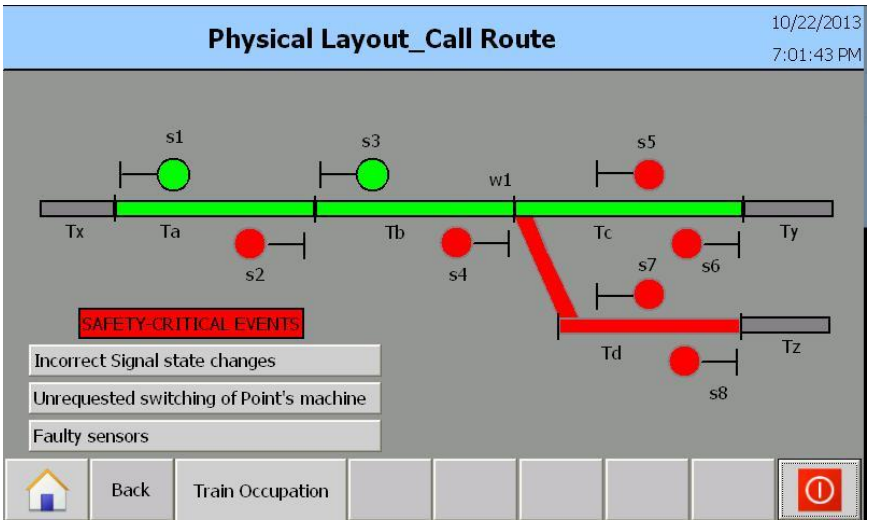


Figure D.10: Physical Layout_Call route screen

*2.5. Train Occupation Screen*

The train occupation procedure for Route 1 has been performed to illustrate the clearing of element indications after a train completes a route as expected.

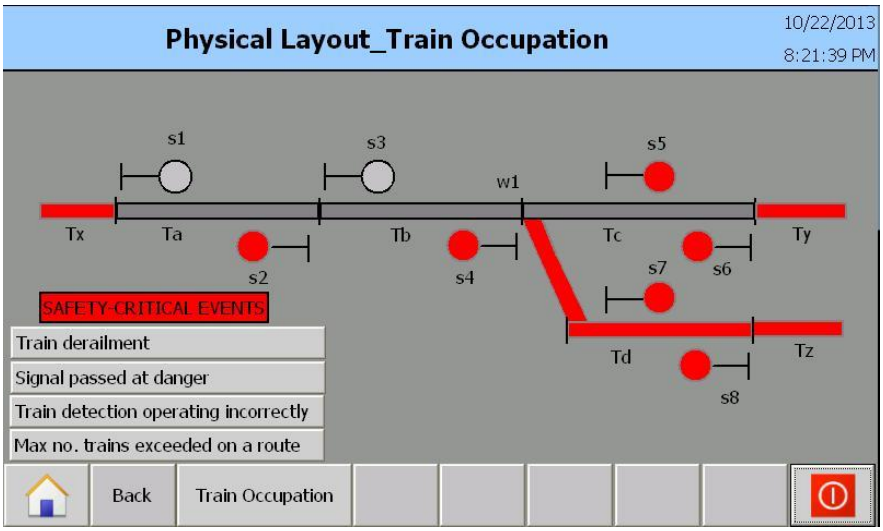


Figure D.11: Physical Layout_Train occupation screen

## APPENDIX E: TEST CASES & RESULTS

### 1. ROUTE 1

The method followed in executing each test case for Route 1 is explained. The same method is applied to the remaining routes, i.e. Route 2 – Route 6 with fine changes in test parameters. The expected and actual outcomes of these test cases are tabulated.

#### *1.1. Case 1: Set a route*

For the first test case all the elements required for Route 1 are set to free, i.e. logical 1. Thereafter the Set Route function is selected to request and reserve the route.

Table E.1: Route 1 – Set route test case

| Element Configurations | Expected State | Actual State |
|---|---|---|
| s1 = 1 (Free) | s1 = Yellow | s1 = Yellow |
| s3 = 1 (Free) | s3 = Yellow | s3 = Yellow |
| Ta = 1 (Free) | Ta = Yellow | Ta = Yellow |
| Tb = 1 (Free) | Tb = Yellow | Tb = Yellow |
| Tc = 1 (Free) | Tc = Yellow | Tc = Yellow |
| Tx = 1 (Free) | Tx = Yellow | Tx = Yellow |
| | Route 1 is set. | Route 1 is set. |

#### *1.2. Case 2: Call a route*

After the route has successfully been set, the Call Route function is selected. The execution of this function indicates that the route is ready for occupation by a train.

Table E.2: Route 1 – Call Route test case parameters

| Expected State | Actual State |
|---|---|
| s1 = Green | s1 = Green |
| s3 = Green | s3 = Green |
| Ta = Green | Ta = Green |
| Tb = Green | Tb = Green |
| Tc = Green | Tc = Green |
| Tx = Green | Tx = Green |
| Route 1 is called. | Route 1 is called. |

### *1.3. Case 3: Train Occupation*

After Route 1 has been set and called, the occupation of the train on the route must be initiated. The change in element indications after the function has been completed must be checked to ensure the train has successfully completed the route.

Table E.3: Route 1 – Train Occupation test case parameters

| Expected State | Actual State |
|---|---|
| s1 = Grey | s1 = Grey |
| s3 = Grey | s3 = Grey |
| Ta = Grey | Ta = Grey |
| Tb = Grey | Tb = Grey |
| Tc = Grey | Tc = Grey |
| Tx = Grey | Tx = Grey |
| Route 1 is cleared. | Route 1 is cleared. |

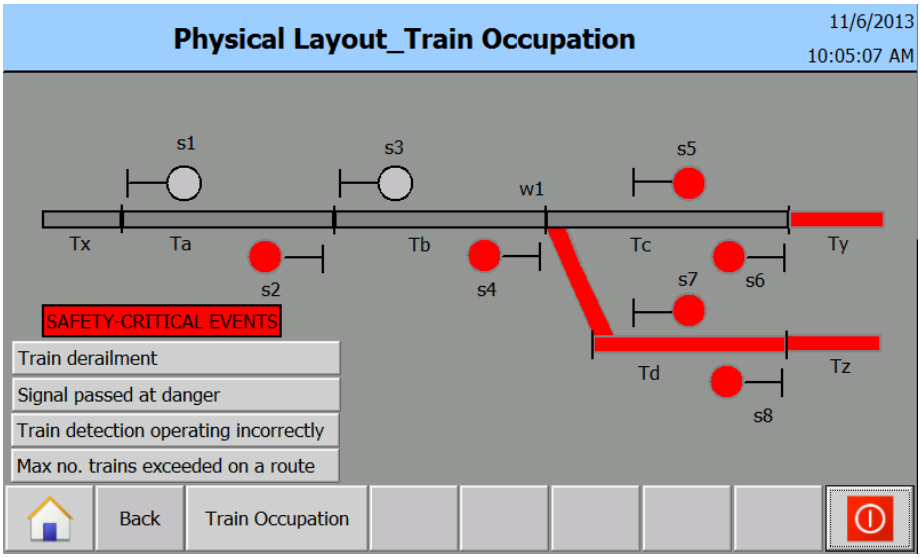


Figure E.1: Observations after train occupation procedure

### *1.4. Case 4: Route cannot be configured*

This test case is designed to evaluate the performance and logic of the interlocking operations. Some of the elements for Route 1 are set to free (i.e. logical 1) and the remaining elements are set to occupied (i.e. logical 0). This technique is aimed at verifying the logic of the route request function as a route cannot be set if all the elements required for the given route are not available. After the

elements are set, the Set Route function must be selected and the behaviour of the system assessed.

Table E.4: Route configuration parameters

| Element Configurations | Expected State | Actual State |
|---|---|---|
| s1 = 1 (Free) | s1 = Yellow | s1 = Yellow |
| s3 = 0 (Occupied) | s3 = Red | s3 = Red |
| Ta = 0 (Occupied) | Ta = Red | Ta = Red |
| Tb = 1 (Free) | Tb = Yellow | Tb = Yellow |
| Tc = 0 (Occupied) | Tc = Red | Tc = Red |
| Tc = 1 (Free) | Tc = Yellow | Tc = Yellow |
| | Route 1 cannot be set. | Route 1 is not set. |

*1.5. Case 5: Initiate an element fault*

After the availability of the elements required for Route 1 has been set to free, a fault is triggered for element s3. The behaviour of the interlocking in response to this element fault is monitored.

Table E.5: Route request – Element Fault

| Element Configurations | Element State | Expected State | Actual State |
|---|---|---|---|
| s1 = 1 (Free) | s1 = 0 (Occupied) | s1 = Red | s1 = Red |
| s3 = 1 (Free) | s3 = 0 (Occupied) | s3 = Red | s3 = Red |
| Ta = 1 (Free) | Ta = 0 (Occupied) | Ta = Red | Ta = Red |
| Tb = 1 (Free) | Tb = 0 (Occupied) | Tb = Red | Tb = Red |
| Tc = 1 (Free) | Tc = 0 (Occupied) | Tc = Red | Tc = Red |
| Tx = 1 (Free) | Tx = 0 (Occupied) | Tx = Red | Tx = Red |
| | Fault triggered for element s3 | Route request is not issued. Route 1 cannot be set. | Cannot issue route request. Route 1 is not set. |

*1.6. Case 6: Set a route and initiate a safety-critical event*

After the availability of the elements for Route 1 has been set to free, the Set Route function is selected. After selecting this function, one of the safety-critical events listed is selected. For this test case, “Incorrect signal state changes” has been selected as shown in Figure E.2. The response of the interlocking to this event is observed.

Table E.6: Set route – Safety-critical event

| Expected State | Actual State |
|---|---|
| s1 = Red | s1 = Red |
| s3 = Red | s3 = Red |
| Ta = Red | Ta = Red |
| Tb = Red | Tb = Red |
| Tc = Red | Tc = Red |
| Tx = Red | Tx = Red |
| Element indications must change to red. Route must be cancelled. | All elements turn red. Route 1 is cancelled. |

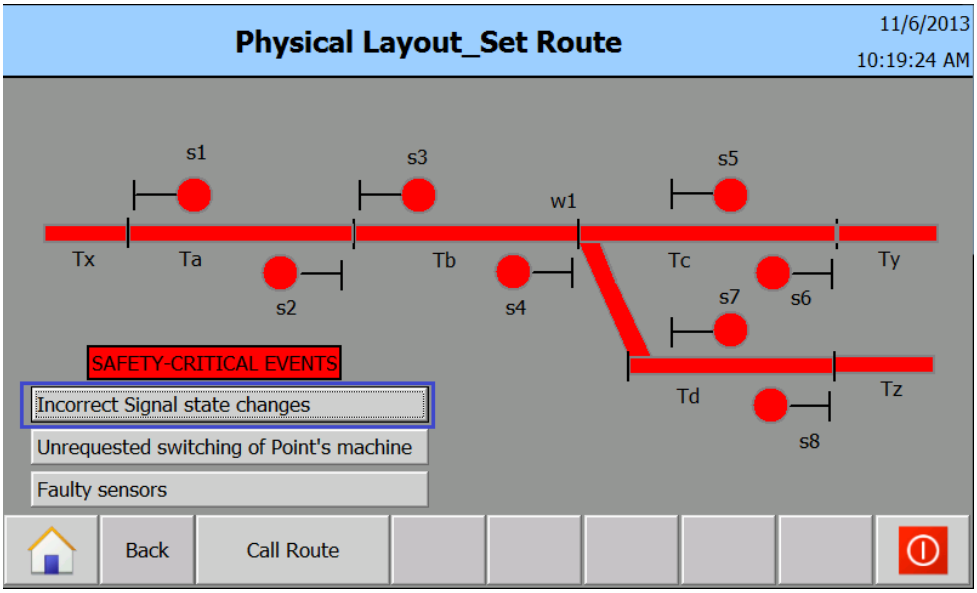


Figure E.2: Safety-critical event triggered

*1.7. Case 7: Call a route and initiate a safety-critical event*

Similar to the above case, after the elements have been set to free and the Set Route function has been executed, the Call Route function must be selected. After this function has been selected, a safety-critical event must be triggered. The "Unrequested switching of point's machine" event has been selected and the behaviour of the interlocking to this event has been assessed.

Table E.7: Call route – Safety-critical event

| Expected State | Actual State |
|---|---|
| s1 = Red | s1 = Red |
| s3 = Red | s3 = Red |
| Ta = Red | Ta = Red |
| Tb = Red | Tb = Red |
| Tc = Red | Tc = Red |
| Tx = Red | Tx = Red |
| Element indications turn red. Route must be cancelled. | All elements in layout turn red. Route 1 is cancelled. |

*1.8. Case 8: Train occupation with a safety-critical event initiated*

Subsequent to the Set and Call Route functions being executed, the Train Occupation function must be selected. After this function has been selected, the "Train derailment" safety-critical event is triggered. The response of the interlocking to this event is observed.

Table E.8: Train occupation – Safety-critical event

| Expected State | Actual State |
|---|---|
| s1 = Red | s1 = Red |
| s3 = Red | s3 = Red |
| Ta = Red | Ta = Red |
| Tb = Red | Tb = Red |
| Tc = Red | Tc = Red |
| Tx = Red | Tx = Red |
| Element indications turn from grey to red. Elements are set to occupied. | All signalling elements turn red. |

2. **ROUTE 2**

*2.1. Case 1: Set a route*

Table E.9: Route 2 – Set route test case

| Element Configurations | Expected State | Actual State |
|---|---|---|
| s2 = 1 (Free) | s2 = Yellow | s2 = Yellow |
| s4 = 1 (Free) | s4 = Yellow | s4 = Yellow |
| Ta = 1 (Free) | Ta = Yellow | Ta = Yellow |
| Tb = 1 (Free) | Tb = Yellow | Tb = Yellow |
| Tx = 1 (Free) | Tx = Yellow | Tx = Yellow |
| | Route 2 is set. | Route 2 is set. |

*2.2. Case 2: Call a route*

Table E.10: Route 2 – Call Route test case parameters

| Expected State | Actual State |
|---|---|
| s2 = Green | s2 = Green |
| s4 = Green | s4 = Green |
| Ta = Green | Ta = Green |
| Tb = Green | Tb = Green |
| Tx = Green | Tx = Green |
| Route 2 is called. | Route 2 is called. |

*2.3. Case 3: Train Occupation*

Table E.11: Route 2 – Train Occupation test case parameters

| Expected State | Actual State |
|---|---|
| s2 = Grey | s2 = Grey |
| s4 = Grey | s4 = Grey |
| Ta = Grey | Ta = Grey |
| Tb = Grey | Tb = Grey |
| Tx = Grey | Tx = Grey |
| Route 1 is cleared. | Route 1 is cleared. |

### *2.4. Case 4: Route cannot be configured*

Table E.12: Route configuration parameters

| Element Configurations | Expected State | Actual State |
|---|---|---|
| s2 = 0 (Occupied) | s2 = Red | s2 = Red |
| s4 = 1 (Free) | s4 = Yellow | s4 = Yellow |
| Ta = 0 (Occupied) | Ta = Red | Ta = Red |
| Tb = 1 (Free) | Tb = Yellow | Tb = Yellow |
| Tx = 1 (Free) | Tx = Yellow | Tx = Yellow |
| | Route 2 cannot be set. | Route 2 is not set. |

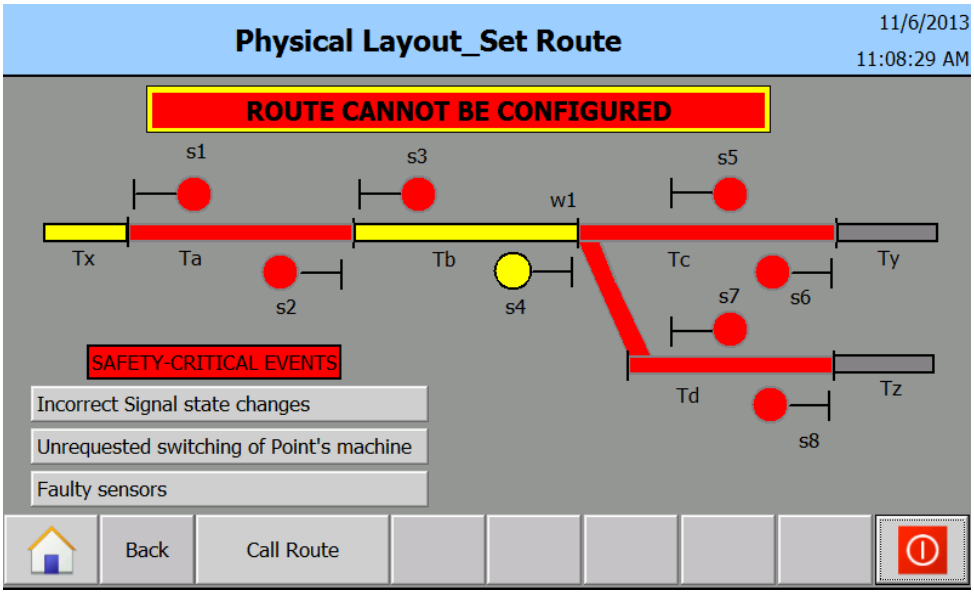


Figure E.3: Route 2 cannot be configured

### *2.5. Case 5: Initiate an element fault*

For this test case, a fault has been triggered for element s4 as shown in Figure E.4.

Table E.13: Route request – Element Fault

| Elememt Configurations | Element State | Expected State | Actual State |
|---|---|---|---|
| s2 = 1 (Free) | s2 = 0 (Occupied) | s2 = Red | s2 = Red |
| s4 = 1 (Free) | s4 = 0 (Occupied) | s4 = Red | s4 = Red |
| Ta = 1 (Free) | Ta = 0 (Occupied) | Ta = Red | Ta = Red |
| Tb = 1 (Free) | Tb = 0 (Occupied) | Tb = Red | Tb = Red |
| Tx = 1 (Free) | Tx = 0 (Occupied) | Tx = Red | Tx = Red |
| | Fault triggered for element s4 | Route request is not issued. Route 2 cannot be set. | Cannot issue route request. Route 2 is not set. |

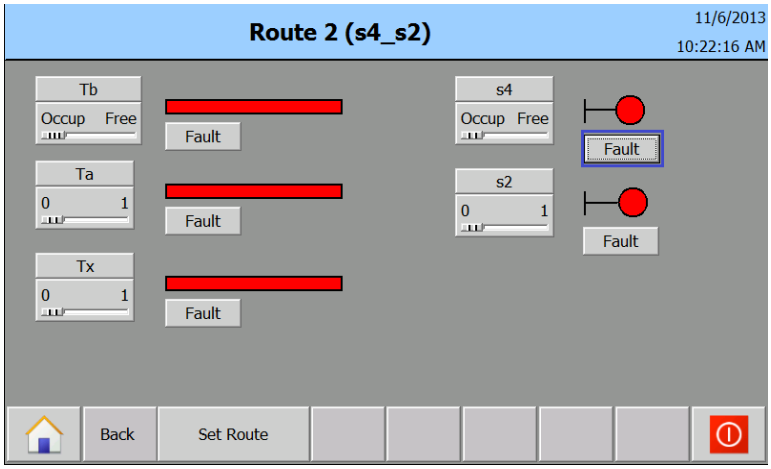


Figure E.4: Element s4 fault triggered

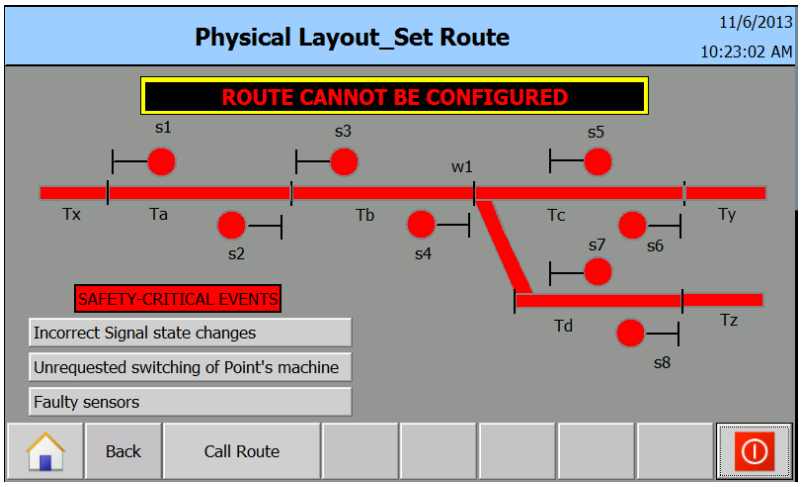


Figure E.5: Route cannot be configured due to s4 element fault

*2.6. Case 6: Set a route and initiate a safety-critical event*

Upon selecting the Set Route function, the "Faulty Sensors" safety-critical event is triggered.

Table E.14: Set route – Safety-critical event

| Expected State | Actual State |
|---|---|
| s2 = Red | s2 = Red |
| s4 = Red | s4 = Red |
| Ta = Red | Ta = Red |
| Tb = Red | Tb = Red |
| Tx = Red | Tx = Red |
| Element indications must change to red. Route must be cancelled. | All elements turn red. Route 2 is cancelled. |

*2.7. Case 7: Call a route and initiate a safety-critical event*

For this test case, the "Incorrect signal state changes" event has been selected.

Table E.15: Call route – Safety-critical event

| Expected State | Actual State |
|---|---|
| s2 = Red | s2 = Red |
| s4 = Red | s4 = Red |
| Ta = Red | Ta = Red |
| Tb = Red | Tb = Red |
| Tx = Red | Tx = Red |
| Element indications turn red. Route must be cancelled. | All elements in layout turn red. Route 2 is cancelled. |

*2.8. Case 8: Train occupation with a safety-critical event initiated*

After the setting and calling of Route 2, the train occupation function is executed. After this function has been selected the "signal passed at danger" safety-critical event is triggered.

Table E.16: Train occupation – Safety-critical event

| Expected State | Actual State |
|---|---|
| s2 = Red | s2 = Red |
| s4 = Red | s4 = Red |
| Ta = Red | Ta = Red |
| Tb = Red | Tb = Red |
| Tx = Red | Tx = Red |
| Element indications turn from grey to red. Elements are set to occupied. | All signalling elements turn red. |

3. **ROUTE 3**

*3.1. Case 1: Set a route*

Table E.17: Route 3 – Set route test case

| Element Configurations | Expected State | Actual State |
|---|---|---|
| s3 = 1 (Free) | s3 = Yellow | s3 = Yellow |
| s5 = 1 (Free) | s5 = Yellow | s5 = Yellow |
| Tb = 1 (Free) | Tb = Yellow | Tb = Yellow |
| Tc = 1 (Free) | Tc = Yellow | Tc = Yellow |
| Ty = 1 (Free) | Ty = Yellow | Ty = Yellow |
| w1 = 1 (Free) | w1 = Yellow | w1 = Yellow |
| | Route 3 is set. | Route 3 is set. |

*3.2. Case 2: Call a route*

Table E.18: Route 3 – Call Route test case parameters

| Expected State | Actual State |
|---|---|
| s3 = Green | s3 = Green |
| s5 = Green | s5 = Green |
| Tb = Green | Tb = Green |
| Tc = Green | Tc = Green |
| Ty = Green | Ty = Green |
| w1 = Green | w1 = Green |
| Route 3 is called. | Route 3 is called. |

*3.3. Case 3: Train Occupation*

Table E.19: Route 3 – Train Occupation test case parameters

| Expected State | Actual State |
|---|---|
| s3 = Grey | s3 = Grey |
| s5 = Grey | s5 = Grey |
| Tb = Grey | Tb = Grey |
| Tc = Grey | Tc = Grey |
| Ty = Grey | Ty = Grey |
| w1 = Grey | w1 = Grey |
| Route 3 is cleared. | Route 3 is cleared. |

*3.4. Case 4: Route cannot be configured*

Table E.20: Route configuration parameters

| Element Configurations | Expected State | Actual State |
|---|---|---|
| s3 = 1 (Free) | s3 = Yellow | s3 = Yellow |
| s5 = 0 (Occupied) | s5 = Red | s5 = Red |
| Tb = 1 (Free) | Tb = Yellow | Tb = Yellow |
| Tc = 0 (Occupied) | Tc = Red | Tc = Red |
| Ty = 1 (Free) | Ty = Yellow | Ty = Yellow |
| w1 = 1 (Free) | w1 = Yellow | w1 = Yellow |
| | Route 3 cannot be set. | Route 3 is not set. |

*3.5. Case 5: Initiate an element fault*

For Route 3, a fault has been initiated for element Tc as shown in Table E.21 below.

Table E.21: Route request – Element Fault

| Element Configurations | Element State | Expected State | Actual State |
|---|---|---|---|
| s3 = 1 (Free) | s3 = 0 (Occupied) | s3 = Red | s3 = Red |
| s5 = 1 (Free) | s5 = 0 (Occupied) | s5 = Red | s5 = Red |
| Tb = 1 (Free) | Tb = 0 (Occupied) | Tb = Red | Tb = Red |
| Tc = 1 (Free) | Tc = 0 (Occupied) | Tc = Red | Tc = Red |
| Ty = 1 (Free) | Ty = 0 (Occupied) | Ty = Red | Ty = Red |
| w1 = 1 (Free) | w1 = 0 (Occupied) | w1 = Red | w1 = Red |
| | Fault triggered for element Tc | Route request is not issued. Route 3 cannot be set. | Cannot issue route request. Route 3 is not set. |

*3.6. Case 6: Set a route and initiate a safety-critical event*

For this test case, the "Unrequested switching of point's machine" critical event has been selected.

Table E.22: Set route – Safety-critical event

| Expected State | Actual State |
|---|---|
| s3 = Red | s3 = Red |
| s5 = Red | s5 = Red |
| Tb = Red | Tb = Red |
| Tc = Red | Tc = Red |
| Ty = Red | Ty = Red |
| w1 = Red | w1 = Red |
| Element indications must change to red. Route must be cancelled. | All elements turn red. Route 3 is cancelled. |

*3.7. Case 7: Call a route and initiate a safety-critical event*

After the Call Route function has been selected, the "Faulty sensors" safety-critical event has been initiated.

Table E.23: Call route – Safety-critical event

| Expected State | Actual State |
|---|---|
| s3 = Red | s3 = Red |
| s5 = Red | s5 = Red |
| Tb = Red | Tb = Red |
| Tc = Red | Tc = Red |
| Ty = Red | Ty = Red |
| w1 = Red | w1 = Red |
| Element indications turn red. Route must be cancelled. | All elements in layout turn red. Route 3 is cancelled. |

*3.8. Case 8: Train occupation with a safety-critical event initiated*

For this test case, the "Train detection operating incorrectly" event has been selected.

Table E.24: Train occupation – Safety-critical event

| Expected State | Actual State |
|---|---|
| s3 = Red | s3 = Red |
| s5 = Red | s5 = Red |
| Tb = Red | Tb = Red |
| Tc = Red | Tc = Red |
| Ty = Red | Ty = Red |
| w1 = Red | w1 = Red |
| Element indications turn from grey to red. Elements are set to occupied. | All signalling elements turn red. |

4. **ROUTE 4**

*4.1. Case 1: Set a route*

Table E.25: Route 4 – Set route test case

| Element Configurations | Expected State | Actual State |
|---|---|---|
| s4 = 1 (Free) | s4 = Yellow | s4 = Yellow |
| s6 = 1 (Free) | s6 = Yellow | s6 = Yellow |
| Ta = 1 (Free) | Ta = Yellow | Ta = Yellow |
| Tb = 1 (Free) | Tb = Yellow | Tb = Yellow |
| Tc = 1 (Free) | Tc = Yellow | Tc = Yellow |
| w1 = 1 (Free) | w1 = Yellow | w1 = Yellow |
| | Route 4 is set. | Route 4 is set. |

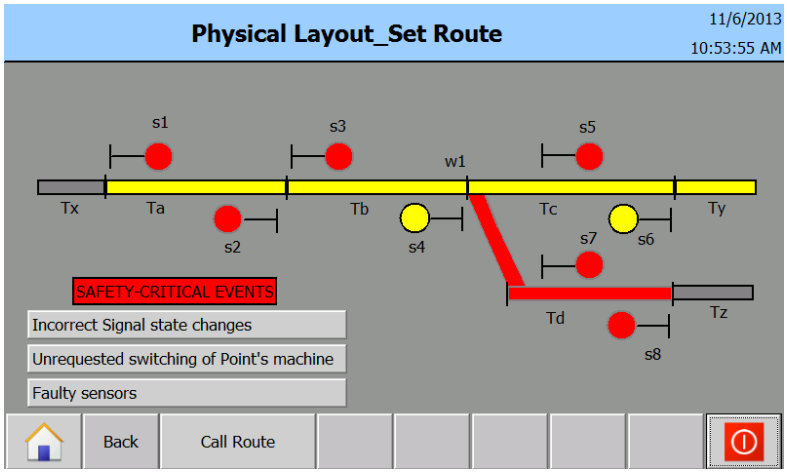


Figure E.6: Set route 4 test

*4.2. Case 2: Call a route*

Table E.26: Route 4 – Call Route test case parameters

| Expected State | Actual State |
|---|---|
| s4 = Green | s4 = Green |
| s6 = Green | s6 = Green |
| Ta = Green | Ta = Green |
| Tb = Green | Tb = Green |
| Tc = Green | Tc = Green |
| w1 = Green | w1 = Green |
| Route 4 is called. | Route 4 is called. |

*4.3. Case 3: Train Occupation*

Table E.27: Route 4 – Train Occupation test case parameters

| Expected State | Actual State |
|---|---|
| s4 = Grey | s4 = Grey |
| s6 = Grey | s6 = Grey |
| Ta = Grey | Ta = Grey |
| Tb = Grey | Tb = Grey |
| Tc = Grey | Tc = Grey |
| w1 = Grey | w1 = Grey |
| Route 4 is cleared. | Route 4 is cleared. |

*4.4. Case 4: Route cannot be configured*

Table E.28: Route configuration parameters

| Element Configurations | Expected State | Actual State |
|---|---|---|
| s4 = 0 (Occupied) | s4 = Red | s4 = Red |
| s6 = 0 (Occupied) | s6 = Red | s6 = Red |
| Ta = 1 (Free) | Ta = Yellow | Ta = Yellow |
| Tb = 1 (Free) | Tb = Yellow | Tb = Yellow |
| Tc = 0 (Occupied) | Tc = Red | Tc = Red |
| w1 = 0 (Occupied) | w1 = Red | w1 = Red |
| | Route 4 cannot be set. | Route 4 is not set. |

*4.5. Case 5: Initiate an element fault*

For this test case, a fault has been initiated for element w1.

Table E.29: Route request – Element Fault

| Element Configurations | Element State | Expected State | Actual State |
|---|---|---|---|
| s4 = 1 (Free) | s4 = 0 (Occupied) | s4 = Red | s4 = Red |
| s6 = 1 (Free) | s6 = 0 (Occupied) | s6 = Red | s6 = Red |
| Ta = 1 (Free) | Ta = 0 (Occupied) | Ta = Red | Ta = Red |
| Tb = 1 (Free) | Tb = 0 (Occupied) | Tb = Red | Tb = Red |
| Tc = 1 (Free) | Tc = 0 (Occupied) | Tc = Red | Tc = Red |
| w1 = 1 (Free) | w1 = 0 (Occupied) | w1 = Red | w1 = Red |
| | Fault triggered for element w1 | Route request is not issued. Route 4 cannot be set. | Cannot issue route request. Route 4 is not set. |

*4.6. Case 6: Set a route and initiate a safety-critical event*

After the Set Route function is selected, the “Faulty sensors” critical event is triggered.

Table E.30: Set route – Safety-critical event

| Expected State | Actual State |
|---|---|
| s4 = Red | s4 = Red |
| s6 = Red | s6 = Red |
| Ta = Red | Ta = Red |
| Tb = Red | Tb = Red |
| Tc = Red | Tc = Red |
| w1 = Red | w1 = Red |
| Element indications must change to red. Route must be cancelled. | All elements turn red. Route 4 is cancelled. |

*4.7. Case 7: Call a route and initiate a safety-critical event*

For this test case, the “Incorrect signal state changes” critical event has been selected.

Table E.31: Call route – Safety-critical event

| Expected State | Actual State |
|---|---|
| s4 = Red | s4 = Red |
| s6 = Red | s6 = Red |
| Ta = Red | Ta = Red |
| Tb = Red | Tb = Red |
| Tc = Red | Tc = Red |
| w1 = Red | w1 = Red |
| Element indications turn red. Route must be cancelled. | All elements in layout turn red. Route 4 is cancelled. |

*4.8. Case 8: Train occupation with a safety-critical event initiated*

After the Train Occupation function has been executed, the "Train derailment" safety-critical event is initiated as shown in Figure E.7.

Table E.32: Train occupation – Safety-critical event

| Expected State | Actual State |
|---|---|
| s4 = Red | s4 = Red |
| s6 = Red | s6 = Red |
| Ta = Red | Ta = Red |
| Tb = Red | Tb = Red |
| Tc = Red | Tc = Red |
| w1 = Red | w1 = Red |
| Element indications turn from grey to red. Elements are set to occupied. | All signalling elements turn red. |

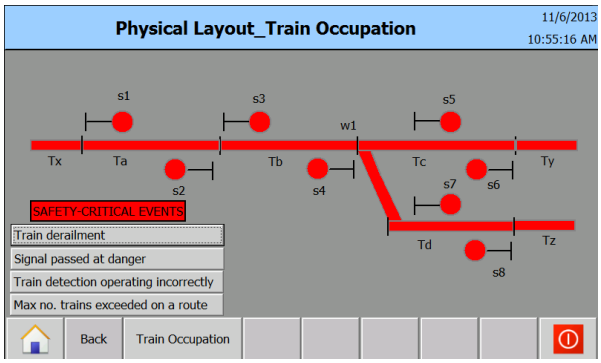


Figure E.7: Safety-critical event triggered

## 5. **ROUTE 5**

### *5.1. Case 1: Set a route*

Table E.33: Route 5 – Set route test case

| Element Configurations | Expected State | Actual State |
|---|---|---|
| s3 = 1 (Free) | s3 = Yellow | s3 = Yellow |
| s7 = 1 (Free) | s7 = Yellow | s7 = Yellow |
| Tb = 1 (Free) | Tb = Yellow | Tb = Yellow |
| Td = 1 (Free) | Td = Yellow | Td = Yellow |
| Tz = 1 (Free) | Tz = Yellow | Tz = Yellow |
| w1 = 1 (Free) | w1 = Yellow | w1 = Yellow |
| | Route 5 is set. | Route 5 is set. |

### *5.2. Case 2: Call a route*

Table E.34: Route 5 – Call Route test case parameters

| Expected State | Actual State |
|---|---|
| s3 = Green | s3 = Green |
| s7 = Green | s7 = Green |
| Tb = Green | Tb = Green |
| Td = Green | Td = Green |
| Tz = Green | Tz = Green |
| w1 = Green | w1 = Green |
| Route 5 is called. | Route 5 is called. |

*5.3. Case 3: Train Occupation*

Table E.35: Route 5 – Train Occupation test case parameters

| Expected State | Actual State |
|---|---|
| s3 = Grey | s3 = Grey |
| s7 = Grey | s7 = Grey |
| Tb = Grey | Tb = Grey |
| Td = Grey | Td = Grey |
| Tz = Grey | Tz = Grey |
| w1 = Grey | w1 = Grey |
| Route 5 is cleared. | Route 5 is cleared. |

*5.4. Case 4: Route cannot be configured*

Table E.36: Route configuration parameters

| Element Configurations | Expected State | Actual State |
|---|---|---|
| s3 = 1 (Free) | s3 = Yellow | s3 = Yellow |
| s7 = 0 (Occupied) | s7 = Red | s7 = Red |
| Tb = 0 (Occupied) | Tb = Red | Tb = Red |
| Td = 1 (Free) | Td = Yellow | Td = Yellow |
| Tz = 0 (Occupied) | Tz = Red | Tz = Red |
| w1 = 0 (Occupied) | w1 = Red | w1 = Red |
| | Route 5 cannot be set. | Route 5 is not set. |

*5.5. Case 5: Initiate an element fault*

For Route 5, a fault has been initiated for element Tz as illustrated in Table E.37.

Table E.37: Route request – Element Fault

| Element Configurations | Element State | Expected State | Actual State |
|---|---|---|---|
| s3 = 1 (Free) | s3 = 0 (Occupied) | s3 = Red | s3 = Red |
| s7 = 1 (Free) | s7 = 0 (Occupied) | s7 = Red | s7 = Red |
| Tb = 1 (Free) | Tb = 0 (Occupied) | Tb = Red | Tb = Red |
| Td = 1 (Free) | Td = 0 (Occupied) | Td = Red | Td = Red |
| Tz = 1 (Free) | Tz = 0 (Occupied) | Tz = Red | Tz = Red |
| w1 = 1 (Free) | w1 = 0 (Occupied) | w1 = Red | w1 = Red |
| | Fault triggered for element Tz | Route request is not issued. Route 5 cannot be set. | Cannot issue route request. Route 5 is not set. |

*5.6. Case 6: Set a route and initiate a safety-critical event*

After execution of the Set Route function, the "Unrequested switching of point's machine" event is triggered.

Table E.38: Set route – Safety-critical event

| Expected State | Actual State |
|---|---|
| s3 = Red | s3 = Red |
| s7 = Red | s7 = Red |
| Tb = Red | Tb = Red |
| Td = Red | Td = Red |
| Tz = Red | Tz = Red |
| w1 = Red | w1 = Red |
| Element indications must change to red. Route must be cancelled. | All elements turn red. Route 5 is cancelled. |

*5.7. Case 7: Call a route and initiate a safety-critical event*

For this test scenario, the "Faulty sensors" critical event has been selected as shown in Figure E.8.

Table E.39: Call route – Safety-critical event

| Expected State | Actual State |
|---|---|
| s3 = Red | s3 = Red |
| s7 = Red | s7 = Red |
| Tb = Red | Tb = Red |
| Td = Red | Td = Red |
| Tz = Red | Tz = Red |
| w1 = Red | w1 = Red |
| Element indications turn red. Route must be cancelled. | All elements in layout turn red. Route 5 is cancelled. |

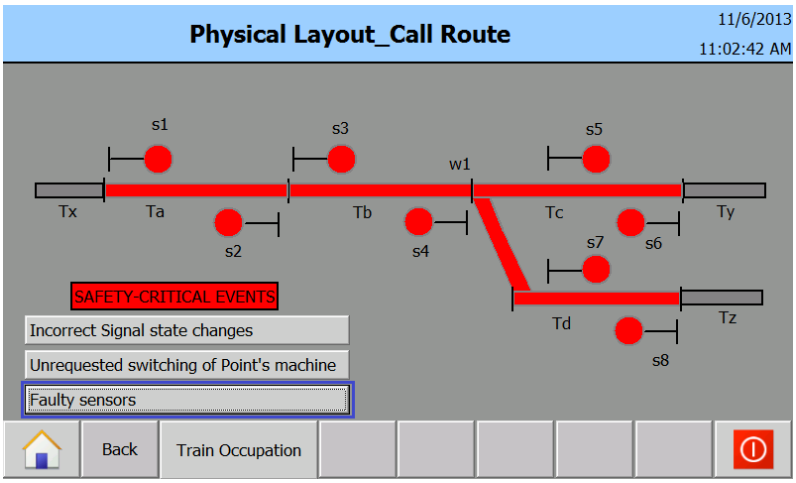


Figure E.8: Safety-critical event initiated after calling route 5

*5.8. Case 8: Train occupation with a safety-critical event initiated*

After Route 5 has been set and called, the Train Occupation function is executed. After execution of this function the "Maximum number of trains exceeded on a route" is triggered.

Table E.40: Train occupation – Safety-critical event

| Expected State | Actual State |
|---|---|
| s3 = Red | s3 = Red |
| s7 = Red | s7 = Red |
| Tb = Red | Tb = Red |
| Td = Red | Td = Red |
| Tz = Red | Tz = Red |
| w1 = Red | w1 = Red |
| Element indications turn from grey to red. Elements are set to occupied. | All signalling elements turn red. |

## 6. **ROUTE 6**

### *6.1. Case 1: Set a route*

Table E.41: Route 6 – Set route test case

| Element Configurations | Expected State | Actual State |
|---|---|---|
| s4= 1 (Free) | s4 = Yellow | s4 = Yellow |
| s8 = 1 (Free) | s8 = Yellow | s8 = Yellow |
| Ta = 1 (Free) | Ta = Yellow | Ta = Yellow |
| Tb = 1 (Free) | Tb = Yellow | Tb = Yellow |
| Td = 1 (Free) | Td = Yellow | Td = Yellow |
| w1 = 1 (Free) | w1 = Yellow | w1 = Yellow |
| | Route 6 is set. | Route 6 is set. |

### *6.2. Case 2: Call a route*

Table E.42: Route 6 – Call Route test case parameters

| Expected State | Actual State |
|---|---|
| s4 = Green | s4 = Green |
| s8 = Green | s8 = Green |
| Ta = Green | Ta = Green |
| Tb = Green | Tb = Green |
| Td = Green | Td = Green |
| w1 = Green | w1 = Green |
| Route 6 is called. | Route 6 is called. |

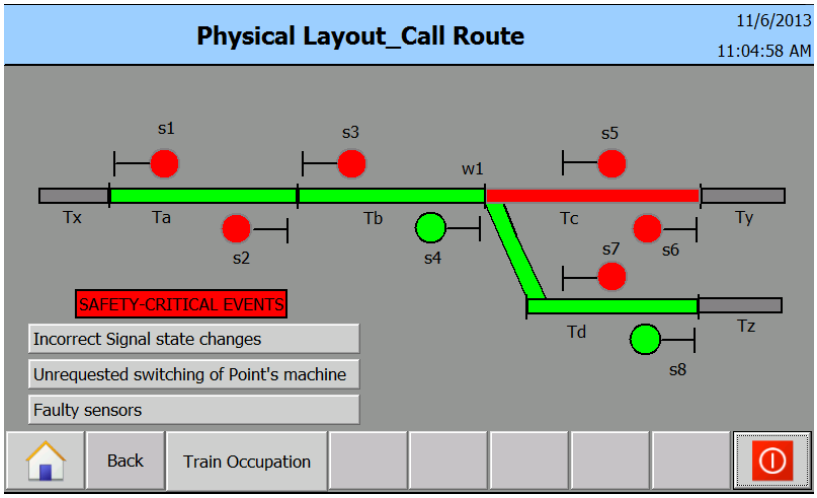


Figure E.9: Evaluation of route call for route 6

*6.3. Case 3: Train Occupation*

Table E.43: Route 6 – Train Occupation test case parameters

| Expected State | Actual State |
|---|---|
| s4 = Grey | s4 = Grey |
| s8 = Grey | s8 = Grey |
| Ta = Grey | Ta = Grey |
| Tb = Grey | Tb = Grey |
| Td = Grey | Td = Grey |
| w1 = Grey | w1 = Grey |
| Route 6 is cleared. | Route 6 is cleared. |

*6.4. Case 4: Route cannot be configured*

Table E.44: Route configuration parameters

| Element Configurations | Expected State | Actual State |
|---|---|---|
| s4 = 0 (Occupied) | s4 = Red | s4 = Red |
| s8 = 0 (Occupied) | s8 = Red | s8 = Red |
| Ta = 1 (Free) | Ta = Yellow | Ta = Yellow |
| Tb = 0 (Occupied) | Tb = Red | Tb = Red |
| Td = 1 (Free) | Td = Yellow | Td = Yellow |
| w1 = 1 (Free) | w1 = Yellow | w1 = Yellow |
| | Route 6 cannot be set. | Route 6 is not set. |

*6.5. Case 5: Initiate an element fault*

For Route 6, a fault has been initiated for signal element s8

Table E.45: Route request – Element Fault

| Element Configurations | Element State | Expected State | Actual State |
|---|---|---|---|
| s4 = 1 (Free) | s4 = 0 (Occupied) | s4 = Red | s4 = Red |
| s8 = 1 (Free) | s8 = 0 (Occupied) | s8 = Red | s8 = Red |
| Ta = 1 (Free) | Ta = 0 (Occupied) | Ta = Red | Ta = Red |
| Tb = 1 (Free) | Tb = 0 (Occupied) | Tb = Red | Tb = Red |
| Td = 1 (Free) | Td = 0 (Occupied) | Td = Red | Td = Red |
| w1 = 1 (Free) | w1 = 0 (Occupied) | w1 = Red | w1 = Red |
| | Fault triggered for element s8 | Route request is not issued. Route 6 cannot be set. | Cannot issue route request. Route 6 is not set. |

*6.6. Case 6: Set a route and initiate a safety-critical event*

After Route 6 has been set, the “Incorrect signal state changes” critical event is triggered.

Table E.46: Set route – Safety-critical event

| Expected State | Actual State |
|---|---|
| s4 = Red | s4 = Red |
| s8 = Red | s8 = Red |
| Ta = Red | Ta = Red |
| Tb = Red | Tb = Red |
| Td = Red | Td = Red |
| w1 = Red | w1 = Red |
| Element indications must change to red. Route must be cancelled. | All elements turn red. Route 6 is cancelled. |

*6.7. Case 7: Call a route and initiate a safety-critical event*

For this test case the "Unrequested switching of point's machine" event is initiated.

Table E.47: Call route – Safety-critical event

| Expected State | Actual State |
|---|---|
| s4 = Red | s4 = Red |
| s8 = Red | s8 = Red |
| Ta = Red | Ta = Red |
| Tb = Red | Tb = Red |
| Td = Red | Td = Red |
| w1 = Red | w1 = Red |
| Element indications turn red. Route must be cancelled. | All elements in layout turn red. Route 6 is cancelled. |

*6.8. Case 8: Train occupation with a safety-critical event initiated*

After the Train Occupation function has been executed, the "signal passed at danger" event is selected and the response of the interlocking is observed.

Table E.48: Train occupation – Safety-critical event

| **Expected State** | **Actual State** |
|---|---|
| s4 = Red | s4 = Red |
| s8 = Red | s8 = Red |
| Ta = Red | Ta = Red |
| Tb = Red | Tb = Red |
| Td = Red | Td = Red |
| w1 = Red | w1 = Red |
| Element indications turn from grey to red. Elements are set to occupied. | All signalling elements turn red. |