Modelling Methodologies for Railway Asset Management

Robert Henry Lee

A Doctoral Thesis

Thesis submitted to the University of Nottingham for the degree of Doctor of Philosophy

August 2020

Abstract

Management of railway assets incurs significant expenditure. Railway asset management modelling can predict the cost and efficacy of an asset management plan, and thus support the asset management planning process. Modelling frameworks can be used to facilitate the development of large, multi-asset, whole life cycle models which can be used to represent large sections of rail track and associated assets. This is achieved with libraries of models and tools with a high level of inter-compatibility.

This research set out to support the development of modelling frameworks for railway asset management. It sought to determine the state of the art of railway asset management modelling in order to find which assets require further modelling development before they can be suitably represented in a framework's model library. It also sought to determine the most accurate and suitable modelling methodology to base the framework upon.

These aims were met by first carrying out a literature review to determine the state of the art of asset management modelling for major railway asset types. This review found Petri net models solved via Monte Carlo methods to be the most suitable modelling methodology for asset management. The level crossing asset class was chosen for the development of several models to explore the different types of Petri net model, concentrating on the computational resources required. This asset class was chosen as no asset management model was found in literature, and the diversity of the asset interactions.

Literature review found several asset classes in need of further development, and some where asset management modelling may not be possible without other advances. The level crossing Petri net models developed demonstrated that computational requirements differ between the various types of Petri net. Stochastic Petri nets were found to simulate quickly, but had a high memory requirement. Coloured Petri nets were found to have the opposite requirements. A novel Petri net type, the Simple Coloured Petri net was developed to create a balance in computational cost. It was further found that complex processes such as scheduling and resource allocation can only be carried out using Coloured Petri nets due to their enhanced feature set.

This work has found that further research on modelling specific asset classes is required to enable the development of a complete asset modelling library for use in a framework. If large models are to be developed, it is recommended that the Simple Coloured Petri net be used to balance computational requirements. Any models requiring complex functions should be developed using the Coloured Petri net methodology.

Acknowledgements

I would like to acknowledge everyone at the Resilience Engineering Research Group for their help and support throughout my PhD studies. In particular I wish to thank Professor John Andrews for this guidance and patience whilst I conducted this work and developed my research skills.

Thanks are given to Network Rail and the Lloyds Register Foundation for their generosity in funding this research.

Lastly I wish to thank Katrina and Phil for supporting me whilst I finalised this thesis.

Abbreviations

ABCL	Automatic Barrier Crossing Locally Monitored
AHBC	Automatic Half Barrier Crossing
ALARP	As Low as Reasonably Practicable
AOC	Automatic Open Crossing
CCTV	Closed Circuit Television
CDF	Cumulative Distribution Function
CPN	Coloured Petri Net
ETTE	Electric Traction Transmission Equipment
HGV	Heavy Goods Vehicle
Kph	Kilometres per hour
LED	Light Emitting Diode
LiDAR	Light Detection and Ranging
m/s	meters per second
Mph	Miles per hour
OLE	Overhead Line Equipment
SCPN	Simple Coloured Petri Net
SPN	Stochastic Petri Net
TPN	Timed Petri Net

Table of Contents

1. Introduction

UK railways are responsible for 1.7 billion passenger journeys each year, covering 65.6 billion kilometres (Office of Rail and Road, 2018), and move 75million tonnes of freight each year (Office of Rail and Road, 2017). The UKs railways are not profitable and are dependent on government grants to continue to operate, this creates financial pressure towards minimising expenditure. This can conflict with the need to provide a safe and reliable service. Care must be taken when deciding where and how limited budgets will be expended managing railway infrastructure assets to ensure the highest level of safety and service provision.

1.1 Railway Systems

The UK's railway comprises a diverse range of infrastructure assets, these assets were constructed in two main phases. Many of the assets in use today date back to the Victorian era when the railway was first constructed, a second major phase of railway construction occurred in the 1950s when the railway was modernised. As a consequence, many railway assets in use today are aging. This, in combination with their volume makes operating the UK's railway highly maintenance intensive. Below, each infrastructure asset class will be briefly outlined.

Track: Track comprises the rails, sleepers and ballast which support and guide trains, in the UK there are 15,811 route kilometres of rail track which are typically renewed as needed, though significant amounts date back to the 1970s (Office of Rail and Road, 2017).

Switches and Crossings: These are mechanical devices which allow trains to change tracks at junctions via the use of moveable rails. There are around 18,000 in use on the UK railway, with many dating from 1970 and the late 2000s following modernisation efforts.

Earthworks: Trains are only able to climb or descend shallow gradients, this frequently necessitates changing the ground level to meet this constraint by either removing or depositing soil, collectively known as earthworks. In the UK there are some 9,933km of embankments where soil is deposited, 7,015km of cuttings where soil is removed, and 1,507km of rock slopes adjacent to track. Most of these earthworks date back to the original construction of the railway around 150 years ago (Network Rail, 2014).

Bridges: Frequently deployed on the UK railway to allow rail track to pass over roads or bodies of water, known as an underline bridge, or to allow roads to pass over rail track known

as an overline bridge. There are 19,367 underline bridges, and 9,132 overline bridges in use, most which are between 100 and 180 years old (Network Rail, 2014).

Tunnels: When terrain is steep it can be more cost effective to tunnel through rather than dig down leading to the creation of rail tunnels. There are 625 tunnels in use on the UK rail network, most of which date back to when the railway was first constructed around 150 years ago (Network Rail, 2014).

Drainage: Drainage infrastructure is deployed on railways to both prevent water ponding on sensitive areas such as track, but also to improve the stability of earthworks which may be damaged by saturation or the flow of water. There are 247,711 individual drainage assets on the UK railway of varying ages, though generally track drainage assets are older than 70years, and earthworks drainage assets are newer (Network Rail, 2012).

Electric Traction: These assets supply electricity along rail track facilitating the use of electric trains either through overhead line equipment or third rails. Approximately 34% of rail track in the UK is electrified, the average electric traction asset is 30 years old (Network Rail, 2012).

Signalling: Signals are vital to the safe operation of a high-capacity railway, performing the critical task of informing drivers if the proceeding track section is occupied and, in many instances, automatically applying the train's brakes if the driver does not respond. There are approximately 500,000 individual signalling assets on the UK railway, with component ages evenly distributed between newly installed and 50 years old (Network Rail, 2012).

Level Crossings: Level crossings provide an interface between road and rail, allowing pedestrians and road vehicles to cross rail track. These can be protected crossings using barriers and lights controlling access, or passive where users must visually confirm if it is safe to cross. There were 720 automatic level crossings, 813 staff operated crossings and 5,144 passive level crossings in use in the UK (Network Rail, 2012).

The aging nature of the UKs railway infrastructure assets poses difficulties in management that are not encountered with newer assets, in addition to the increased maintenance requirement previously mentioned: there may be problems associated with obsolescence where spare parts or knowledge required are no longer available; older assets may no longer meet new design, safety or service provision requirements; information on the construction and maintenance of an asset may have been lost, complicating inspections and assessment. This adds further complexity to maintaining the UK's railway effectively, without violating financial, service or safety related constraints.

1.2 Railway Safety

Safety on the UKs rail network is of paramount importance, guided by the principle that risk of injury to all persons should be kept as low as reasonably practicable (ALARP), meaning that all methods to reduce risk of injury must be implemented when both effective and feasible (RSSB, 2002). Many types of accidents occur on UK railways, the most serious typically occur when, due to collision or other reasons, a train's wheels leave the confines of the rails. This is known as a derailment, notable examples of accidents where derailments occurred include:

- Southall crash, September 1997, due to poor communication between signalling staff and driver inattention a passenger train collided with a freight train derailing both, injuring 139 and causing 7 deaths (Uff, 2000).
- Ladbroke Grove rail accident, October 1999, two trains collided when a driver miss-read a signal resulting in 31 deaths and 523 injuries (Cullen, 2001).
- Great Heck rail crash, February of 2001, where a high-speed train collided with a road vehicle which came to rest on a rail line after the driver lost control of the vehicle. There were 82 injuries and 10 fatalities (Health and Safety Executive, 2002).
- Hatfield rail crash, October 2000, a high-speed train derailed on route to Leeds due to a fractured rail, injuring 70 and causing the death of 4. This accident precipitated the collapse of Railtrack PLC discussed in the next section (Office of Rail Regulation, 2006).
- Potters Bar, May 2002, a passenger train derailed just south of Potters Bar station due to a points failure, causing one of the carriages to collide with the structure of the station, injuring 76 and killing 7, resulting in Network Rail changing its management practices towards external contractors, also discussed further in the next section (Office of Rail Regulation, 2003).

Derailments are associated with such high numbers of injuries and fatalities due to both the loss of control that occurs when a train derails, and that derailments typically occur during high energy collisions. Fire is a serious risk to the safety of rail users, but fortunately is rare, and usually confined to underground railways, examples include:

- King's Cross fire, November 1987, a fire started in a wooden escalator and rapidly spread throughout the station injuring 100 people and causing 31 deaths, prompting far greater fire awareness and planning at UK stations (Fennell, 1988).

- Channel Tunnel fire, September 2008, a fire broke out within the channel tunnel started by a lorry on a freight train, causing minor injuries (Santo-Reyes & Beard, 2016).

Accidents due to trains colliding with people on the railway are much more common, this can be due to trespass or a work-related injury. Between 2017-2018 there were 2 workforce fatalities on the railway, and 27 due to trespass (RSSB, 2018). Serious injury and loss of life can result from the failure or collapse of the various structures in use on a railway, such as bridges, tunnels or stations, notable and recent occurrences detailed below:

- Tay Bridge disaster, December 1879, one of the most well-known structural failures on the UK railway occurred when the Tay Rail Bridge collapsed due to inadequate design against wind loading, killing 75 (Rothery, 1879).
- Penmamshiel tunnel collapse, 1979, during an upgrade of the tunnel to allow larger freight containers to pass through the tunnel roof collapsed whilst 15 workers were inside, 2 were killed in the collapse (Department of Transport, 1983).
- Glanrhyd Bridge collapse, October 1987, a rail bridge over the river Towy was partially swept away overnight during heavy rainfall. A passenger train left the tracks passing over the bridge falling into the river below resulting in 4 deaths and changes in operational practice to prevent similar incidents (Department of Transport, 1990).

These incidents highlight the variety and severity of the potential hazards inherent to operating a railway. Many of accidents listed above were due to events what were not well understood at the time of occurrence and resulted in changes to railway operation and management to prevent future occurrences.

1.3 Railway Maintenance Expenditure

As a whole the UK railway does not operate at a profit, for 2016-17 the income for the entirety of the UK rail industry was £19.0 billion including government grants, and the expenditure £19.5 billion (Office of Rail and Road, 2018). Of the expenditure, £1.3 billion was spent on maintaining rail assets and £2.7 billion spent on replacing rail assets. This highlights the cost of operating the UK railway, and the financial constraints placed upon tasks such as maintenance. Figure 1-1 breaks down renewal spending by asset class.

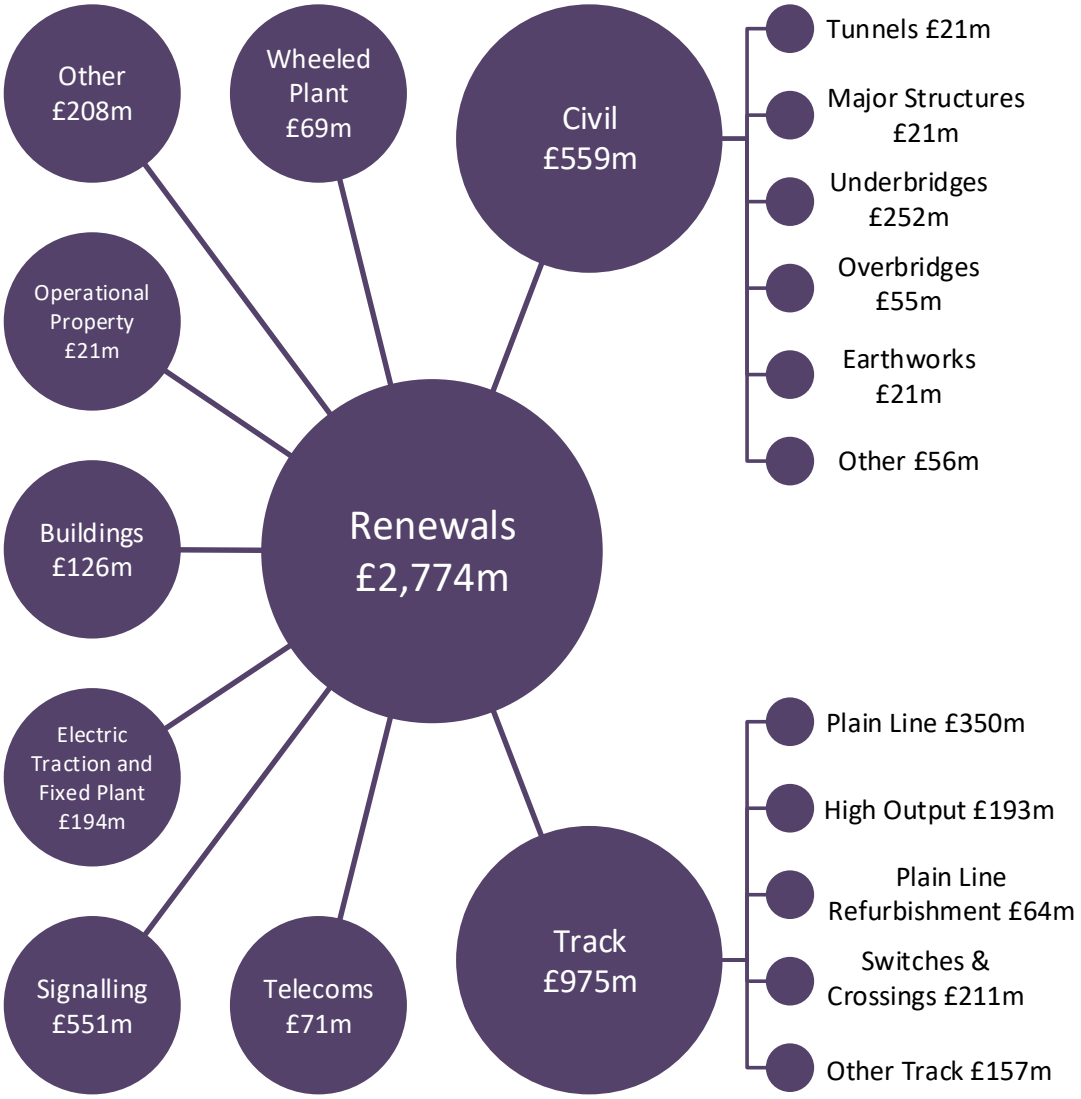

Figure 1-1 Diagram Showing UK Railway Renewal Expenditure 2016-17

Figure 1-2 shows maintenance spending by asset class.

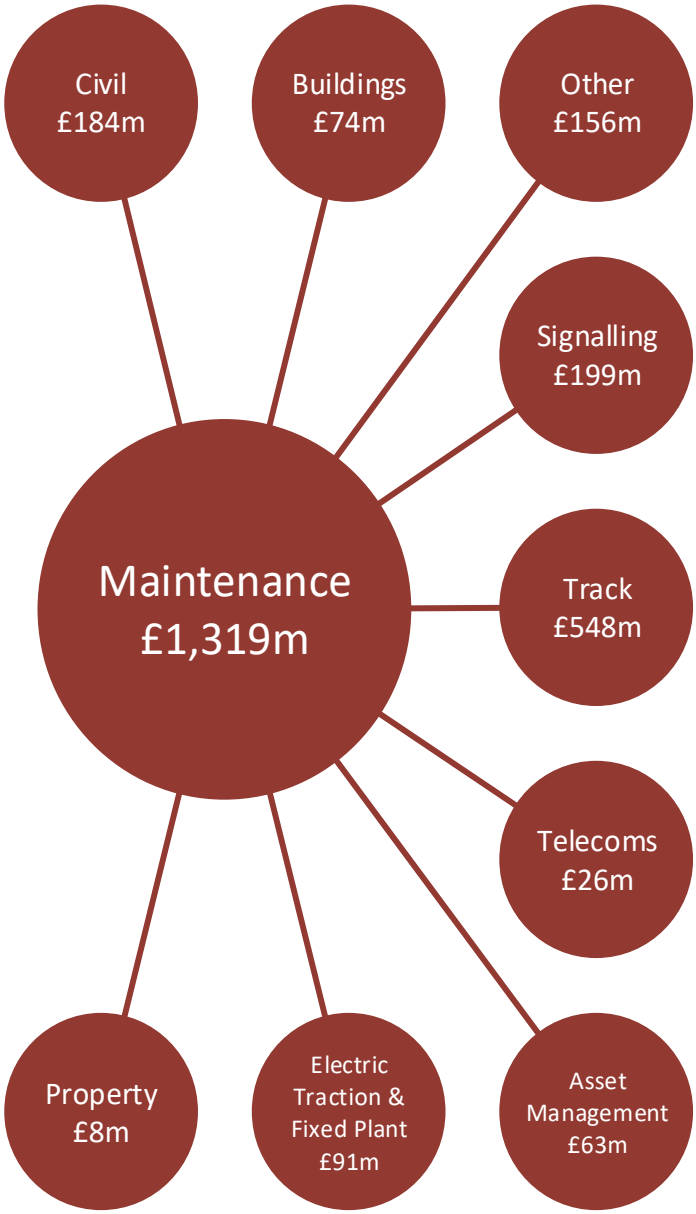

Figure 1-2 Diagram Showing UK Railway Maintenance Expenditure 2016-17

Track: Rail track has the highest expenditure of any asset class. There are typically 700,000 minutes of train delays on the UK railway every year due to track failures, of which on average each costs £5,000 due to the delay fines alone. The Hatfield rail accident mentioned earlier occurred due to a fractured rail as a consequence of inadequate maintenance (Office of Rail Regulation, 2006). This accident precipitated the collapse of Railtrack PLC. It caused millions of minutes of train delays from emergency speed restrictions whilst remedial work was carried out renewing rail track now considered unsafe, with the cost estimated at £580million.

Switches and Crossings: Potters Bar occurred two years after the Hatfield accident due to a point's failure. It was found to have occurred due to inadequate maintenance by an external

contractor employed by Network Rail, prompting them to perform all track maintenance in-house, and incurring a £3m fine for its negligence alone.

Earthworks: for the 2016-17 fiscal year Network Rail spent £21m renewing earthworks. Unit costs will vary on a case by case basis, but have been estimated at £3-5,000 per meter of reconstruction but only £1-2,000 per meter of corrective action (McGinnity, et al., 1998), highlighting the potential cost savings from taking early action to repair at risk earthworks.

Bridges: There are a large number of bridges in use on the UK rail network, associated with significant maintenance and renewal costs. In 2009 a masonry bridge over a canal near Feltham partially collapsed due to the scouring action of the water flowing beneath. Prior to failure the bridge was inspected but the staff attending did not have sufficient experience or training to discover the failure in progress. As a consequence of this Network Rail had to close the line over the bridge whilst the bridge was repaired at a cost of several million pounds. This would otherwise have been repairable at a substantially reduced cost (RAIB, 2010).

Drainage: Drainage failures are often associated with much more costly and disruptive track and earthworks failures. An example of this occurred November 2009 when a blockage in a drainage ditch triggered the collapse of a cutting on the eastern side of the Gillingham tunnel in Dorset. (RAIB, 2010).

Level Crossings: Level crossings are a significant hazard and source of disruption to the UK railway. It is an area where upgrades and expenditure can be very clearly linked to a hazard reduction. There are between 4 and 12 fatalities on UK crossings every year (RSSB, 2017). Automatic level crossings are associated with the highest accident rate of any crossing type. With investment they can be replaced with more robust crossing types with demonstrably lower accident rates.

The examples presented here highlight the many routes through which maintenance activities can affect overall costs and safety on the railway. The engineering field devoted to problems of this nature is known as asset management, in a railway context its purpose is to minimise costs whilst maximising safety and service reliability/provision. It achieves this aim through the gathering and processing of information, used to inform infrastructure asset managers on how to allocate their limited resources to achieve the maximum benefit.

1.4 Asset Management

Asset management strategy describes an organisation's asset management objectives at a high level, for Network Rail this is to maintain their asset base at its current condition level, improving it where possible, whilst keeping to the ALARP principle defined earlier and managing financial constraints. This then governs the development and implementation of the asset management policy for each asset class which relates Network Rail's strategic aims to the needs of a specific asset class. Finally this is used to create an asset management plan which governs the many maintenance and management operations defined below:

Servicing: Servicing is performing some task which reduces or prevents unnecessary wear to an asset in order to maximise its working lifespan. Assets such as rolling stock may be highly dependent on servicing to prevent premature failures whereas other assets such as signalling have few to no effective servicing options.

Inspection: This process discovers and monitors asset degradation, commonly comprising of a visual inspection of an asset, though it can include automated inspection and reporting systems. Discovery of a degraded or failed railway asset allows repair processes to be initiated and hazard mitigation measures to be implemented. Prompt discovery and remediation of a minor issue may prevent progression to more serious one as highlighted in the example of the Feltham bridge collapse or Gillingham cutting failure described earlier.

Preventative Maintenance: This is a maintenance action carried out due to an expectation that an asset or component is likely to fail in the near future. This could be based on the age of the asset, the level of use it has had or some aspect of its condition. Though it is an expense, as it is scrapping a still functional asset, it mitigates the potentially greater expense that would be incurred if the asset were to fail in use, and any hazards that would arise due to its failure. As an example, had the rails which failed triggering the Hatfield rail accident been replaced prior to their failure, it would have been dramatically less costly in terms of both expense and human life.

Reactive Repair: a reactive repair is a maintenance action carried out because an asset or rail component has failed. Typically it is much more expensive to repair an asset reactively as little prior planning or scheduling can occur. There may be expenses related to delayed or cancelled services. However, there are assets which can only be repaired in this manner due to there being no practical measure by which to determine if failure is likely or imminent. An example of this is accidental damage to buried signalling cables during excavation works.

Renewal: A renewal is a replacement of an asset or component at the end of its useful life with a new one. There are several motivations to carry out such an action which mitigate the expense of replacement: the asset may be obsolete and no longer practical to operate due to limited availability of spare parts or an incompatibility with modern equipment or practises; the asset may be in poor condition with refurbishment either impractical or too costly compared to simply replacing the asset; the asset may have a history of requiring excessive maintenance; the asset may have insufficient performance or capacity.

Refurbishment: a refurbishment is a substantial maintenance undertaking applied to an asset which aims to improve a degraded condition to prevent further degradation and effectively extend the life of the asset. Many assets can survive and function indefinitely via refurbishments, an example of this is masonry structures which require periodic repointing and replacement of damaged bricks but there is no limit on the number of times this action can be repeated.

Opportunistic Maintenance/Renewals: This is a maintenance or renewal action carried out because an opportunity arose to do so at significantly reduced cost or earlier than planned. This may occur when work is performed nearby so the cost of mobilising equipment/resources has already been incurred or when a possession to work on another asset occurs.

When setting the asset management plan for a rail asset all of the above processes should be considered in terms of how they further a strategic aim such as to keep safety risk to a minimum whilst keeping to a limited budget. For a highly critical asset such as the rails which comprise rail track the budget could be expended on frequent, thorough inspections which aim to discover failed states promptly allowing reactive repairs to be carried out. It could also be achieved via preventative maintenance, where the rails are replaced at a specified condition, expending the limited budget, but keeping the probability of failure low.

The variety of maintenance processes available create a system with many degrees of freedom. For any given asset there are multiple asset management policies or plans which can achieve the same strategic aim. These plans will differ in their effectiveness at achieving the strategic aims, and they will differ in the quantity and nature of the resources they require. The level of experience possessed by the decision maker can help steer them towards an optimal plan but owing to the complexity and uncertain nature of infrastructure management analytical models are best employed to guide the decision-making processes effectively.

1.5 Asset Management Modelling

Models are an abstraction, where a complex system is represented by a simpler model for the purposes of human understanding and analysis. Asset management modelling is primarily concerned with mathematical modelling, where mathematical language and functions are used to describe and emulate real world infrastructure processes. This requires an appropriate level of abstraction to enable the real-world system to be represented mathematically.

Degradation models predict the deterioration of an asset over time, supporting decision making by providing predictions of asset state changes over time, or usage. Early examples of this include deterministic ballast degradation models created from laboratory testing data (Alva-Hurtado & Selig, 1981). Contemporary models have used the Markov modelling methodology extensively to model bridge deteriorations (Cesare, et al., 1992; Ng & Moses, 1998; Morcous, 2006) and have seen application in track deterioration modelling (Prescott & Andrews, 2013).

Hazard models for railways have seen less attention historically but with the occurrence of high-profile accidents are gaining more attention. These model the occurrence and effects of hazards to estimate metrics related to safety. They have been applied to level crossing assets which historically have a high accident rate. This began with the development of non-linear mathematical functions relating crossing usage to accident rates (Stott, 1987) progressing to contemporary level crossing risk models using Petri nets (Medjoudj & Yim, 2007; Ghazel, 2009). Network Rail use their own hazard model to assess risk at level crossings to support decisions on crossing management (RSSB, 2010).

Maintenance models have seen recent development to help manage rising railway maintenance expenditure. Such models may be used to predict the effectiveness or determine optimal scheduling of maintenance actions. A paper by Audley and Andrews modelled the effectiveness of repeated tamping interventions on railway ballast degradation (Audley & Andrews, 2013). Zhao et al created a mathematical model to predict track maintenance costs which considered the cost savings from opportunistic renewals (Zhao, et al., 2009).

Models such as these consider specific aspects of a railway assets largely in isolation, limiting their effectiveness at predicting broad measures of performance which are affected by multiple interacting processes. Recently, integrated models have been developed which combine previously independent models of degradation and maintenance enabling whole

asset – whole life modelling to occur. Examples of such work include bridge asset models for steel bridge elements (Yianni, et al., 2016) and concrete bridge elements (Le & Andrews, 2015) which integrate both degradation and maintenance enabling the prediction of whole asset system costs over its lifespan.

The integration of maintenance and degradation models removes the need for broad assumptions about either of these linked processes. This facilitates the application of asset management models to move from prediction to optimisation. An integrated model utilises many variables representing an asset management plan, each of which may have a significant effect on the predicted performance of the asset. Methodical exploration of these variables can be used to search for an optimal asset management plan. This may provide reduced costs or increased performance, depending on the target of the optimisation, relative to other potential plans.

1.5.1 Modelling Frameworks

Integrated models are comprised of multiple distinct models whose coherent interaction are critical to achieving their intended purpose. Frameworks describe the data flow and relationships between constituent models and tools which comprise the integrated model. This provides a coherent structure which can be used to produce integrated models from simpler models in a reproducible manner.

Frameworks can be employed in the creation of new models by adopting the specified data flow and structure. This ensures new models have some level of compatibility with pre-existing models which conform to the framework, allowing incorporation of effective elements of older models into new. This reduces development time for new models, as common elements may not need to be recreated. Established frameworks may be associated with large libraries of mature tools and models enabling the modelling process to focus on novel aspects. Frameworks often do not specify techniques or methods, as a technical standard might. Their purpose is to guide and support modelling development, rather than ensure replication. This balances development support, with the degrees of freedom needed to create new models and improve upon the current state of the art (Argent, et al., 2006).

In general, using a framework reduces the effort required to develop a model. This can be used to develop models faster, or enable the development of larger models. This is of particular importance to railway asset management modelling when developing whole asset models. As many different interacting asset classes comprise a railway, modelling a complete

section of a railway is a complex task. This can be simplified with the use of modelling frameworks.

1.5.2 Innovative Intelligent Rail Framework

The Innovative Intelligent Rail Project (In2Rail) is an EU project which aims to facilitate the development and integration of advanced rail technologies and processes into the European rail network to meet expected growth within the transport sector. Work Package 6, Smart Infrastructure – Maintenance Strategies and Execution, developed a framework for developing railway asset management models. This is intended to support the progression of railway asset management modelling from individualistic narrow scope models which focus on specific assets to integrated models capable of modelling multiple interacting assets within the same model.

The framework outlines the development of integrated asset models from three constituent models: degradation; maintenance; and functional models. A network topology model provides information to the integrated asset model relating to local asset conditions to enable whole life cost modelling for specific assets. The integrated asset model and network topology model both provide processed data to a risk model, and a service provision model. These relationships are shown in Figure 1-3 and all described in more detail later. Raw and processed data alike are to be made available in libraries, along with the models developed. This is to enable re-use of previously developed models or data, either when creating new asset level models, or to allow asset level models to be integrated to form network level models.

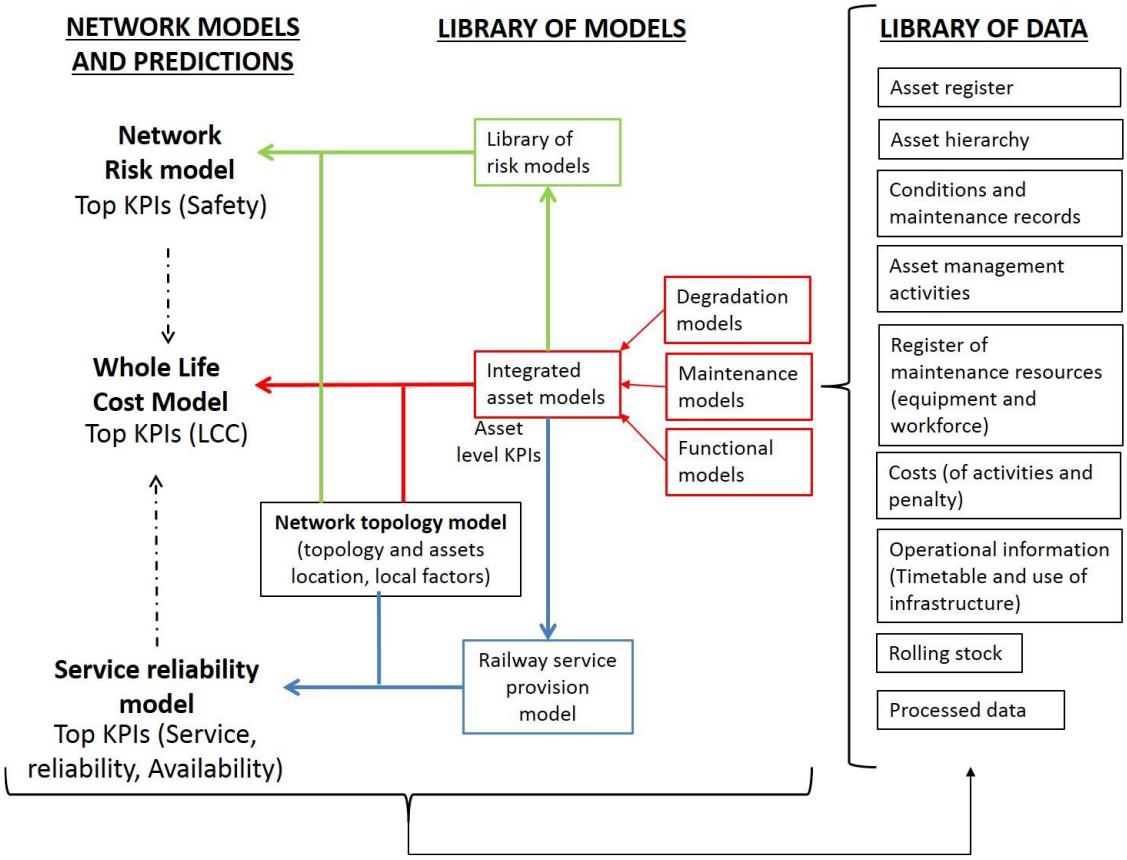

Figure 1-3 Schematic Map of the Modelling Framework (Fecarotti, et al., 2018)

1.5.2.1 Whole Life Cost Modelling Development Framework

The purpose of a whole life cost asset management model is to predict the level of expense incurred over the life of an asset, as a function of its asset management plan and strategy. The modelling framework visualises how data flows between the constituent tools/models to support the development of a whole life cost model. The framework comprises five core elements: degradation modelling; maintenance modelling; an integrated asset model; a cost analysis tool; and a whole life cost predictor. Figure 1-4 displays a flow chart showing the data flow for the framework.

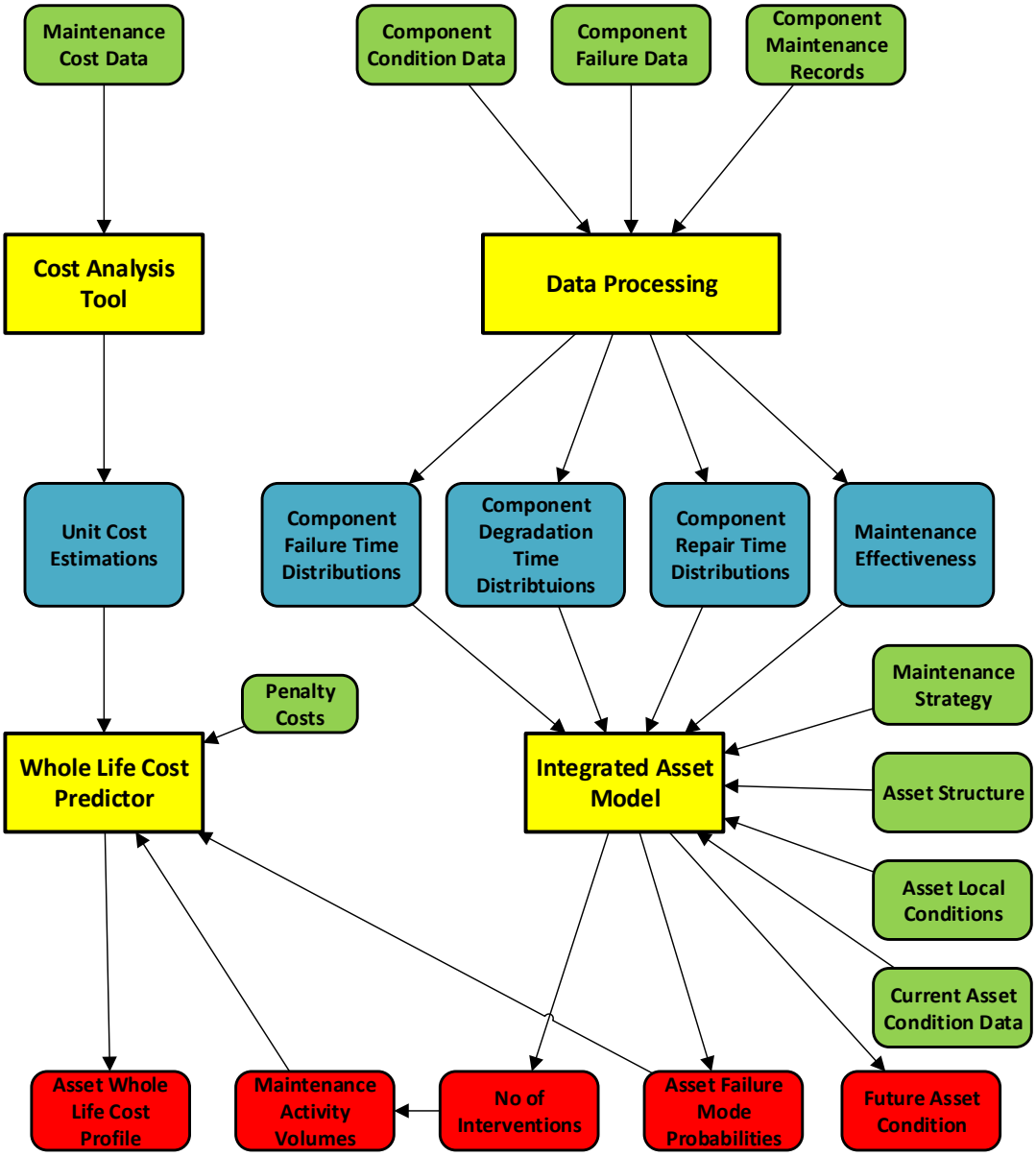

Figure 1-4 Innovative Intelligent Rail Asset Management Modelling Framework (Fecarotti, et al., 2018). Yellow shows analysis / modelling, green shows inputs, blue shows intermediate data inputs, and red shows predictions.

As the name implies, the degradation model models how the asset degrades over time, incorporating the influence of local environmental factors and current asset state on its rate of degradation. To achieve this, statistical analysis is carried out on component condition and failure data to produce the statistical parameters the degradation model requires. The distribution types for the degradation model are not specified, only that they are a function of time. The appropriateness of any given distribution will depend on the asset it is applied to and the condition data available. Similarly, the methodology underlying the degradation model and its structure is not specified.

The maintenance models cover two aspects of the maintenance process, scheduling and its effectiveness. Scheduling and time to repair is specified as stochastic process, with distributions and parameters determined following analysis of component maintenance records. Maintenance effectiveness is determined from condition records concurrent with maintenance records. Condition records will show some change in condition following maintenance, which can be used to estimate the effectiveness of any given maintenance activity. The framework does not specify if this should be stochastic, deterministic or some combination of both. Functional models represent the activities and operations comprising a maintenance action. The outputs of both the degradation and maintenance models, and the models themselves are stored in libraries for use as required in any subsequent modelling activity.

The maintenance and degradation models are combined to form the integrated asset model. This directly models the interaction between degradation states and maintenance. Management aspects such as inspections are modelled to discover degraded states; degraded states trigger maintenance actions such as renewals or repairs; servicing attenuates the progress of degradation, all governed by the maintenance strategy input to the model. This enables the model to produce a prediction of the number and type of interventions needed, which can be used later to estimate a cost profile over the life of the asset. It also enables the model to predict how the asset condition evolves over time.

The whole life cost profile is produced using a cost analysis tool, this analyses data containing maintenance costs to determine unit costs of maintenance activities. These costs are then used with the maintenance activity volumes predicted using the integrated model to determine life time costs due to maintenance. The unit cost of any penalties is used alongside the probability of asset failure, also predicted by the integrated asset model, to determine the number and scale of any poor performance related penalties levied.

Each asset modelled within the framework is created using an asset model template. This template is configured based on the asset's structure, maintenance plan, current condition state and environmental conditions to create a unique asset model. Despite the common model template, this leads to different model behaviour and outcomes, enabling a single asset model template to be used to create models for different assets that share the same core functions and features. An asset model template may be designed for a highly specific type of asset such as an Automatic Half Barrier Crossing, but could also be designed to be used with a wider range of assets.

This framework adopts a bottom-up approach to model development as separate degradation and maintenance models are developed prior to integrated models. Doing so enables the degradation and maintenance models to be verified and assessed as to how effectively they replicate the data used to create their structure and governing parameters. It also enables the models to be applied and refined in use prior to the development of the integrated asset model. This is valuable as it is difficult to assess the integrated asset models themselves as they model the emergent effects of changing asset management plans, for which there is no prior experience.

1.5.2.2 Service Provision Model Development Framework

Railways typically allocate their service by either employing a timetable outlining the specific arrival and departure times, or by specifying an arrival interval for their trains. Service provision of a timetabled serviced can be measured in terms of trains delayed or cancelled. Services may be cancelled or delayed due to asset failures, or restrictions applied on asset usage to preserve safety such as temporary speed restrictions. The occurrence of these events is in turn dependent on the asset management strategies employed across the asset base. The In2Rail framework outlines a service prediction model which creates predictions as a function of the asset failure probabilities generated by the integrated asset model.

In order to predict the impact of asset failure on service prevision, a range of information is required. A network topology model will specify the relative locations of assets on the network, along with rules relating to their operation. Traffic management rules are required in a service provision model. These specify how any spare capacity within the network can be used to counteract the effects of asset failure. They specify which services are prioritised when this additional network capacity is apportioned. The service provision model itself must know how these aspects interact to generate predictions, which may then be used to evaluate a given asset management strategy's effect on service provision. The data flow is summarised in Figure 1-5.

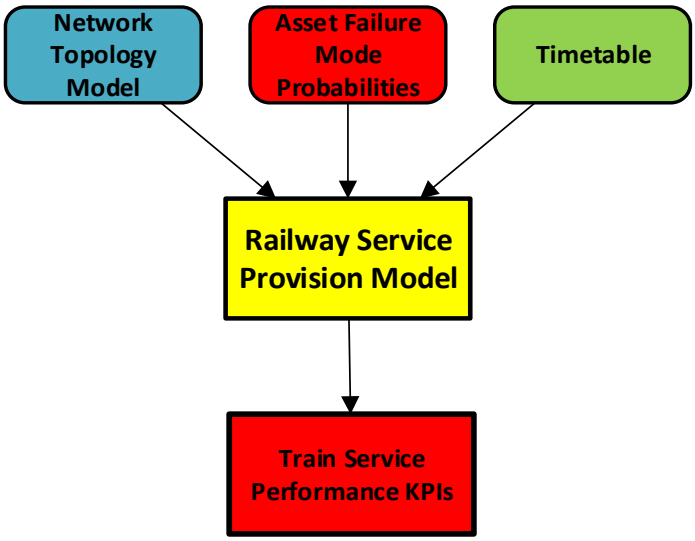

Figure 1-5 Service Provision Model. Yellow shows analysis / modelling, green shows inputs, blue shows intermediate data inputs, and red shows predictions.

1.5.2.3 Risk Model Development Framework

The final aspect of the In2Rail modelling development framework concerns risk modelling. It incorporates traditional risk modelling approaches which model the occurrence of a hazardous event separately to the consequences of a hazardous event to predict risk. The probabilities for the asset failure events which are responsible for the occurrence of hazardous events are provided by the integrated asset model. This is summarised in Figure 1-6.

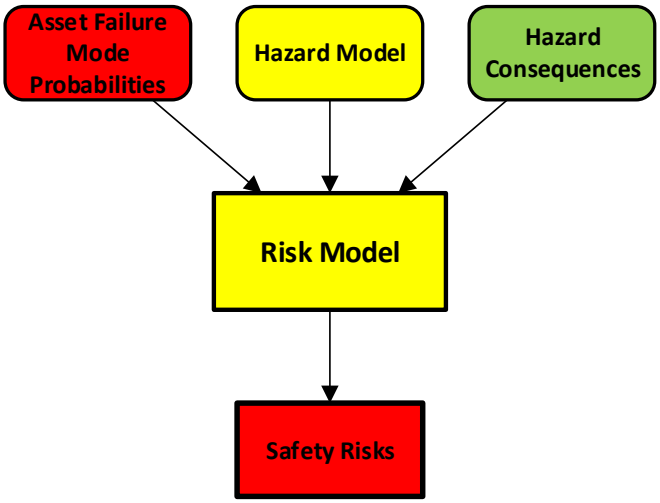

Figure 1-6 Risk Model Development Framework. Yellow shows analysis / modelling, green shows inputs, and red shows predictions.

1.5.3 Frameworks Summary

This chapter has described how frameworks can be employed to support the development of the state of the art asset management modelling. They achieve this by enabling the independent development of models which share a coherent data structure. This coherence allows effective and accepted modelling elements to be re-used, freeing development time to be spent on novel aspects.

The In2Rail asset management modelling framework was described, outlining how it supports the development of whole asset – whole life modelling, over multiple levels of abstraction. It achieved this by specifying the modelling data flow, whilst staying largely agnostic of the modelling methodologies required. Specific data elements where specified to be parameters of non-specified probabilistic distributions, implying a stochastic modelling approach should be employed to perform certain tasks.

Based on the current state of the art, it is not known which modelling methodology is best suited for use in a modelling framework. Monte Carlo analysis of models has become popular in academic literature owing to its improved accuracy over Markov models when modelling rule-based maintenance plans and decision making. However, authors have reported computational difficulties when analysing large multi asset models using a Monte Carlo Method. These computational difficulties will limit the size and scope of analysis of models developed using a framework.

1.6 Aims and Objectives

This research project aims to employ an asset management framework, to enable decision making on a whole system, whole life basis for each railway infrastructure asset class. This entails reviewing the methods developed and used to model asset management for each asset class, to determine their efficacy and ultimately if these models are sufficient to meet the needs of the asset class.

It aims to explore the computational requirements of different modelling methodologies when applied to modelling large multi asset models, highlighting the most computationally efficient. This will maximise the number and complexity of assets models a framework derived model may include, granting greater freedom when using a modelling framework.

Project objectives are summarised below:

1. Perform a literature review to investigate each asset class and establish methods available to model the asset condition, review the strengths and weaknesses and establish the most effective method.
2. For an asset where no adequate models exists, develop an appropriate model to fulfil the asset management requirements.
3. Establish the best modelling techniques for use in railway asset management modelling frameworks.

1.7 Thesis Outline

The first chapter of this thesis outlines the key concepts and motivation for applying modelling frameworks to railway infrastructure asset management. The second chapter reviews asset management decision support modelling methodologies, highlighting their general capabilities and weaknesses. This supports the third chapter, which reviews the state of the art of asset management modelling of each major type of railway infrastructure asset. This highlights aspects where additional modelling development may be beneficial to the aim of enabling whole system-whole life modelling.

The third chapter highlights level crossings as an asset lacking degradation or maintenance models. Chapters 4 – 8 outline the development of an integrated asset model for level crossings. The fourth chapter reviews level crossing systems. The fifth chapter develops a risk model for a specific class of level crossing. The sixth chapter presents models used to analyse the level crossing class's electrical and mechanical systems. The seventh chapter presents several asset management models for the level crossing class and reviews their suitability. Chapter 8, the final modelling chapter presents a model to enable resource sharing between railway assets within the model.

The final chapter of the thesis presents a summary, conclusions and potential further work.

2. Asset Management Decision Support Methods

In the rail industry, finances are constrained so asset management decisions must be made carefully in order to expend limited budgets effectively. Infrastructure asset management comprises many distinct interventions such as: inspections, servicing, repairs & renewals. Each of these interventions have their own financial and resource costs, often dependent on conditional factors such as asset type, location and availability of staff and plant.

Decision support tools exist to provide quantification of different asset management plans by estimating their costs and effects on the asset base. This enables a more thorough comparison between plans, allowing the decision making process to move beyond simply relying on the experience of the decision maker. In this chapter the most common methodologies underlying asset management decision support tools will be outlined, in advance of discussion of their application in the following chapter.

2.1 Markov Modelling

All Markov models assume that the future state of the model is affected only by the current state of the model, such that the model is memoryless, this is known as the Markov Property, represented formally below, equation (2-1) (Norris, 1997).

$$P(X_n = x_n | X_{n-1} = x_{n-1}, \dots, X_0 = x_0) = P(X_n = x_n | X_{n-1} = x_{n-1}) \quad (2\text{-}1)$$

2.1.1 Markov Chain

Markov chains are a stochastic model comprising a system of states. In infrastructure asset management those states typically refer to the degradation or operational state of an asset. Changes from one state to another are governed by transitions.

Considering only discrete state space Markov chains, there are two categories. Discrete time Markov chains are the simplest. The transitions between states are the probabilities of the state changing from one to other per the elapsed time in each time step. These can be formed into an n by n matrix where n is the number of states in the model, with the probability of each state changing into each other state input. Known as a Transition Probability Matrix (TPM). The state of the system at any discrete time step can then be evaluated by the product of the initial system state and the TPM raised to the power of the number time steps (Cinlar, 1975).

The continuous Markov chain allows evaluation of the system state at any given time, providing an advantage over the discrete Markov chain. As per the other, it comprises discrete states and transitions between them. These transitions are now expressed as rates, and formulated into a Transition Rate Matrix (TRM). Unlike the simple solution of the discrete time variant, this requires the formulation and solution of a set of differential equations.

Commonly understood limitations of Markov chains are outlined below:

Constant Probability of State Change: In keeping with the Markov property, all transition probabilities or rates must be constant with time. Therefore to use a Markov chain model it must be assumed that the states changes for a given asset occur with constant probability. This can be negated somewhat by increasing the number of states in a Markov chain model allowing finer control of the degradation rate of an asset over its whole life but increasing computational requirements.

State Space Explosion: For every extra state, the transition probability matrix increases in size by one additional row and column, creating an exponential increase in complexity when performing the matrix operation to obtain future state probabilities. This creates an effective limit on the number of permissible states in a Markov chain model. This limits its application to complex systems that contain large numbers of states.

2.1.2 Semi-Markov

Markov chains are limited by the Markov property, an assumption of memorylessness. This contradicts observations that degradation rates can be dependent on age for railway infrastructure (Sobanjo, 2009). The semi-Markov process model partially relaxes this requirement allowing the use of non-constant probability distributions governing sojourn times in each state, such as Weibull or Log-normal. The Markov property is still valid between states, as state history does not affect sojourn times, only the residence time in the state hence the semi-Markov name.

2.2 Regular Petri Nets

A Petri Net is a directed graph where nodes, referred to as places, are connected by arcs to other places via transitions. Places hold a discrete number of tokens. Tokens move from place to place via transitions when specific conditions are met (BSI Group, 2012). When modelling a system, places are commonly used to represent states, where the presence of a token denotes the validity, or degree of validity of that state.

Figure 2-1 shows a basic Petri net. The arc beginning at Place 1 and terminating at Transition 1 specifies that Transition 1 requires one or more tokens to be present in Place 1 to enable Transition 1. If there were multiple arcs terminating at Transition 1 from multiple places, each condition must be met for the transition to be enabled.

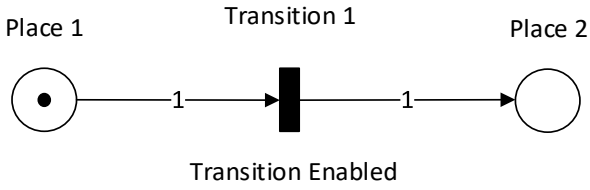

Figure 2-1 Basic Petri Net with Enabled Transition.

The enabled transition can now 'fire', shown Figure 2-2. When this occurs the tokens specified by the arcs terminating at a transition are subtracted from their respective places. The outgoing arcs from a transition specify the number of tokens to be added to the connected place when the transition fires. The number of tokens within the Petri net do not need to be conserved.

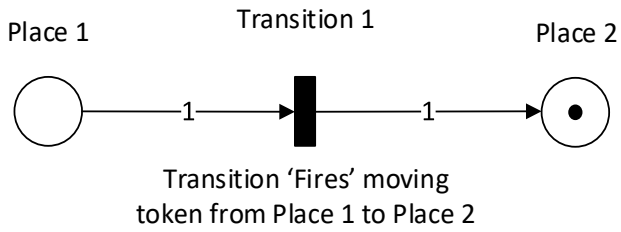

Figure 2-2 Basic Petri Net Transition Firing.

In Regular Petri nets, if there are multiple tokens enabling a transition, the transition will fire consecutively till there are no longer sufficient tokens to enable the transition. The number of tokens within a place may also be depicted by a number within the place, rather than using token icons.

2.2.1 Petri Net Elements

Elements of Petri nets not previously defined are outlined below:

Figure 2-3 Delay Transition

Delay Transition: This Transition fires a pre-determined time after its firing condition has been met, the number denoting the delay.

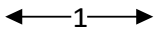

Figure 2-4 Test Arc

Test Arc: These are two standard arcs superimposed which results in no net movement of tokens after a transition fires, but is required to enable a transition.

Figure 2-5 Stochastic Delay Transition

Stochastic Delay Transition: This Transition fires with a delay determined via random variable. The random variable may be chosen using any distribution. Later in this thesis different colours are used to denote different distributions.

Figure 2-6 Reset Transition

Reset Transition: This Transition behaves differently to those introduced thus far, when the firing condition of this transition is met it resets connected places to a pre-programmed number of tokens. This action can be achieved using other Petri Net structures, however this form is significantly more concise (Andrews, 2012).

Figure 2-7 Conditional Stochastic Delay Transition

Conditional Stochastic Delay Transition: The parameters of the statistical distribution used to select the delay time of this Transition change depending on the state of linked Places.

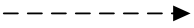

Figure 2-8 Conditional Arc

Conditional Arc: This arc is used with the Conditional Transition indicating which places the Transition properties depend on.

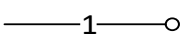

Figure 2-9 Inhibitor Arc

Inhibitor Arc: This Arc inhibits the firing of a Transition when an equal or greater number of tokens are present in the connected Place than that specified by the arc. This allows greater flexibility when designing a Petri net.

With places and tokens representing asset and system states, transitions are used to model the change of system and asset states by moving tokens from place to place. Stochastic systems are modelled with stochastic transitions. Very simple Petri net models might have

an accessible analytical solution, however in practise, Petri nets are often analysed using Monte Carlo methods, described later.

2.3 Coloured Petri Net

Coloured Petri nets incorporate the graphical modelling approach with high level elements of programming languages. Such as: non-integer variables; control statements; and complex functions. A further key difference with regular Petri nets is that a Coloured Petri net transition will fire concurrently when enabled by multiple sets of tokens (BSI, 2019). In contrast, a regular Petri net transition would fire sequentially in this scenario.

The most advanced development of Coloured Petri nets has been conducted by Jensen. Their developed syntax is based on mathematical expressions. Software developed by Jensen uses an interpretive compiler to convert these expressions into machine code for simulation (Jensen & Kristensen, 2009). Despite being well developed, this syntax and software is unsuitable for the Colour Petri net modelling work later in this thesis. Mathematical expressions may give little indication of the potential computational processes which can solve them, and no indication of the actual process employed. As later work in this thesis considers computational costs and processes, programming expressions which explicitly state the computational processes will be used instead.

The graphical notation used is broadly based on Jensen's language, however programmatic elements will be expressed in a form similar to the programming language C++ and Visual Basic. These are introduced below.

In a CPN tokens hold values, known as Colours. These can be literal colours, numbers, or any other object. In this work, Colours are limited to variables types commonly used in programming languages. The entirety of Colours which a token holds is referred to as a Colourset. Figure 2-10 shows an example Colourset definition for tokens representing work orders. Lines 2-4 define Colours, the Colour AssetNumber is defined as an integer, AssetType as a string, lastly, GenerationTime is defined as a double to give intra-day precision. Line 1 uses all these Colours to define the Colourset UrgentWorkOrder. Later in modelling chapters, Colours and Coloursets may be abbreviated for clarity in diagrams. The Colourset UrgentWorkOrder, for example may be referred to as UWO.

```
1. Colour UrgentWorkOrder As AssetNumber x AssetType x GenerationTime
2. Colour AssetNumber As Integer
3. Colour AssetType As String
4. Colour GenerationTime As Double
```

Figure 2-10 Example Colour Definition.

Places hold no additional significance in CPNs compared to regular PNs. For diagrammatic clarity, however they are depicted as larger than regular PN places, shown Figure 2-11. The token Coloursets which a CPN place can hold are listed in the centre of the place.

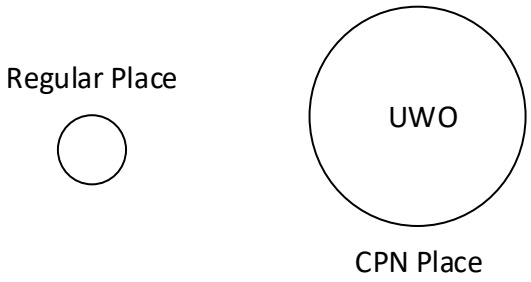

Figure 2-11 Place Type Comparison

Arcs from Places to Transitions still specify a token condition to be met. However, their role is expanded within the CPN methodology. Firstly, arcs specify the token Colourset required, and the number of tokens. This is shown in the example in Figure 2-12, a single UWO token is specified as 1'UWO. Normally however, if only one token is required the 1' is dropped. AN = AssetNumber creates the variable AN, and sets it equal to the AssetNumber colour value of the token. The variable AN may then be passed to any functions within the transition or outgoing arcs. All variables created in this manner are local only to the directly connected transition's functions and outgoing arcs. Due to this, variable names may be reused across different transitions, but no interaction is implied.

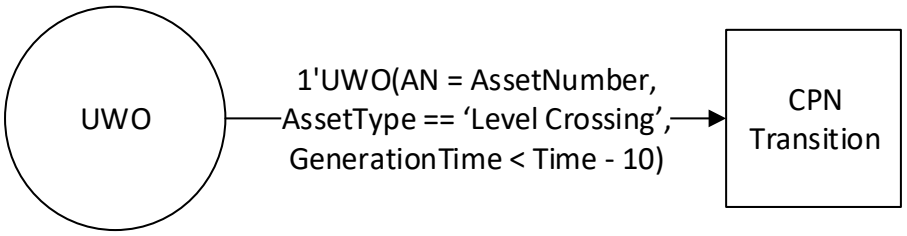

Figure 2-12 CPN Incoming Arc Example.

Comparison operators can also be used within incoming arcs. These specify specific colour values or conditions which a token must meet to enable the transition. In the above example, the AssetType Colour must be equal to 'Level Crossing', and the GenerationTime Value must be lower than the current simulation time minus 10. Simulation time is considered a global variable, and thus can be accessed by any function without being passed in.

Transitions may hold up to two functions, a Guard function and a Delay function, illustrated in the next example. Figure 2-13 contains two places and a transition. The arcs from place to transition assign variables for AssetNumber and GenerationTime.

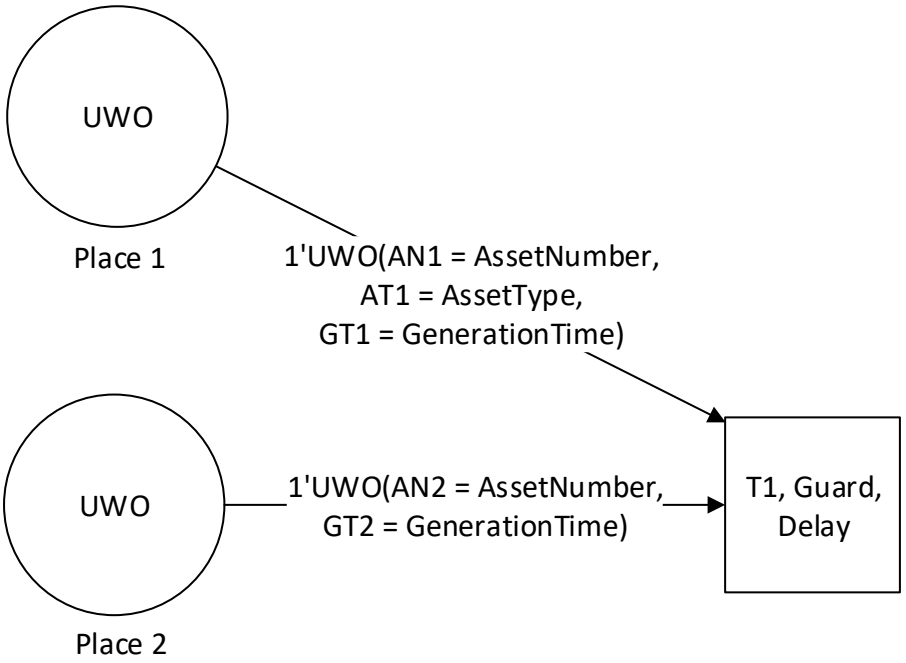

Figure 2-13 CPN Transition Example.

The Guard function within transition T1 evaluates whether T1 can be enabled, assuming there are tokens present in both places. The guard function is defined in Figure 2-14 over lines 5 to 8. The first line specifies the function type and the transition it belongs to, the parenthesis show the variables which must be passed to the function. The next line evaluates if the variables AN1 and AN2 are equivalent, if yes, the function returns enabled, and otherwise returns disabled.

```
5.  Transition T1  Guard(AN1, AN2) {
6.          If (AN1 == AN2) { Return Enabled }
7.          Else { Return Disabled }
8.  }
9.  Transition T1 Delay (GT1, GT2) {
10.         Bool Overdue = false
11.         Double Delay
12.         If (GT1 < Time – 10 Or GT2 < Time – 10) {
13.                 Overdue = true
14.         }
15.         If (Overdue == true) {
16.                 Delay = Random_LogNormal(0,0.25)
17.         }
18.         Else {
19.                 Delay = Random_LogNormal(2,0.25)
20.         }
21.         Return Delay, Overdue
22. }
```

Figure 2-14 Transition Function Examples.

The delay function begins in a similar fashion, specifying the transition it belongs to and its type on line 9. Lines 10 to 11 declare variables used within the function. The variable Delay

is always used to set transition firing delay time. The body of the function evaluates if either GenerationTime is older than 10days. This decides which LogNormal distribution parameters are used to generate the random delay time. The penultimate line of the function line 21, sets the firing time delay, and allows the Overdue variable to be accessed by any outgoing arcs (transition to place). Sometimes, colours of tokens will be accessed within functions which have not been explicitly passed as a variable via an arc. This is done for brevity where otherwise large numbers of variables will need to be passed.

CPN transitions can be enabled and fire concurrently. If there are multiple tokens in both Place 1 and Place 2 several independent instances of the arcs and functions will be evaluated. This will create several independent instances of the transition being enabled and ultimately firing, referred to as transition modes.

Figure 2-15 shows the sole outgoing arc from the example transition T1. This arc contains a function, which tests the value of Overdue, created by the transition's delay function. If true, an UrgentWorkOrder token is created and added to place 3. This expression uses the variables created from the incoming arcs to set the value of the Colours within the new token.

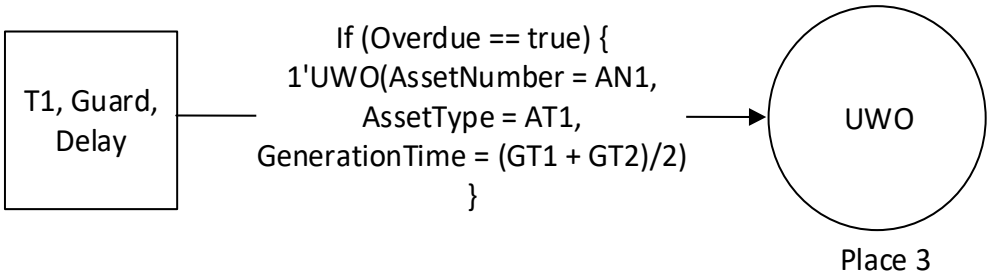

Figure 2-15 Outgoing Arc Example

2.4 Monte Carlo Modelling

Monte Carlo analysis has been regularly used in the field of reliability analysis and more recently in infrastructure asset management. It has found use here because the mathematically complex nature of reliability models and infrastructure asset management models often precludes an accessible analytical solution without significant limiting assumptions, such as the memoryless assumption in Markov models. Monte Carlo methods are agnostic of the complexity or distributions used with a model. Provided sufficient samples can be drawn within an acceptable computation time any infrastructure management model can be solved using this method.

Solving a model via Monte Carlo is reliant upon the generation of random numbers. It is not possible to generate true random numbers via computer, as this can only be achieved using physical processes such as radio nucleotide decay. However, computers can approximate random numbers by using pseudorandom number generators. These computational devices produce sequences of numbers which satisfy some, or most, of the statistical requirements for random number generation but not all. The pseudorandom number generator used in this work is Mersenne Twister, as it is regarded as suitable for general purpose use (Matsumoto & Nishimura, 1998).

2.4.1 Random Sampling from Distributions

The pseudorandom numbers generated must be used to sample from the required distributions. Regardless of distribution, this relies upon inverse transform sampling. This states that there is a one to one correspondence between any two Cumulative Distribution Functions (CDF), as by their nature all CDFs are monotonic increasing (Rubinstein & Kroese, 2017). Therefore, any CDF function can be equated to the equivalent CDF of the uniform distribution over [0,1] output by pseudorandom number generators. An example is shown over equations 2-2 to 2-5.

$$F(x_i) = \rho_i$$

Where:

$F(x_i)$ is the CDF of the desired distribution

ρ_i is an uniformly distributed random number between [0,1]

2-2

Applying this to the exponential distribution CDF, shown 2-3

$$F(x) = \int_0^x \alpha e^{-\alpha x}\, dx = 1 - e^{-\alpha x}$$

2-3

Applying the inverse cumulative distribution method yields the equation shown in 2-4 where ρ_i is a random number between [0,1].

$$\rho_i = 1 - e^{-\alpha x_i}$$

2-4

Solving for x_i, equation 2-5 provides an equation that can be used for sampling.

$$x_i = -\frac{1}{\alpha}\ln(1 - \rho_i)$$

2-5

Using this formula the uniform randomly distributed output between [0,1] from the pseudo random number generator, ρ_i, can be used to sample from the exponential distribution, and through the same process other statistical distributions such as Weibull, Normal and Log-Normal. However, in practise it is not necessary to derive formulas such as these as most programming languages have libraries available for sampling from these common distributions. These libraries will be used when sampling random numbers when solving the models presented in this thesis.

2.4.2 Convergence

Monte Carlo methods can be used to analyse a Petri net with stochastic elements. A Petri net is simulated following the rules described earlier. When a transition with a stochastic delay is enabled, a pseudorandom number is generated using the distribution and parameters specified by the transition and used as the delay. This affects the outcomes of the simulation. If the model is being analysed to find the value of a continuous variable such as system availability, a different result will be produced with each simulation. Simulations are repeated until sufficient data has been extracted such that the averaged result of all the simulations, known as sample mean, approaches the true result, known as the population mean. This process of repeating the simulation to allow the sample mean to tend towards the population mean is known as convergence. Determining the number of times to repeat the simulation is important, as insufficient samples may result in the sample mean not approximating the population mean. Convergence can be determined by inspection, but more formally by using the central limit theorem.

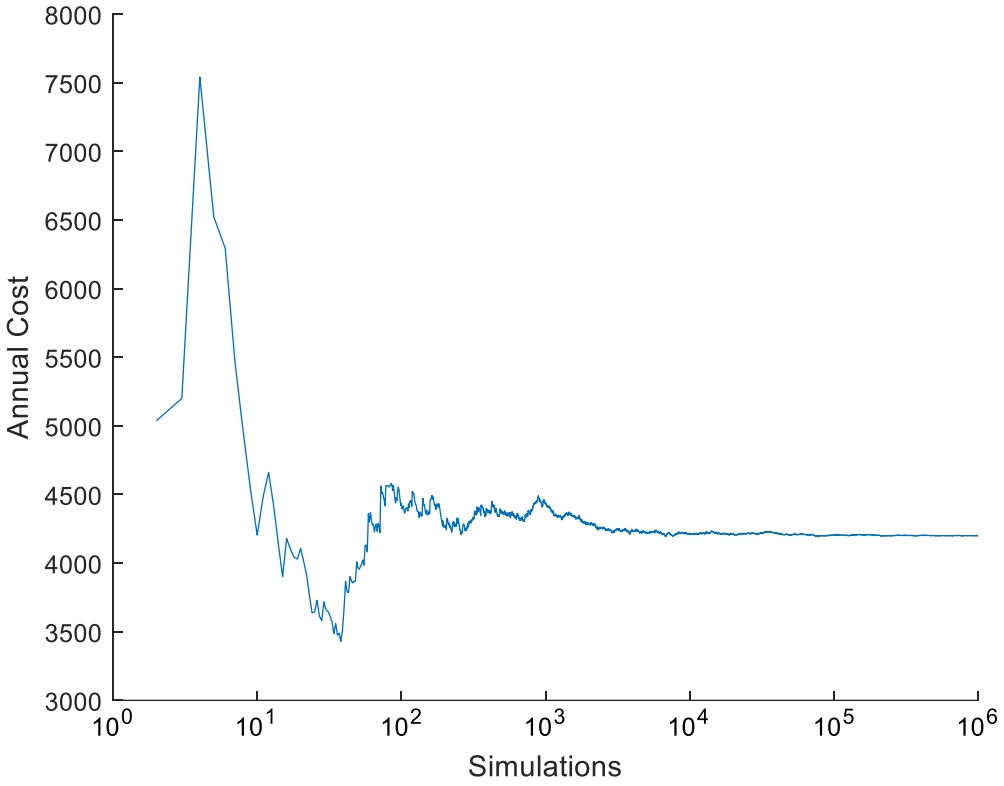

Figure 2-16 Monte Carlo Convergence Example

Figure 2-16 shows an example of convergence for a Petri Net presented later in this thesis to determine the annual maintenance cost of a railway asset, analysed using Monte Carlo. The x-axis uses a log scale as the convergence rate of any Monte Carlo method is the reciprocal of the root of the number of samples, and therefore not linear. By inspection it appears sufficient convergence has been reached by 100,000 simulations as the fluctuations become indistinguishable at the scale used. The central limit theorem was used to create confidence intervals, equation 2-6 (Thomopoulos, 2012), the 90% confidence interval produced the result in equation 2-8,

$$P\left(\bar{x} - \frac{\lambda\sigma}{\sqrt{N}} \leq \mu \leq \bar{x} + \frac{\lambda\sigma}{\sqrt{N}}\right) \cong \frac{1}{\sqrt{2\pi}} \int_{-\lambda}^{\lambda} e^{-u^2/2}\, du$$

2-6

$$P\left(4210 - \frac{1.645 * 3600}{\sqrt{100000}} \leq (z) \leq 4210 + \frac{1.645 * 3600}{\sqrt{100000}}\right) \cong \frac{1}{\sqrt{2\pi}} \int_{-1.645}^{1.645} e^{-u^2/2}\, du$$

2-7

Simplifying,

$$P(4190 \leq (z) \leq 4230) \cong \frac{1}{\sqrt{2\pi}} \int_{-2.58}^{2.58} e^{-u^2/2}\, du$$

2-8

It is estimated that 90% of all sample means comprising 100,000 samples from the model will produce average annual maintenance costs within the [£4190, £4230] interval, which is an acceptable degree of accuracy.

2.5 Optimisation

Asset management models are used to predict the properties of an asset management plan or maintenance policy on key performance indicators such as cost or risk. Using mathematical or computational methods in conjunction with a model enables the automated discovery of optimal policies. Typically, the objective of an optimisation is to minimise costs or maximise safety, though may also be applied to performance measures relating to capacity or reliability. A critique of automated optimisation is that it can prioritise one characteristic over another, such as reduced maintenance costs at the expense of degraded asset condition or reductions in safety. This is discussed below, alongside some common methods, considering their implications for the optimisation of large multi asset models.

2.5.1 Single Candidate Optimisation

Neighbourhood search is one of the simplest heuristic optimisation algorithms. From a starting set of model parameters, changes to each parameter are tested in turn by evaluating the model. The incremented parameter which has the greatest desired effect on the objective is selected to form the new set of model parameters in the next iteration of the algorithm. The algorithm ends when no changes produce an improvement in the objective, yielding a single optimised set of model parameters, known as a candidate solution (Rayward-Smith, et al., 1996).

This form of optimisation heuristic is much more computationally efficient than an exhaustive search, and therefore an attractive option for optimising computationally demanding models. However, it is liable to get stuck in local maxima or minima and fail to locate more optimal candidates. Algorithms such as Tabu Search and Simulated Annealing are based on Neighbourhood search but crucially include features to allow them to escape local maxima or minima. Like neighbourhood search, they produce a single optimised candidate.

A simple Tabu Search can escape local features by forcing the selection of the least bad parameter, and making undoing that parameter change Tabu for several iterations enabling the optimisation to progress. A variety of implementations of Tabu searches have been developed, using the premise of not allowing some parameter selections or changes for a period of iterations (Glover, 1990). Tabu Search is rarely used as the sole method of optimisation in literature however, because it does not converge over successive iterations. It is often used in conjunction with other methods to improve their efficiency by preventing the exploration of consistently ineffective parameter changes (Liang, et al., 2012).

Simulated Annealing uses random number generation to allow it to break out of local features. When a set of parameters are tested and the optimisation objective found to be worse than the previous iteration, a neighbourhood search would discard those parameters. Simulated Annealing would instead generate a random number, and compare it against a Cooling Function. Should the random number be smaller than the Cooling Function, the new test parameters are accepted even though they are less optimal. This allows Simulated Annealing to escape local maxima or minima, though does not guarantee it. The Cooling Function may be determined through any formula deemed suitable, however it should decrease with successive iterations of the algorithm. This will cause the solution to converge as the randomly generated number becomes progressively less likely to be lower than the value of the Cooling Function and the algorithm ends in a local maxima / minima with no further iterations possible.

Applying optimisation algorithms to single objectives such as cost may find cost optimised policies, however these may be at the expense of increased risk. Multi Objective Optimisation allows the concurrent optimisation of several objectives, such as cost and risk. Use of an objective function which relates cost and risk can be used to achieve this, commonly where a monetary cost is assigned to risk or an acceptable limit placed on either.

With use of such an objective function, algorithms like Simulated Annealing or Tabu search can be used to optimise for both cost and risk. However, there are limitations to this approach. To illustrate these, consider Figure 2-17, this shows the concept of a Pareto Front. For a multi objective optimisation, the Pareto Front is the boundary between the region where candidate solutions may be found – shown as blue dots, and the infeasible region where no candidate solutions can be found. Candidate solutions on the Pareto Front represent the most optimal solution for the given ratio of cost to risk.

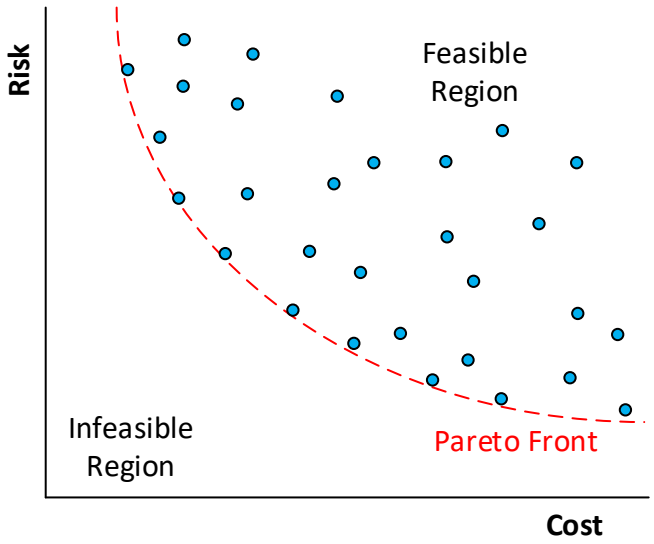

Figure 2-17 Idealised Pareto Front for Cost vs Risk.

A single candidate algorithm such as Simulated Annealing will only produce a single point. As a heuristic method it is not certain the candidate produced will be on the Pareto Front. Repeating the optimisation will provide further confidence in the location of the Pareto Front and that the candidates produced are optimal. Further, the type and configuration of the Objective Function may influence which regions of the Pareto Front are populated.

There are practical limitations to algorithms which produce a single candidate only. Some candidate solutions may exploit assumptions in the model and therefore be dubious, whilst other candidate solutions may be impractical to implement and therefore unusable. For any application where there is an intention to implement an optimised asset management plan or policy there should be multiple candidates to choose between. A further practical limitation is being able to relate the two quantities being optimised using an Objective Function, in terms of risk this could be assigning a monetary value to lives lost or hazardous events which may be impossible or otherwise ethically inappropriate.

Due to these limitations, more computationally expensive multi candidate optimisation algorithms should be considered for optimising asset management even when the computational burden of the asset management model is significant.

2.5.2 Genetic Algorithms

Genetic Algorithms are a fundamentally multi candidate search heuristic where a process emulating evolution is used to create optimal solutions. An initial population of potential candidate solutions is created using parameters randomly chosen from the search space. The best candidates within the population are then interbred to create the next population by

averaging the parameters between each parent as well as introducing random mutations. This is repeated till for several generations until the Parento Front is reached (Deb, 2001). Because this algorithm is multi candidate, an Objective Function is no longer required, though can be used. Instead, ranking algorithms such as Kung's method can be used to identify the most optimal candidates without specifying a relationship between cost and risk (1975).

This method can be computationally expensive though is considered robust. Many authors have applied Genetic Algorithms to the optimisation of Petri Nets within the field of asset management. It has been applied to optimising maintenance, D'Souza developed a model for the maintenance of offshore oil and gas installations (2017), applying a multi-objective Genetic Algorithm to optimise for cost and safety.

A rail track asset Petri Net model was optimised using multi-objective a Genetic Algorithm to minimise maintenance costs and maximise track condition by Audley (2014), the model contained 8 parameters but with 83,200 total potential combinations. The Genetic Algorithm was performed with an initial population of 20 and required 25 generations to obtain the Pareto Front. A Coloured Petri Net model was created for overhead line equipment maintenance modelling and optimised using a Genetic Algorithm by Kilsby et al (2017), this featured an initial population of 100 candidates was maintained over 200 generations to locate the Pareto Front for a model containing 100 variables.

Based upon this discussion, it is recommended that for large multi asset models, efficient single candidate optimisations methods such as Simulated Annealing should be used when optimising for a single objective only. However, for multiple objective optimisations the additional computational cost of using a Genetic Algorithm may be justified if a convenient relationship between the optimisation objectives is not available.

2.6 Chapter Summary

This chapter was intended to provide an understanding of the modelling methodologies currently in use for infrastructure asset management decision support, namely Markov and Petri Nets. This will provide a suitable background with which to interpret the literature presented in the following chapter, and to evaluate their effectiveness in a railway asset management modelling framework.

Markov models are simple to analyse, however this simplicity comes at the expense of assuming varying degrees of memorylessness depending on the specific Markov method.

The impact this assumption has on infrastructure modelling will depend on the asset itself. As will be highlighted in the next chapter, some asset's degradation rates are conditional on asset history limiting the accuracy of Markov when applied. Further, Markov models experience state space explosion, where each additional state modelled increases the computational requirement exponentially when pursuing an analytical solution. Ultimately, this means that Markov can be effectively applied to some assets, but not others, placing doubt on its effectiveness as the core of an asset management framework.

In contrast to Markov, Petri net models solved via Monte Carlo do not assume memoryless, enabling the model to be conditional on asset history. The computational cost of repeated simulations of a model to achieve a sufficient level of convergence is an issue requiring exploration in the context of frameworks, as such models could become large and slow to simulate. These models have a further advantage of being very easy to modify, enabling easier development and more flexible usage. Changing the statistical distribution used in a model solved via Monte Carlo is trivial, however doing the same for Markov Model may require re-evaluating the mathematics and Markov model type.

Various methods for model parameter optimisation have been explored. Global search methods such as Genetic Algorithms are considered much more robust than single candidate algorithms. However, their computational cost is far greater, making selection of an appropriate method an important consideration for railway asset management where models may be computationally extensive to simulate.

3. Modelling Review of Railway Assets

This literature review aims to directly fulfill the first objective of this thesis, to determine the state of asset management modelling for railway assets and infer which methods might be best suited for whole system, whole life modelling. For each asset it outlines the strengths and weaknesses of the different modelling methodologies applied. From this, it may be possible to determine the most effective modelling methodology for a given asset. This review will highlight assets where there are no effective models. This enables the development of new models to fill the modelling capability gaps, which is the second objective of this thesis.

The assets reviewed within this section are as follows:

1. Bridges
2. Track
3. Electric Traction Transmission Equipment
4. Level Crossings
5. Earthworks
6. Tunnels
7. Signaling

3.1 Bridges

Bridges are a common feature in any transportation network, allowing passage over impassable features such as rivers or allowing transport networks to cross over each other. Bridges employed on the UK rail network are categorised as either overline bridges, where the railway passes under the bridge with road traffic and pedestrians crossing the rail line using the bridge, or underline bridges, where the rail line uses the bridge and road traffic passes underneath.

There are many differing design philosophies and materials available with which to construct bridges, such as: masonry arch, steel beam, steel truss and reinforced concrete. Regardless of construction type or material, bridges suffer from degradation due to inadequately mitigated environmental exposure and use, which increases the risk of collapse. Maintenance actions can prolong the lifespan of bridges by reducing the rate of degradation and repairing degraded bridge elements. Bridge models which integrate both degradation modelling, and the restorative effects of maintenance actions support effective allocation of limited maintenance budgets.

3.1.1 Markov Models

Jiang and Sinha published the first Markov model for bridge degradation (Jiang & Sinha, 1989). The purpose of the model was to predict the serviceable life remaining for highway bridges. This model utilised 7 states, each representing the bridge's overall condition from very good to very poor. A single transition probability connects each degradation state with the state proceeding. The final degraded state is an absorbing state. The model used a discrete 1-year time step, assuming a maximum of 1 state transition per timestep. For a service life between 0 and 60 years, the Transition Probability Matrix (TPM) was recalculated every 6 years to reflect changing degradation rates with bridge age. The data used to create the TPM was sourced from highway bridge inspections from the American state of Indiana, whose bridge inventory comprises steel and reinforced concrete types.

A polynomial regression model of condition against age was created from the same data for comparison with the Markov model. It was observed that this regression model was only useful for predicting the degradation of bridges whose present condition lay on the regression curve. The Markov model however was capable of predicating bridge degradation regardless of initial state and age. A chi squared test found minimal difference between the model predications if the initial bridge state lay on the regression curve. It was observed that the regression model tended to overestimate bridge lifespan compared with the Markov model.

This early example of a bridge deterioration model did not consider the effects of corrective or preventative maintenance on bridge service life. In effect, assuming that only age influences degradation rates. Later authors would find that bridge deterioration rates correlate more with maintenance levels than bridge age (Sianipar & Adams, 1997).

A subsequent Markov model for bridge degradation was produced by Cesare et al (Cesare, et al., 1992). In this model bridges are comprised of discrete elements relating to major structural components. Each element is modelled using 7 states relating to condition, modelled using a discrete time step. The TPM for this model was produced using inspection data from America's New York state, comprising reinforced concrete and steel bridges. This model utilises a non-stationary TPM to model areas where the degradation rate of one element is conditional on the degradation state of another element. It achieves this by updating the TPM every time step based on the expected element degradation states using a simple linear formula.

Unlike the previous Markov model, this model includes maintenance. A separate Repair Probability Matrix (RPM) is presented whose function is to the control the rate at which lower condition states return to higher condition states. Different repair policies may be applied via modifications to the RPM. The paper presents an example of a renewal only policy, where elements in the 3 most severely degraded states have a 10% chance of being replaced each time step. A further example is given showing an implementation of imperfect repair, where 90% of all repair actions result in a return to the highest condition state and 10% return the element to the second highest state. The expected number of repairs is summed to predict the lifetime maintenance cost for a typical bridge.

This paper has made some important advances over previous work. Modelling on a per element basis enables more accurate condition modelling. The failure of a bridge may be precipitated by only one or two individually degraded elements regardless of the overall bridge condition. Bridge element condition depends on both degradation processes reducing condition states and maintenance processes restoring condition states. Therefore, the integration of maintenance in this model is important for predicating the future condition states.

However, the implementation highlights problems with integrating maintenance in Markov models. Maintenance policies and processes contain both stochastic and deterministic elements. Maintenance effectiveness is difficult to predict and therefore well represented as a stochastic process in a Markov model. The decision of whether to renew a severely degraded bridge is dependent on the availability of resources. An area authority may have the resources to repair a fixed number of bridges per year. This fixed number of repairs cannot be modelled using the Markov methodology and therefore must be approximated by repairing a fraction of the degraded bridges. This will result in the model predicting more or less maintenance than is actually feasible each time step, depending on the expected condition states. The limitations of integrating maintenance in Markov models are explored further later in this section.

Schere and Glagola explored using Markov models to optimise maintenance decisions for an entire inventory of around 13,000 bridges for the American state of Virginia (Schere & Glagola, 1994). They determined that modelling this number of bridges with shared resources was impossible using the established Markov bridge models due to the computational requirements. Their paper outlined that by modelling the number of bridges expected to be in any given condition state, rather than the bridge's individual condition

states, the number of states required for modelling would be reduced. The complexity of their hypothetical model was further reduced by categorising similar bridges together and modelling them as one. This assumes each similar bridge is representative of its class, which may reduce the accuracy of any model using this assumption.

The paper then presents an example model similar to that described by Jiang and Sinha. The authors then tested the Markov property. Using historical inspection records, they found commonly occurring condition state sequences comprising the results of three consecutive inspections. Statistical testing was performed to see if the 1st state in the sequence of inspections affected the probability of a state change between 2nd and 3rd inspection. The authors found the 2nd to 3rd state transition probability to not be conditional on the 1st state, finding the Markov property valid for their dataset.

As part of their PhD thesis, Robelin presents a model for the degradation of bridge decks (Robelin, 2003). The model is formulated as a Markov chain modelling the degradation state of the deck in terms of reliability index. This model is then used to develop an augmented state Markov decision process model, where annual maintenance decisions are made based on other information in addition to the deck condition. The maintenance decision process may also consider the previous maintenance action, the time elapsed since it was carried out and the deck condition which triggered that action. The maintenance decision for each condition state was optimised using dynamic recursion.

Semi Markov models partially relax the Markov property. Previous states still do not affect the probability of transitioning to future states, however in a semi-Markov formulation the duration within the state does affect transition probability. This is achieved with a traditional TPM in addition to a holding time probability. Like a continuous Markov chain model, this results in the formulation of differential equations. Ng and Moses developed a semi Markov model for bridge degradation. In their model, 7 states are used, with the sojourn time between each state governed by a Weibull distribution (Ng & Moses, 1998). Maintenance actions have been implemented on a continuous time basis, though little detail is provided on what specific policies have been used or tested. The model was found to be produce lifetime degradation results similar to previous Markov models and regression techniques.

Morcous set out to evaluate some of the assumptions inherent to creating and using Markov degradation models (Morcous, 2006). To achieve this a 6-state discrete Markov chain degradation model was created. Data from inspection of Canada's Quebec province highway bridges was used to create the TPM. These bridges were typically inspected every 2 years,

but some bridges were inspected more or less frequently. A second TPM was formulated from this data, using Bayes theorem to adjust the TPM to reflect the variations in inspection frequency. When the TPM was used in the model the working lifespan prediction increased 22% over the non-adjusted TPM. This represented an improvement in modelling accuracy over the non-adjusted TPM results. The Markov property was also tested using similar methodology to previous authors and found to be acceptable.

A paper was published by Yang et al exploring Markov modelling for degradation of Hong Kong's transportation bridges (Yang, et al., 2009). Using inspection data, the Markov property was tested using the method employed by Schere and Glagola, and found to be valid. The authors proposed a framework to determine the most effective type of Markov model to use when modelling bridge degradation based on the inspection frequency and data available. The framework states that for bridges with periodic inspections and where the Markov property is believed to be valid a discrete Markov chain is appropriate. If the Markov property is not valid a semi-Markov model is deemed appropriate. If inspections are not intended to be periodic a semi-Markov model is also recommended. However, it may be more appropriate to implement a continuous Markov chain model if the Markov property is valid.

Mašović and Stošić developed a semi-Markov decision process model to optimise bridge maintenance decision making (Mašović & Stošić, 2015). The model features six states and two possible repair actions which are applied to states three and worse. A bridge may be renewed, or repaired, with repairs stochastically restoring the condition to one of several previous condition states. Linear programming was used to find the optimal maintenance decision between repair and renewal for each condition state.

Le and Andrews paper demonstrates a continuous Markov chain bridge degradation model (Le & Andrews, 2013). This model had many differences to those seen up to this point. This is the first example in literature of a continuous Markov model which did not use a semi-Markov methodology. This model uses 4 condition states per element, with 4 corresponding maintenance states. After each inspection there is a discrete computation phase, the values of the 3 degraded states are added to the corresponding maintenance states and then set to zero. The probability of being in each state at any given time is obtained by summing the probability of the degradation state and its corresponding maintenance state.

The Transition Rate Matrix (TRM) for this model was derived from maintenance records. The purpose of this was to avoid spontaneous condition improvements that may be seen on

inspection records due to maintenance occurring. From these records the Mean Time to Intervention (MTTI) can be derived for minor repairs, major repairs and renewals. The cost and duration of the repair was used to infer if a repair was minor, major, or a renewal, each of which features in the model in addition to servicing. The inverse of the MTTI was used to estimate the transition rate between each state. This required the following assumptions about the maintenance records: the transition probability is constant, repairs always restore the element to a new condition, repairs occur promptly after the degraded state is discovered, repair cost and duration is only affected by the degree to which the element is degraded. The maintenance records were also used to determine the Mean Time to Repair (MTTR), which was used as the TRM values for the repair states.

The paper presents a method for performing opportunistic maintenance on pairs of similar elements. This requires 16 paired condition states, and 16 paired corresponding repair states, accounting for each possible combination element of degradation using the 4 levels of degradation previously described. This enabled maintenance to be carried out on the pair of elements simultaneously. This was simulated with 32 pairs of elements, requiring 2048 states. This reportedly required 1 minute of computation to analyse 60 years of degradation. Different maintenance strategies were tested by modification of the discrete computation phase. For example, a renewal only policy was explored by only adding the value of the worst condition state to its corresponding repair state to reflect renewals occurring only when the element has reached a very poor condition state.

The use of maintenance records to determine MTTF in this paper was novel. The models displayed an extensive integration of maintenance with degradation modelling, however in doing so highlighted an issue with the Markov modelling methodology. The opportunistic maintenance model considers this form of maintenance occurring only to pairs of elements, in reality maintenance may be performed opportunistically on any number of elements. However, to model this using the structure employed in this model it requires a number of states equal to the number of condition states per element raised to the power of the number elements which are repaired opportunistically. This clearly results in an explosion of the number of states required for increasing the number of elements.

The results of the renewal only maintenance strategy highlights deficiencies when using Markov degradation models with integrated maintenance to predict future condition states. Renewal is a fundamentally discrete event, prior to renewal the element will be in a very poor condition, and after in an as new condition. This will result in oscillations between new

and very poor condition over several lifespans of the element as it is renewed. However, this is not observed in the results of this maintenance strategy, which instead rapidly reach a steady state without oscillation. This occurs because Markov states represent the probability of an element being in any given state. At any given time, there must be some probability that an element will be in the very poor state. Renewal actions are modelled as occurring in proportion to this probability. Consequently, a highly discrete maintenance action such as renewal is modelled as continuous, with the restorative effects of renewal being spread out over the lifespan of the element in proportion to its probability of being in the very poor state. This behaviour will be present for all condition states, and maintenance strategies, but it is most evident when considering only renewals.

At first glance, one may consider fixing this behaviour by implementing a minimum state probability which must be reached prior to the implementation of maintenance. This would force oscillations in the condition state, however, would do so at the expense of complicating the maintenance strategy being analysed. As the state represents a probability, a minimum state probability before renewal would imply a maintenance strategy where an element in very poor condition state could be ignored for a period of time. This is clearly a considerable deviation from analysing a renewal only maintenance policy. From this, it can be concluded that Markov degradation models with integrated maintenance are only effective at predicting the steady state condition of bridge elements when the beneficial effects of maintenance actions occur in a continuous manner.

The Markov property requires the assumption that degradation processes are memoryless. Many of the papers discussed have tested this property and found it valid to be valid. There is no clear evidence presented thus far that the use of a semi-Markov methodology offers any advantage in modelling accuracy. A further general limitation of Markov modelling relates to the expansion of the number of states required for a model, generally referred to as state space explosion. Schere and Glagola determined that state space explosion would render Markov unable to model large numbers of bridges sharing common maintenance resources without simplifying assumptions (Schere & Glagola, 1994). Le and Andrews found considering opportunistic maintenance results in a rapid expansion of the number of states required (Le & Andrews, 2013).

The most significant issues highlighted in this review relate to the integration of maintenance into Markov degradation models. The work of Cesare demonstrated the difficulties of translating a maintenance strategy into stochastic terms, when forced to express renewals

as a probability, not as a quantity (Cesare, et al., 1992). No authors have considered preventative maintenance using the Markov methodology. Additionally, no authors have applied any maintenance effectiveness limits to their bridge models, where repeated maintenance becomes less effective. Le and Andrews model demonstrated the difficulties created by applying maintenance policies based on state probabilities, not actual states (Le & Andrews, 2013).

3.1.2 Lifetime Analysis Models

Frangopol et al developed a reliability based approach to bridge degradation modelling (Frangopol, et al., 2001). Their model uses a numeric variable referred to as the Beta Index which correlates to different degradation states (Ghosn & Frangopol, 1998). Condition indexes do not refer to what degradation has occurred, this prevents modelling maintenance actions which are specific to the degradation. After a randomly determined period of time the Beta Index decreases at a constant rate representing initiation of a defect, and progressive worsening of the defect. This correlates to a probability of bridge failure either due to shear forces or bending moment exceeding structural resistance. In this paper, the authors extend this concept to include preventative maintenance. Preventative maintenance is modelled as occurring at a random time from the construction of the bridge and is repeated at a randomly determined frequency. This is to explore the effects of uncertainty on lifespan of the bridge. Preventative maintenance reduces the rate of reduction in Beta Index. The effectiveness and duration of this effect is also determined randomly. These random elements are governed using a Weibull distribution. The model is solved using a Monte Carlo method to predict changes in Beta Index over time. The model was applied to predicting the cost of maintaining an inventory of bridges.

The model is applied to three scenarios varying when the initial degradation starts relative to the state of the preventative maintenance applied to the bridge. The model's results show that the application of preventative maintenance decreases bridge management costs by reducing the degradation rate. However, the results of this model were not validated against any historical data.

Yang et al created a model using lifetime functions to model bridge degradation. This model builds upon the previous work by Frangopol et al (Yang, et al., 2005). Bridge condition is modelled using a binary performance indicator, the probability of failure of the bridge. This model considers a bridge as a system of elements required to support the structural integrity of the bridge. These elements feature redundancy and are expressed using reliability block

diagrams. The probability of failure of an element is expressed using either a Weibull or exponential distribution. The application of preventative maintenance reduces the rate of increase of the probability of failure of an element. Whilst the application of corrective maintenance reduces the probability of failure. The model was created using data from American highway bridges.

A further lifetime model was developed by Noortwijk and Klatter (Noortwijk & Klatter, 2004). This model was used to predict the cost of maintaining an inventory of Dutch concrete bridges. To achieve this data from inspection records for the bridges was used to obtain parameters for a Weibull distribution to predict the lifespan of a concrete bridge. The authors note that this approach underestimates the lifespan of the bridges, as renewals are most commonly carried out before a bridge reaches a dangerous state. It was found that the Weibull distribution was a good fit for lifetimes of the Dutch concrete bridges using the method of Bayes factors.

Agrawal et al compared a reliability method to model degradation of bridge condition with Markov modelling (Agrawal, et al., 2010). The Markov model was formulated with 7 states, using a discrete time step. The reliability model also used 7 states, with the sojourn time set as the mean time computed from a Weibull distribution. The later model could have been formulated as a semi Markov model, but was instead solved via Monte Carlo. Data from inspection of the American state of New York's bridge stock was used to obtain the parameters required for both models. Various structural elements were modelled using both techniques. It was found that Markov generally provided higher estimates of element lifespan/condition. The authors suggested that this may be due to different data processing techniques used for each methodology, but no evidence was provided as to the most accurate model.

A lifetime analysis was performed on UK railway bridges by Le and Andrews (Le & Andrews, 2016). In this study they used maintenance records to fit Weibull parameters for the time to bridge to degrade from an as new condition, to good, poor and very poor states. As in previous studies by Le and Andrews, this was enabled by inferring the degree of degradation of a bridge element by the cost and duration of the repair. As well assuming the element was restored to as new condition by previous repair work. Some of the Weibull parameters obtained were dubious, with beta values less than 1. This implies that the probability of reaching the degraded state reduces with time. The authors note this may be due to

insufficient data points, resulting in the sample Weibull parameters not reflecting the underlying population Weibull parameters.

The models reviewed here have all been highly individualistic, unlike the Markov models presented previously, the lifetime analysis papers all use their own frameworks for constructing and analysing their models. This has made interpretation and comparison of the models difficult. There are examples in both Markov models, and life time analysis models of the application of corrective maintenance and renewals. The lifetime analysis model literature contains examples of preventative maintenance modelling by both Frangopol et al (Frangopol, et al., 2001) and Yang et al (Yang, et al., 2005). There are no examples of opportunistic maintenance modelling in the lifetime analysis papers, but there is an example with Markov by Le and Andrews (Le & Andrews, 2013). As such, it is not clear if either Markov or lifetime analysis methodologies offer a more comprehensive integration of maintenance into degradation models.

The focus of many of the lifetime analysis papers was to investigate the costs of maintaining a large inventory of bridges. Here lifetime analysis methods appear to have an advantage over traditional Markov owing to their lack of state space explosion experienced when modelling large numbers of bridges with Markov models (Schere & Glagola, 1994). Markov models are capable of modelling bridges whose degradation history deviates from the average bridge (Jiang & Sinha, 1989) but no lifetime analysis papers have explored this. However, as the aim of many of the lifetime models presented is to explore management of bridge inventories, representing only the average bridge is acceptable for their purpose.

3.1.3 Petri Net models

Le and Andrews were the first authors to publish a Petri net model for degradation of bridge elements (Le & Andrews, 2016). The model used a coloured Petri net formulation. This enables the model to include any number of elements, without increasing the number of states required to represent the model. The model is analysed using a Monte Carlo method, where the mean result of the model converges with repeated simulations. This model departs from previous examples by modelling quantified levels of degradation, instead of generalised condition states which could reflect any variety of forms of degradation. The aim of this was to create a more accurate representation of the condition of the bridge, and to enable more detailed integration of maintenance activities and effects.

The element degradation models are highly detailed. The concrete degradation module models surface spalling, deep spalling and rebar corrosion using discrete states representing

the percentage area of the element affected. The transitions controlling progression between states use Weibull distributions, the parameters of which depend on the condition of the element's waterproofing, expressed as three states. Rebar corrosion is modelled as dependent on deep spalling, by using inhibitor arcs, rebar corrosion degradation states cannot exceed that of deep spalling. The same relationship exists between deep spalling and surface corrosion. The steel and masonry element degradation modules are similarly detailed.

An inspection and maintenance module models' periodic inspections. Inspections reveal degraded states and schedule maintenance. Maintenance scheduling times are modelled using a stochastic process. The time for maintenance to complete, and expense incurred, is proportionate to the type and severity of the degradation of the element and is reduced for any elements repaired simultaneously reflecting the cost savings from opportunistic maintenance. Repairs always restore the element to an as new condition. Should the element degrade significantly between the inspection that revealed the degradation state and the maintenance commencing, the repair action will be cancelled. This reflects the highly constrained nature of railway infrastructure repair. It is generally not possible to extend the possession time of railway infrastructure if it becomes apparent the initial possession time is insufficient to complete the repair. The number of maintenance actions that may be performed on an element are limited by type, once exceeded the only option available is to replace the element. This is to prevent an implausible scenario where regular application of minor maintenance can keep an element in service indefinitely.

The model creates predictions of the degradation state of each bridge element over the lifespan of the bridge. It also predicts maintenance volumes, which are used to create an estimate of maintenance costs over the bridge's lifespan.

A later paper by Le et al used coloured Petri nets for bridge degradation modelling (Le, et al., 2017). The model presented shares many similarities with the previously reported model. The key difference between the models is that this model uses generalised condition states, rather than detailed descriptions of actual degradation states. Element condition is expressed as one of four states from new to very poor as per previous papers by the same authors. Protective coating of each element is modelled as one of five states from new to completely ineffective. The Weibull transition parameters modelling the sojourn time between element condition states are conditional on the condition of the protective coating.

Yianni et al also published a paper using coloured Petri nets for bridge element degradation (Yianni, et al., 2016). Their model shares many common features with the models published by Le at al. The biggest difference lies in the degradation modelling module. Yianni's degradation module is based on the severity extent (SevEx) inspection approach used by Network Rail. SevEx expresses element condition in terms of both the severity of the degradation and the extent of coverage of the element. There are 7 levels of severity and 6 level of extent. This is similar to the detailed degradation level module presented by Le and Andrews (Le & Andrews, 2016). The transitions between states are modelled using the exponential distribution. The transition parameters were determined using condition and maintenance records from Network Rail. Testing of the degradation module using spilt sample analysis found the module to be accurate.

As per previous Petri net models, an inspection and maintenance module reveals element conditions and schedules maintenance. Unlike previous models, perfect repairs do not occur. Maintenance is modelled as a measure to control and improve element condition, rather than restore to a perfect condition. The modules uses variable inspection frequencies, where inspection frequency increases in proportion to the level of degradation. This is employed as a risk management measure, as maintenance interventions do not restore elements to a perfect condition, inspection frequency is increased as elements degrade so that critical levels of degradation are discovered promptly. The model produces whole lifecycle cost and condition predictions.

Fuzzy logic represents Boolean systems where there are degrees of truth to information to account for uncertainty or imprecision where a simple true or false value would be inappropriate. Chiang et al developed a fuzzy Petri net for condition assessment and maintenance decision making for concrete bridges (Chian, et al., 2000). This allowed a decision making Petri net to incorporate the uncertainty in the assessment of the condition of a concrete bridge considering defects such as: cracking; spalling; delamination; settlement; and scour, in addition to the uncertainty in the following decision on the appropriate maintenance action – both examples of aleatory uncertainty. It was developed using a Petri net to allow concurrent processes and rule-based reasoning. The model was applied to a bridge in northern Taiwan, successfully interpreting condition and making maintenance decisions.

Applications of Petri nets to modelling bridge degradation have occurred only recently, therefore there is no development of the state of the art beyond the initial models published.

Despite this, the Petri net models reviewed are the most feature rich of any of the modelling methodologies considered here, containing:

- Conditional degradation rates
- Repair limits
- Element level modelling
- Opportunistic maintenance
- Minor, Major maintenance and complete renewal of elements
- Servicing
- Degradation classification including both severity and extent
- Maintenance failures
- Variable inspection frequencies
- Stochastic maintenance scheduling

Petri nets have shown considerable advantages over Markov models in terms of the size of the state space. The Markov models reviewed require a separate set of states for each element included in the model, whereas Petri nets need only add additional tokens which utilise the same single set of states. The additional logical constructions defined in the Petri net formulism also reduce the number of states required to fulfil specific functions. This is apparent when considering Le and Andrews Markov implementation of opportunistic maintenance, requiring an exponentially increasing number of states for each additional element which may be maintained opportunistically (Le & Andrews, 2013). Their equivalent Petri net model processed these interactions using only additional logic within a transition (Le & Andrews, 2016). A similar geometric state space explosion may be expected if Markov models were applied to modelling specific degradation process states, rather than general condition states.

Compared with Petri nets, integration of maintenance into lifetime analysis models appears underdeveloped. Lifetime analysis models include basic features of maintenance such as servicing, repairs and renewals, but lack advanced features such as opportunistic repair, maintenance failures and effective repair limits. The models by Frangopol et al (Frangopol, et al., 2001) and Yang et al (Yang, et al., 2005) are solved via Monte Carlo methods, and as such will have few modelling limitations preventing their extension to include advanced maintenance practises with equivalent accuracy to that of Petri nets. However, without the clear formalisms of Petri net modelling, doing so may result in models which are difficult to easily compare and expand upon. Something which is of importance when using a library of

single asset models to form a whole asset model of an entire route, and hence a poor choice of modelling methodology to underpin a whole asset modelling framework.

Many of the lifetime analysis models were developed to aid management of entire inventories of bridges, aided by their low computational burden. As Petri nets have not been applied to modelling entire bridge inventories it is not clear if it is feasible, or if they have any advantages. However, it does appear that Petri nets are more suited to whole inventory modelling than Markov methodologies, as they do not suffer from the state space expansions that limit Markov implementations.

3.1.4 Bayesian Network Models

Calvert et al developed a Dynamic Bayesian Network model for bridge asset degradation (Calvert, et al., 2020). The model was applied to masonry abutments to predict the probability of spalling brickwork, degradation of pointing, movement of blockwork, cracking and the formation of hollow voids. The use of a Bayesian Network to model these forms of degradation allows the model to account for the inter-dependence of these degradation processes. For example, the probability of hollow voids occurring is conditional on the presence of spalling brickwork and pointing degradation, the probability of movement of blockwork is then in turn conditional on the probability of hollow voids. Analysis of the model shows how the probability of these states occurring changes over time, assuming no maintenance is carried out.

Maroni et al developed a Bayesian Network model to estimate the depth of flood included scour on bridge abutments (Maroni, et al., 2020). The model aims to account for the large amount of uncertainty in assessing and modelling scour due to flooding. The authors note that assessing flood scour damage often only takes place long after the flood event has passed for reasons of safety and practicality, and that as a consequence it is difficult to determine when the damage occurred during the flood event. The Bayesian Network is structured around a series of hydraulic equations to calculate an estimate of scour due to channel constriction and formation of vortices, with input variables forming the root nodes, and child nodes representing key steps in the calculation. The root nodes use normally distributed probability density functions allowing the uncertainty in their estimation to propagate through the various calculations, ultimately calculating a probability distribution for scour depth rather than a single value. The unusual nature of this Bayesian Network necessitates a Monte Carlo based solution.

3.1.5 Bridge Modelling Conclusions

This review has shown that bridge degradation modelling is a highly developed, with publications of stochastic models spanning three decades. Despite this, there is little modelling development of masonry bridges or elements. Newer models feature increasingly sophisticated integration of maintenance within the degradation processes, enabling accurate whole lifecycle models and exploration of the effects of different maintenance policies. No modelling methodology appears to have a clear advantage in modelling degradation alone.

Petri nets appear to be much better suited to degradation modelling with integrated maintenance, owing to their standardised formulation allowing detailed dynamic processes to be represented concisely. Solving Petri nets via a Monte Carlo method also relaxes a lot of the assumptions required for Markov models. These assumptions were not found to affect degradation modelling, but were shown to be detrimental to modelling rule based maintenance strategies.

As modelling the asset management of bridges is particularly well developed, it provides a good comparison of modelling methodologies with respect to their suitability for use at the core of a whole asset modelling framework. Markov models have been shown to have increasing computational requirements as the number of states expands. Given a whole asset model of a route will contain many individual assets, and therefore states, it is unlikely that Markov models will be computationally feasible. The life time analysis models lack coherent features, as such they will have poor inter compatibility, hindering their use in a library of models. Petri nets have shown neither of these weaknesses in the literature reviewed thus far.

3.2 Rail Track

The UK's rail network uses primarily ballasted track. This type of track consists of a deep bed of aggregate, known as ballast, upon which sits rail track. The track itself comprises rails and sleepers, attached together using clips. This system may decay in a multitude of ways (Litherland, 2019):

- The rails suffer from physical wear and fatigue, which without repair or replacement leads to rail fracturing and potential derailment.
- Clips affixing rails to sleepers can fail, with sufficient clip failures rail gauge may be affected.

- The sleepers themselves may degrade, with the degradation mode being specific to the sleeper's construction material.
- The aggregate comprising ballast will degrade and settle overtime, this will induce miss-alignment in the rails affecting ride quality when minor and requiring speed restrictions when major.

3.2.1 Markov Models

Podofillini et al developed a continuous time Markov model to determine the probability of a rail fracture under different maintenance strategies (Podofillini, et al., 2006). The paper focuses on automated ultrasonic inspections, and the optimisation of their implementation. These ultrasonic inspections are modelled as an imperfect process where any given inspection may fail to reveal a degraded rail. The model also features a common cause inspection failure module, where all ultrasonic inspections may fail. The rail itself is modelled with five degradation states: New; minor damage requiring a scheduled repair, moderate damage requiring immediate repair, major damage requiring a line closure, and fracture. These have corresponding states for discovered and undiscovered conditions, with the exception of fracture which is assumed to always be immediately discovered.

Continuous Markov models were developed by Shafahi et al (Shafahi & Hakhamaneshi, 2009) and Bai et al (Bai, et al., 2015) to model the change in track condition indexes over time. Shafahi et al used an Iranian railway case study to model the change in Combined Track Record over time, finding their Markov model superior to a regression method for predicting future states. Bai et al performed similar work, modelling the change in Track Quality Index using a Chinese rail line as a case study.

The most detailed Markov model reviewed was presented by Prescott and Andrews (Prescott & Andrews, 2013). This model focuses on ballast and track deviation, and features four condition states for deviation, relating to actions initiated as a consequence of the condition. The model also considers the known condition state, which may only be equal to or a better condition than the actual condition. This requires ten states to model. Inspections occur periodically, which updates the states in the model.

The track deviation may be restored via tamping, a maintenance procedure which repositions ballast. As with Andrew's related work on Petri nets, this deteriorates the ballast resulting in faster development of track deviation after each tamp, up to eight tamps. This is modelled with an additional set of ten states for each number of previously completed tamps, each with a different transition probability rate reflecting a faster degradation rate.

The authors comment that the model was not computationally intensive to solve. However, they note that should additional maintenance procedures be included the number of states within the model will increase greatly which could cause a computational problem.

Sharma et al developed a Markov Decision Process model for rail track maintenance (Sharma, et al., 2018). The model features five condition states, relating to Track Quality Index value bands. From states two to five a minor maintenance action may be carried out, restoring the track condition to one state better than the current state. From states three and worse, there is a major maintenance option which will restore track condition to at least two states higher, and potentially further with the probabilities specifically defined for each state. The decision making was optimised using a value iteration algorithm, finding a set of decision policies for each state which saved an estimated 10% in maintenance costs over the original maintenance policies. However, this optimisation does not assign a cost to degradation of the asset other than expensive and immediate repair of serious defects. Further, the optimisation does not consider derailment risk arising from serious defects, and so it is not clear how optimal the solution found would be if applied to a real-world scenario where minimising risk and maintaining the asset base is valued as highly, if not higher than maintenance costs.

3.2.2 Petri Net Models

The first Petri net model for rail track asset management was published by Quiroga and Schneider (Quiroga & Schnieder, 2011). The purpose of this model was to explore the effectiveness of fault and condition reporting systems. Multiple asset types are included in the model, however each uses the same asset degradation state structure, with only transition parameters changing between assets. This state structure features four states: working, undiscovered degraded state, discovered degraded state and failed. Unlike prior rail degradation Markov models, an asset is not restricted to moving through these states sequentially. The model features a variety of stochastic transitions, whose parameters vary.

Approximately 19thousand assets were modelled, based on a case study of a 100km French high speed rail line. Though the asset degradation Petri net was the same for each asset, the transition parameters are varied based on asset type and usage level. Maintenance was modelled through an unspecified decision algorithm, featuring preventative and corrective maintenance, and some consideration of staff management. Despite the number of assets modelled, simulation was not found to be computationally intensive when solved via Monte Carlo.

The next iteration of Petri net track model was presented by Andrews (Andrews, 2012). This is a stochastic Petri net model, solved via a Monte Carlo method. It considers track deviation and rectification via tamping of short sections of track. The model may be duplicated enabling modelling of longer track sections, comprised of shorter independent track sections. It features four condition states, related to the management actions they trigger. The first state represents track in good condition requiring no action, the second state triggers maintenance, the third speed restrictions and the final state triggers a line closure. These states are discovered via periodic inspection. The model is used to evaluate given maintenance strategies by predicting maintenance volumes, usage restriction and future condition states.

This advances upon work by Quiroga et al as the inspection, maintenance and renewal processes are modelled within the Petri net. As per the Markov model published by Prescott and Andrews, the degradation rate of the asset is also conditional on the number of previous tampings carried out to restore track geometry. Andrews presents some novel extensions to stochastic Petri nets, such as the reset transition, convolution transition and conditional transition.

Rama and Andrews later develop a stochastic Petri net model of degradation and maintenance for sleepers and rails (Rama & Andrews, 2015). The assets are modelled using multiple condition states as per previous models. Preventative maintenance occurs at periodic intervals with reactive maintenance occurring as required. The model outputs predictions of maintenance volumes, future condition states and usage restrictions.

This work advances upon previous works via the use of a hierarchical structure. Each element of the model, such as rail degradation or maintenance is modular, fitting within the hierarchy. This enables coherent configuration of line models comprising smaller track sections. This further enables modelling of opportunistic maintenance by allowing track maintenance to trigger maintenance actions in other nearby track sections. Previous work by Quiroga had achieved this, but by relying upon custom software which may not be easily replicated, adapted or used within a library.

In their PhD thesis, Audley, published a track model which builds upon the model published by Andrews (Audley, 2014). This model was similarly based on track geometry deviation and rectification, using a stochastic Petri net, but provided several advances on the previous work. The model incorporates tamping, and an additional form of maintenance,

stoneblowing. Both of which serve to restore track geometry, but also influence stochastic degradation parameters. In this model, these parameters are also conditional on line speed.

Audley includes stochastic maintenance effectiveness into his Petri net model. With increasing repetition of maintenance, subsequent maintenance becomes less effective. This is modelled by an increasing probability that the maintenance action will fail to restore the track section to a 'good' state, with maintenance eventually becoming largely ineffectual. The model includes a high-level extension referred to as a Decision Making Transition. This is used to create a scheduling system. These transitions consider various states to decide on the type of maintenance action to be taken. Conventional Petri nets control the availability of maintenance equipment. This fulfils the function of the decision algorithm first presented in work by Quiroga, but does so in way that is closer to the Colour Petri net paradigm.

Shang et al developed a Coloured Petri net model for track geometry degradation over multiple track sections (Shang, et al., 2017). The paper uses the model to explore speed restrictions as a strategy to reduce and delay maintenance actions. This model was constructed using a hierarchy. The asset model comprises: asset states and degradation; maintenance processes; and operational rules which govern usage, such as line speed. A management model comprises a further three aspects: maintenance decision making; inspections; and maintenance grouping.

Degradation is modelled using a gamma process, where the passage of each train randomly increments a continuous measure of track degradation. The parameters governing the stochastic decrease in condition are dependent on the speed of the train. This type of degradation modelling is not possible within the stochastic Petri net paradigm without a significant extension as it is fundamentally continuous. A continuous variable could be approximated with a conventional Petri net by using a large number of tokens, sufficient to gain the required level of detail. However, the number of tokens output when a transition fires is fixed.

The model incorporates several forms of maintenance to restore track geometry: tamping; stone blowing, rail lifting and renewal. The former three methods are considered imperfect, and may not restore the geometry to a perfect state with a given probability. Maintenance occurs following a decision taken by the system model, and occurs after a stochastic delay. Maintenance actions may be delayed or grouped together in order to reduce the volume of maintenance carried out. These processes are modelled entirely within the Coloured Petri net paradigm.

A comprehensive whole asset Petri net model was presented by Litherland (2019) in their PhD thesis. This stochastic Petri net models many different assets classes along the Bletchley Park line. Assets modelled included: ballast; sleepers, rails; track circuits; switches and crossings; tunnels; and signalling assets. The work focuses mainly however on the track assets. The model was constructed using a hierarchical framework, similar to work by Rama and Andrews, to enable the efficient construction of a coherent model despite the diversity of assets and processes (Rama & Andrews, 2015). The hierarchy comprised system level modules which control the availability of maintenance resources and line usage restriction. Beneath this level is the track level comprising each defined section of track. At the lowest level, each track section contains degradation and maintenance modules for each asset present in a given section of track.

A variety of maintenance processes are modelled including: tamping, stoneblowing, rail grinding and replacement. The model includes opportunistic maintenance actions, where maintenance carried out within a track section can be extended to include other assets within the section. A range of decision making transitions control which maintenance actions are to be undertaken. Once a maintenance action has been decided by the model, it occurs after a stochastic delay, determined using a log-normal distribution. Maintenance resources are limited within the model, should insufficient maintenance resources be available when the stochastic delay is completed the maintenance action will wait until the required resources are available. The model does not consider resource availability when generating the stochastic delay, nor does it directly consider the mobilisation time for the resource, though this is arguably considered in the stochastic maintenance delay.

The novel level of complexity of this whole asset model created computational challenges in solving via a Monte Carlo method which have not previously been encountered. Litherland mitigated this computation issue by simulating the model on a GPU, increasing simulation speed by a factor of approximately ten.

Sachan and Donchak developed a stochastic Petri net for asset management of railway switches and crossings (Sachan & Donchak, 2020). The model considered degradation, inspection, servicing and renewals of the switch system, allowing predictions of the number of maintenance interventions needed over the lifetime of the asset. Asset degradation was modelled using Weibull distributed stochastic delay transitions. Fuzzy numbers using a triangle membership function were applied to the Weibull scale factor to account for uncertainty in this parameter. Fuzzy numbers enable modelling the uncertainty of variables,

a membership function maps a weight to the value of a variable using a membership function. This enables inclusion of uncertainty in parameters or measurements rather than true or false conditions. This uncertainty could arise from epistemic uncertainty owing to the quantity or scope of the data used to derive the parameter, or also from aleatory uncertainty from human qualification of when a switch has degraded or failed.

Bayesian learning been applied to Petri nets for the purpose of railway asset management modelling. Chiachío et al developed a track degradation model using their Plausible Petri Net methodology (Chiachío, et al., 2018). The purpose of this model was to demonstrate a maintenance decision expert system which learns and adapts to uncertain asset condition data via a Bayesian learning method to improve its decision making over time. A physics-based degradation model was used to model track degradation for each loading and unloading cycle, with track deflection periodically measured. There is uncertainty in the track deflection measurements preventing them be using in conjunction with a rigid set of maintenance rules corresponding to their values.

Based on the measurements obtained, the Plausible Petri Net makes maintenance decisions based on the degree of belief of the condition of the track, which is updated as additional condition data in the form of deflection measurements or detailed inspections are carried out following a decision to inspect the asset. The results of the modelling shows that as the simulation progresses, the accuracy and uncertainty of the track condition prediction improves.

3.2.3 Track Modelling Conclusions

The literature review of asset management models of track assets yielded many models. Markov models were often limited to single assets with few maintenance processes. This appears to be due to an exponential increase in complexity created as additional processes and the associated states are included. Petri net models were the most developed with little scope for improving the modelling of track specifically. Whilst Petri net models are not immune to scaling issues, many examples in literature of large multi-asset models using Petri nets were found.

The most well developed model was presented by Litherland, as this work included the most asset classes and maintenance processes. However, this work and others are limited by their reliance on stochastic maintenance scheduling, where maintenance occurs after a stochastically generated delay. This is a potential source of inaccuracy when modelling the effects of maintenance strategy, as the delay between identifying a need for maintenance

and carrying out that maintenance is constrained by the availability of resources. Resource availability is dependent upon the volumes of maintenance required at any given time, which is in part dependent upon the maintenance strategy. It should therefore be expected that maintenance strategy influences the time till maintenance is carried out.

Single asset models are forced to assume a fixed or a stochastic maintenance delay as resources are frequently shared between multiple assets on a rail line which are not included within the single asset model. The whole asset model created by Litherland enables the development of a deterministic maintenance system. As all assets on the line feature within the model, an accurate picture of the pending maintenance actions can be created. This can be used to allocate limited maintenance resources based on their availability, urgency of work and time to mobilise, rather than using a randomly generated maintenance delay time.

Such a method will increase the complexity of an already complex model making its implementation a non-trivial task. Further to this, Litherland already reports computational limitations simulating their model on contemporary computing hardware. The addition of such complexity may necessitate major changes in methodology to incorporate.

3.3 Electric Traction Transmission Equipment

Electric Traction Transmission Equipment (ETTE) refers to the infrastructure which conveys electrical power to electric trains. There are two general types of ETTE used for this purpose: Overhead Line Equipment (OLE) where the electric delivery is handled above the train; and third rail systems, where the electric source is an additional rail underneath the train. OLE is the industry preferred method for connecting electric trains to a source of electric power, as it is more cost effective and can provide more power than third rails. In the UK 63% of electrified line uses OLE (RSSB, 2011).

No examples of Markov models for ETTE were found, and only one example of a Petri net modelling asset management of ETTE. Kilsby et al published a paper describing a Coloured Petri Net for modelling asset management of OLE (Kilsby, et al., 2018). This model considered multiple OLE components: catenary wires; droppers; insulators; registration equipment; return structures and structural supports. This diverse selection of components was incorporated into the model using a hierarchal framework which enabled specification of the precise number, location and history of the component.

Each component has multiple associated degradation states. These vary by component, but typically comprise: working; maintenance required; and failed. These states may be

discovered during inspections. Each component is modelled with only a single degradation process. Degradation rate may be decreased by preventative maintenance which corrects deviations in the OLE. Upon discovery of a degraded or failed state maintenance is scheduled with a stochastic delay. Nearby failed OLE components may be repaired or inspected opportunistically subject to time limitations.

The model can be configured for specific OLE layouts and initial asset condition states, allowing it to be easily applied to real world assets. Also modelled are maintenance strategies including inspection frequency, maintenance response time and renewal conditions. The model predicts: maintenance volumes, failures, costs and future condition of the OLE assets. ETTE has seen limited modelling in academic literature, however the model by Kilsby is feature rich leaving little scope for meaningful additions.

3.4 Level Crossings

Level crossings are a railway asset class which allow roads or foot paths, and rail lines to intersect. This facilitates the safe passage of road, or pedestrian traffic over inherently dangerous rail lines. Crossings can either be passive (76% of UK crossings), where a user mostly relies on their wits, good judgement and telephone equipment to safety traverse the crossing, or they can be protected (24% of UK crossings), where a mixture of light signals and barriers may be provided to inform of the user when to wait, and when it is safe to traverse the crossing.

There are three main types of level crossing within the UK; passive, manually controlled and automatic, described below.

Passive level crossings consist of a gate and road surface across rail track, usually with a phone to contact a signalman to determine if it safe to cross – a particularly important feature considering the time required to open both gates and drive a vehicle across may span minutes during which time a previously unseen train may be bearing down on the crossing. At present there are approximately five thousand passive level crossings on the UK network, responsible for around four and a half fatalities per year (Evans, 2011).

Manually Controlled Crossings use road traffic lights and full barriers which close the road entirely to protect road users from trains. These are controlled via signalling staff who monitor the crossing via CCTV to determine if it the crossing is clear and operating correctly. There are currently approximately 800 of these crossings in use on the UK rail way,

responsible for around one fatality every two years (Evans, 2011). An example of a manually controlled crossing can be seen in Figure 4-2.

Automatic Half Barrier Crossings (AHBC) use both road traffic lights and a half barrier to warn road users of an oncoming train and partially block the road leading into, but not out of, the crossing. Unlike the Manually Controlled crossing type, this crossing is controlled using a complex electromechanical control system with limited human oversight. There are around 500 of these crossings currently in use in the UK, typically deployed in rural areas where they are expected to see less road traffic. This type of crossing is responsible for around three deaths per year, giving it the most adverse risk profile of any crossing type on a per asset basis.

3.4.1 Risk Modelling

In the late 1980s the first major review into safety at automatic crossings was published by Professor Stott, automatic crossings having been in service since the 1960s. The review centres on Automatic Open Crossings (AOC), a now obsolete type of crossing where only road traffic lights warn road users of an oncoming train. Stott discovered that the majority of accidents at these crossings were caused by bad actor behaviour, drivers who knowingly decided to run the red lights despite being cognisant of the risks.

Stott found a non-linear relationship between traffic volume and accident rate. Initially, increasing volumes of traffic corresponded to an increased risk of a collision between road vehicles and trains. This is to be expected, as an increased traffic volume creates additional chances for a road vehicle and a train to collide. However, Stott also found that beyond a point, further increases in road traffic decreased the collision risk. It was hypothesised that this effect was due to a combination of two factors. Drivers may be reluctant to commit driving offences when their actions are likely to be witnessed, as would occur at highly trafficked crossings. Secondly, the presence of other road vehicles may impede dangerous behaviour as a driver will have to traverse the length of any vehicles queuing to pass the crossing, and the crossing itself all whilst on the wrong side of the road (Stott, 1987).

A paper by Ghazel presents a model for predicting collisions at level crossings using stochastic Petri Nets (Ghazel, 2009). The traffic simulation module considers traffic jams as the sole cause of collisions. Impatient drivers may enter the shared area of level crossing despite their exit from the shared area being blocked by traffic. Should a train pass the crossing before the road vehicle can exit the shared area a collision will occur. Stochastic transitions control the

generation of traffic jams, the arrival of road vehicles, and whether a road vehicle will enter the shared area of the crossing without a clear exit path.

In their PhD thesis, Ishak presents a further stochastic petri net model for determining level crossing collision risk (Zaharah, et al., 2011). Uniquely, this model considers distinct traffic flow regimes, from free flowing traffic, to near jam conditions via six graduations. This model factors in the effect of traffic approach speeds and even considers acceleration and deceleration times for different types of vehicle. This is used to determine if a road vehicle can react fast enough to avoid a potential collision at an unprotected crossing.

A model included in the British Standard for petri net modelling dependability analysis includes a simple collision model (BSI Group, 2012). The arrival of vehicles and trains are modelled stochastically. Approaching cars are categorised as: good actors who will always obey the instruction of the level crossing system; as bad actors who will ignore the safety systems if they cannot see a train passing the crossing; or as bad actors who will always ignore the safety systems regardless of whether a train is visible or not. This selection is modelled using decision making transitions which make a selection depending on the value of a random number.

The operation of the crossing system itself is modelled as either active or inactive. The crossing may fail in a safe manner where the system remains active in the absence of an approaching train. And may fail in a hazardous manner, where it fails to operate despite an approaching train. This enables collisions to occur where either the road driver or the crossing is at fault. Unlike the model by Ishak, driver reaction time or vehicle stopping distances do not factor into collisions. A collision occurs simply when both road vehicle and train occupy the shared space at the same time with no evasive actions modelled for either.

A novel paper presented by Medjoudj uses a radically different modelling approach to those discussed thus far using standard Petri Nets (Medjoudj & Yim, 2007). The simulation of this model begins at the accident state where a train and a vehicle have collided, from there the simulation runs in reverse exploring the potential causes of the collision. This contrasts with the other models, where collision between train and vehicle occurs stochastically after many vehicles/trains have been simulated as entering and leaving the crossing area. The purpose of this model is to discover novel causes of failure in level crossing systems that can result in a collision. However, the level crossing system modelled was very simple, allowing cut sets resulting in collision to be determined through inspection alone, with nothing to suggest how

effective this method would be when applied to a system complex enough to benefit from a methodology to locate causes of collisions.

3.4.2 Level Crossing Modelling Conclusions

This literature review found many risk models for level crossings, of which the most relevant have been described. Whilst predicting collision risk is an important part of managing level crossings, no models were found for general asset management aspects such as degradation or maintenance. Of the risk models reviewed, none were particularly detailed, with many focusing on the exploration of isolated causes of collisions only. This presents the possibility of developing both a novel level crossing asset management model. And, a novel collision risk model which incorporates appropriate elements of the models reviewed here which has not be previously attempted.

3.5 Earthworks

Earthworks are a product of rail necessity, as conventional trains are only able to traverse shallow gradients. Typically, far shallower than natural topography. To enable the construction of efficient railways earthworks are required to reduce natural steep gradients to an acceptable value. When the land under a rail line must be raised, earth is deposited to create an embankment. Cuttings are the opposite of embankments. Where the natural landscape is too high for the rail line the earth is cut away allowing the rail line to pass through at the appropriate height. Failure of an embankment can result in a loss of support of the track lain above it. Cutting failure may result in earth blocking a rail line resulting in derailment and collision.

The majority of railway earthworks in the UK were constructed during the Victorian era when the railways were first built. This creates general problems in asset management of earthworks:

The construction of the UK railway and its earthworks pre-dates the ground-breaking works of Karl von Terzaghi, widely credited as the father of geotechnics. Prior to whom, geotechnical engineering practice was generally based on trial and error. Because of this many of the railway earthworks currently in use on the UK rail network may not meet current design standards due to insufficient factors of safety (Gunn, et al., 2018).

Modern earthworks are often constructed using homogenous materials. This enables simple calculation of loads and resistances allowing a factor of safety to be computed. However, the Victorian era embankments pre-date this practise. Embankments from this era typically

comprise whatever material happened to be available, complicating any present day attempt to calculate factors of safety (Gunn, et al., 2018).

The age of UK earthworks in itself creates challenges due to degradation and failure of earthworks. This creates significant costs relating to investigation and repair, with Railtrack PLC spending £11million per year (Railtrack, 1999), and London underground spending £70milion over five years (London Underground Limited, 1999) as examples.

3.5.1 Markov Models

A number of Markov based deterioration models have been published for earthworks degradation. Tinoco et al published a Markov chain based model to predict deterioration in rock slopes (Tinoco, et al., 2015). The model uses five condition states based off the Slope Quality Index (SQI), a subjective measure of visual deterioration of a rock slope face (Pinheiro, et al., 2015). Tinoco does not apply his model to any available dataset, instead using synthetic data to validate the model, leaving its effectiveness when applied to real slopes uncertain.

Network Rail recently developed their Strategic Decision Support Tool to support asset management decision making for their earthworks portfolio. The tool models whole life costs over a total of 100years, providing earthworks condition and maintenance cost estimations in addition to optimisation of an asset management plan (Power, et al., 2016).

Degradation of earthworks assets is performed using a Markov Chain with six states. These states correlate to risk of failure of the earthworks. This is considered on the basis of both condition of the earthwork and its design. This creates some confusion when considering transitions between states. A well designed earthwork in poor state of repair could be assigned the same state within the model as a poorly designed earthwork in good condition. The model would then assume both assets degrade at the same rate, but this is unlikely as the earthworks are both in different conditions.

A paper by Denysuik et al presents a novel Markov model for slope deterioration (Denysuik, et al., 2016). This model has five deterioration states, modelling slope deterioration over a continuous time frame. The model includes a variety of potential maintenance effects which may improve condition state, reduce condition deterioration rate or pause deterioration. The rate of degradation and magnitudes of the effects of maintenance actions were chosen following expert consultation.

3.5.2 Earthworks Modelling Conclusions

Literature review found a small number of Markov models for degradation of earthworks, however none were well developed. The models by Tinoco and Denysuik et al relied upon synthetic data or expert opinion. Whereas the model employed by Network Rail fails to separate failure risk due to poor design from failure risk due to poor condition in its degradation model.

The reason for the lack of development of asset management models for this asset class likely relates to the heterogeneity of earthworks. Factors such as age, materials composition and state, water content, and topology all affect the stability of an earthwork. Some of these factors are expensive or impossible to properly ascertain in-situ. This largely hidden heterogeneity renders data-driven estimation of stability ineffective, limiting the usefulness of any conventional asset management modelling of earthworks via Markov or Petri net model.

3.6 Tunnels

Literature search failed to find any published asset management models for railway tunnels, or related works.

3.7 Signalling

Few asset management models were found for railway signalling systems. Patra and Kumar developed a Coloured Petri net to explore availability of conventional track circuits (Patra & Kumar, 2009). Whilst basic, the model featured degradation, inspection, corrective and preventative maintenance. The authors found maintenance strategies affected availability of the track circuits. Estevan presented a Markov model for signalling system degradation in their PhD thesis (Estevan, 2015). The model featured 11 states covering 5 components. Each component featured a degraded and failed state. The model was used to predict measures of availability and safety for varying degradation rates. Pour developed mathematical optimisation methods for the allocation of maintenance resources (Pour, 2017). These were applied to maintaining ERTMS signalling assets in Denmark.

3.8 Chapter Summary

One of the aims of this thesis is to determine the most appropriate modelling methodology for railway infrastructure asset management, focusing on the application of a modelling framework to allow the creation of models incorporating multiple assets of varying types. This chapter has reviewed academic literature on modelling for each major railway asset category in order to assess both the level of modelling development for that asset, but also

the most appropriate modelling methodologies. From this review, several core processes which an asset management model may feature have been identified, namely: Degradation, inspection and decision making, and maintenance. The applicability and suitability of each of the common modelling methods to modelling these processes is discussed below and used to frame the selection of a methodology.

3.8.1 Degradation Modelling

Review of the modelling literature found many degradation models, highlighting model features which are important to modelling degradation. The asset degradation model is core to asset management modelling. The time step of the degradation model therefore constrains both modelling maintenance decision processes and modelling maintenance itself. Many early Markov models used fixed time steps aligned with inspection and maintenance decision cycles ignoring the variability of inspection and maintenance timing. Later models by authors such as Le and Andrews used continuous Markov chains where the degradation state of the asset could be evaluated at any time step (Le & Andrews, 2013). Petri Net models solved via Monte Carlo have no mathematical limitations affecting time step, and so all the models found in literature could have had their degradation state evaluated at any time regardless of whether this was implemented. The Dynamic Bayesian Network developed by (Calvert, et al., 2020) was limited to a discrete time-step.

Many of the papers identified presented arguments for the use of non-constant degradation rates or probabilities, with Weibull frequently used to model sojourn times between different condition states. Petri nets present no difficulty in adopting different statistical distributions, as the Monte Carlo solution method may simply sample from different underlying distributions. Markov models commonly assume a constant degradation rate, however several examples of Markov models which use other sojourn distributions were found for modelling bridge degradation, referred to as Semi-Markov. The Dynamic Bayesian Network model developed by Calvert et al assumed a constant degradation rate for the root degradation processes modelled, however literature from other fields show that other distributions can be used with Dynamic Bayesian Networks to model asset degradation (Xiaobing, et al., 2018).

Dynamic conditional effects on degradation were identified by several authors as significant factors. Use of conditional transitions in Petri nets enabled authors such as Le and Andrews to model detailed conditional effects of reinforced concrete degradation (Le & Andrews, 2016), where the severity and extent of degradation of the concrete determined the

degradation rate of the steel rebar within. Jiang and Sinha were able to model conditional degradation within a Markov chain model by simply changing the transition rates every time step as a function of the age of the bridge (Jiang & Sinha, 1989), to model older bridges aging faster. Prescott and Andrews created a Markov model for railway ballast degradation which was conditional on the number of maintenance interventions previously applied (Prescott & Andrews, 2013), this was achieved with the use of additional sets of condition states for each number of maintenance interventions carried out, which were manually updated each timestep. Conditional effects are fundamental to Bayesian Networks, and their application to bridge degradation modelling was demonstrated by Calvert et al (Calvert, et al., 2020).

In terms of features, there is little to distinguish between the modelling methodologies discussed when applied to asset degradation. There is however a clear useability argument in favour of Petri nets. As a primarily graphical modelling language solved via Monte Carlo, development of the model itself is simple. Considering the example of sojourn time distributions, changing either a Markov or Dynamic Bayesian Network requires a mathematical reformulation, whereas a Petri net needs to simply sample from a different distribution when simulated.

3.8.2 Maintenance Modelling

When considering modelling of maintenance, clear differences show between Petri net and Markov modelling methodologies. Both Markov and Petri net models predict state probabilities concerning asset degradation, solving Markov models generally involves calculating these probabilities directly. However, the solution method for Petri nets is quite different. Petri net models in literature simulate the asset degradation and management process using random variables, and derive their state predictions by averaging a large number of samples using a Monte Carlo method. This enables Petri nets to model rule-based maintenance, where Markov models are unable to.

This Markov model deficiency is clearest when considering a renewal only maintenance policy, where an asset is allowed to degrade until it reaches its worst permissible condition and is replaced with a new asset. Work by Le in his PhD thesis demonstrated that when a Markov model is applied to a renewal only maintenance policy it fails to predict the oscillations in condition which would be expected as the asset degrades and is subsequently replaced (Le, 2014). This occurs because in a Markov model the maintenance actions are modelled in proportion to the predicted probability of the state which triggers that maintenance being true. There is always some small chance of an asset being sufficiently

degraded, therefore there is always some small amount of maintenance being modelled to correct it. This series of minor maintenances does not reflect the reality of a renewal only maintenance policy, and is the cause of Markov models being unable to accurately reproduce the oscillations in condition which would be expected. Petri nets however, are able to model the oscillations in condition because the degradation and asset management process is simulated. Random variables determine the time till a specific asset degrades, at which point a renewal is modelled by restoring the asset's condition. This is repeated till a Monte Carlo solution converges.

Further examples of the issues caused by this inability to model rule-based maintenance appear in literature. None of the Markov models found in literature applied limits on maintenance volumes, in reality, the actions specified by a maintenance policy are constrained by the availability of resources. The simple graphical notation of Petri nets and lack of complex underlying mathematics has allowed many authors such as Litherland (2019) to model cessation or delays to maintenance due to availability of vital equipment which has not been reproduced in any Markov model found.

There were no examples of asset management maintenance modelling for Bayesian Networks found in literature. However, like Markov models, Bayesian Networks operate solely in terms of probability, predicting the probability of any given state being true. From this, it is likely that Bayesian Networks will suffer from the same deficiencies when modelling rule-based maintenance as Markov models do.

When considering computational cost however, it not clear which methodology is superior. The computational cost of Monte Carlo solutions is high, and authors such as Litherland have reported protracted computation times to solve large models (Litherland, 2019). There are computational issues also present with Markov modelling, each additional state included with a Markov model increases the number of rows and columns in the transition matrix by one, creating a geometric increase in computational complexity as the number of state increases. This issue is compounded by some specific scenarios where modelling opportunistic maintenance policies requires a geometrically increasing number of states as further opportunistic scenarios are incorporated (Le & Andrews, 2013).

3.8.3 Inspections and Decision Making

The last core asset management modelling process considered is inspection and decision making, this is the process of discovering asset degradation and deciding an appropriate maintenance action. Many models simply make a rigid assumption that all inspections

perfectly identify the state of the asset, and that the most appropriate maintenance activity is then carried out, however this may ignore the uncertainty in human assessment and its subsequent effects on decision making.

Several Petri net models were found in literature which incorporated elements of fuzzy logic. Chiang et al developed a fuzzy logic Petri net to model the uncertainty in assessment of concrete bridge elements (Chian, et al., 2000), further modelling the uncertainty in the decision-making process to account for the natural variability in human decision making. Chiachío et al developed a model for track maintenance where inspection data was uncertain (Chiachío, et al., 2018), the model was developed using a novel Petri net type which incorporated Bayesian learning. This model was able to cogently make maintenance decisions despite the uncertainty stemming from the measurements available.

Several examples of Markov Decision Process models were found applied to bridge and track asset management decision making, however none considered the uncertainty in the measured condition states. However, in other fields Markov Decision Processes have been extended to include uncertain states, known as Latent Markov Decision Processes (Madanat, 2000). The track maintenance model by Prescott and Andrews discussed in this chapter incorporated uncertainty in asset state knowledge through the use of multiple states accounting for each possible track condition state and a corresponding known condition state.

3.8.4 Selection of a Methodology

The application of Markov Models, Petri nets, and to a limited extent, Bayesian Networks to railway asset management modelling has been reviewed. Suitability towards modelling degradation, maintenance and inspection processes has considered. The review has found that the differences between the methods for modelling degradation are marginal, with only the flexibility and ease of use of Petri nets standing out. It has further found that there has been perhaps insufficient consideration of inspection and related uncertainty in railway asset management modelling in general. Though, the discovery of Petri net models which integrate Bayesian learning and fuzzy logic into Petri nets found during this review highlights the overall flexibility of the Petri net modelling method.

Maintenance modelling capability was found to clearly differentiate the methodologies. Petri nets were found to be superior to Markov owing to their ability to model rule-based maintenance, enabling them to model maintenance policies such as renewal only, or limit maintenance works by resources presently available. Owing to the similarities in the

probability based computation of Bayesian Networks and Markov models, it is believed they will feature the same deficiencies, though examples were not found in literature. Therefore, it is believed that Petri nets represent the most suitable modelling methodology for the further development of railway asset management modelling.

The most significant draw back to Petri net modelling identified was the computational cost of Monte Carlo modelling, some authors found the computational times to be large and this requires further investigation. This investigation is carried out later in this thesis, focusing on the different computational costs of both regular and colour Petri net variants.

3.8.5 State of the Art for Modelling Railway Asset Types

This review also sought to find assets which require development of an asset management model. It was found that track, bridges, and electric traction transmission equipment had well developed models. Earthworks did not have a well-developed model, but it is currently unclear if asset management models for earthworks are feasible owing to the unique nature of each railway earthwork asset. No asset management models for railway tunnels were found. And only limited models for signalling systems. It was found that there are no asset management models for level crossings, despite their high-risk nature and need for regular inspections and maintenance. Of the three assets identified without sufficient asset management modelling capability, level crossings will be developed further. This decision was made based on the existing level of development of aspects of level crossing modelling and the availability of technical schematics for UK level crossing systems.

4. Level Crossing Systems

The UK rail network contains a diverse range of level crossing types. These range from passive crossings, consisting of a surface that can be traversed by both trains and road traffic and a set of gates, to protected crossings. The latter type of crossing utilises combinations of barriers and lights which prevent access to the shared crossing area whilst a train is approaching or passing. There are multiple types and configurations of protected crossing.

Passive level crossings consist of a gate and road surface across rail track. A phone may be provided to contact a signalman to determine if it safe to cross. This is an important feature considering the time required to open both gates and drive a vehicle across may span minutes during which time a previously unseen train may be bearing down on the crossing. A typical passive crossing is shown in Figure 4-1. At present there are approximately five thousand passive level crossings on the UK network, responsible for around four and a half fatalities per year (Evans, 2011). Often these fatalities are caused directly due to user or signalling person error.

Figure 4-1 Passive Level Crossing

Manually Controlled Crossings use road traffic lights and full barriers which close the road entirely to restrict access to the shared space of the crossing. Typically, these crossings are controlled via a signalman who monitors the crossing via CCTV. A modern variant instead uses a Light Detection and Ranging (LiDaR) to confirm that the shared area is clear. The use of the LiDaR unit serves two purposes, it reduces the chance of operator error, and reduces signalling staff workload, as they need no longer routinely view the CCTV feed. There are currently approximately 800 of this class of crossing in use on the UK railway, responsible for around one fatality every two years (Evans, 2011). An example of a manually controlled crossing can be seen in Figure 4-2.

Figure 4-2 Manually Controlled Crossing

Automatic crossings function largely autonomously. The earliest form of automatic crossing was the Automatic Open Crossing (AOC). This crossing featured only lights to warn road drivers of an approaching train. Many of these crossings exhibited an unacceptable number of fatal road-rail collisions, averaging 1 collision per AOC asset per 20 years. Investigation determined the cause to be varying levels of non-compliance (Stott, 1987). As a consequence of its poor safety record, almost all AOCs were upgraded to either Automatic Half Barrier Crossings (AHBC) or a variation of the Automatic Barrier Crossing Locally Monitored (ABCL). The ABCL design used cheaper, less reliable barrier equipment compared to the AHBC, compensated for by the provision of a protecting signal and low maximum line speed of 55mph (RAIB, 2014).

Both AHBCs and ABCLs use road traffic lights and a half barrier to warn road users of an oncoming train and partially block the road leading into, but not out of, the crossing. An example can be seen in Figure 4-3. Unlike the Manually Controlled crossing type this crossing is controlled using a complex electromechanical control system. There are around 450 ABHC crossings currently in use in the UK, typically deployed in rural areas where they are expected to see less road traffic.

In 2018 the first full barrier automatic crossing was installed near Glasgow. This crossing is similar in design to the cheaper ABCL half barrier crossing, however it features full barriers and a LiDaR unit to confirm the crossing shared space is clear. This crossing was installed on a rail line with a very low speed of 15mph, which sees little road traffic as it only allows access to a small light industrial area.

Figure 4-3 An Automatic Half Barrier Crossing

This work focuses on the AHBC type of protected crossing. The AHBC has the greatest mechanical complexity of all the protected crossings, as in addition to employing both barriers and lights, it is controlled by an electromechanical control system. Any model developed for an AHBC has the potential to be repurposed for simpler crossing types by removing model features not present in any simpler crossing type. In addition to this, the AHBC has the most adverse safety record of any commonly deployed crossing type. The number of fatalities at AHBCs varies significantly by year, however averages at 6.7 fatalities per 1,000 level crossing assets per year (Evans, 2011). This section of the report presents and explores AHBC systems providing the basis for modelling later in the report.

An AHBC's protection systems comprise four sets of lights and two barriers. The lights are positioned on the near side and the far side of the road approaching the crossing for both directions. A single barrier is positioned to block the left hand lane of traffic. The purpose of this barrier configuration is to block entry to the shared area of the crossing, but not egress. These protection systems are activated by track circuits and controlled by a relay logic based control system. Figure 4-4 shows a typical layout of an AHBC.

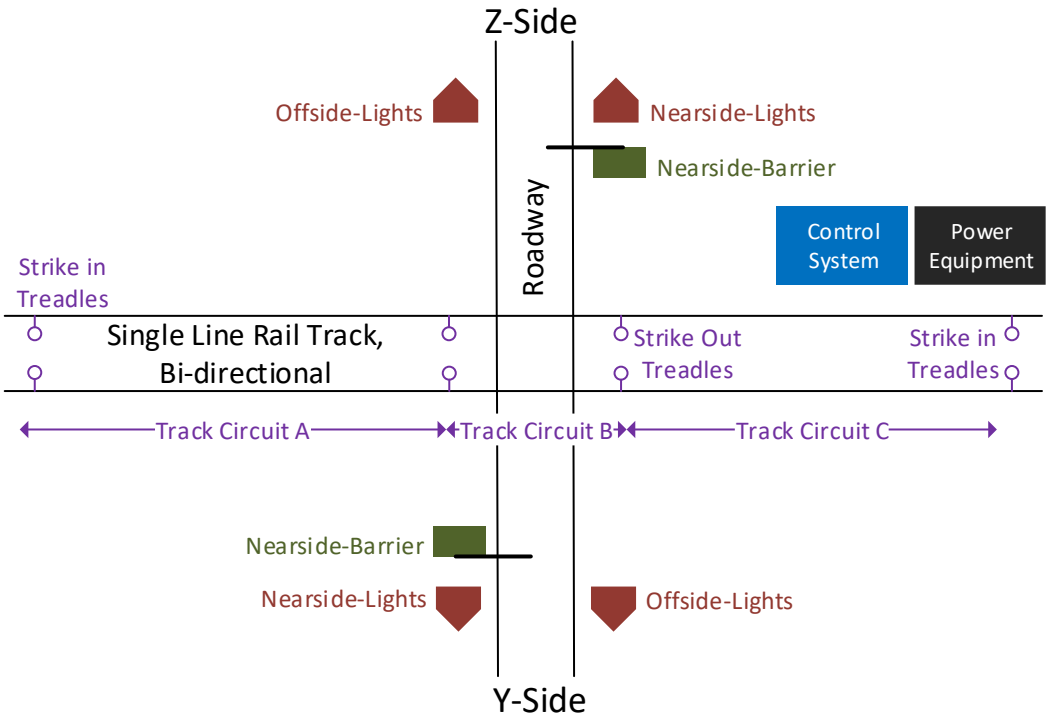

Figure 4-4 Automatic Half Barrier Crossing Layout.

4.1 Protection System

AHBC protection systems follow a fixed operational sequence following the detection of an approaching train. The lights illuminate first warning road vehicle drivers of both an oncoming train, and the imminent decent of the barriers. The full operational sequence is shown in Figure 4-5. The times shown are minimum times, and may be longer in practice. Once the barriers have descended, they remain so, and lights remain active until the train passes the crossing. Track circuit layouts are configured so that the fastest trains using the line will arrive as the sequence has completed. As the track circuit length is fixed, slower trains will cause the protection systems to operate for longer periods. This can be relevant when considering driver behaviour.

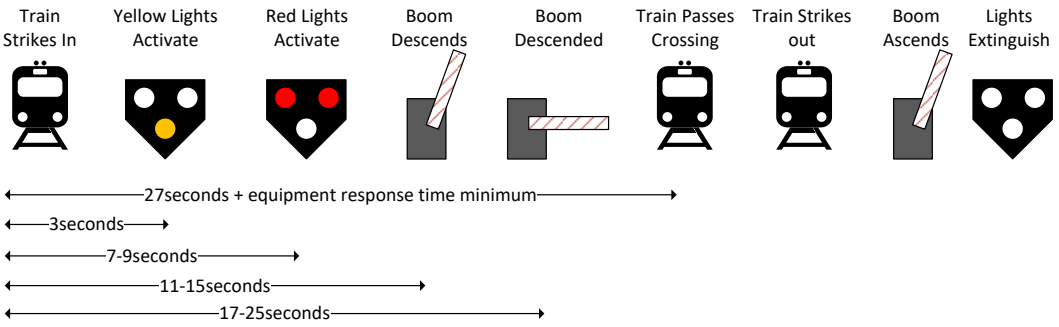

Figure 4-5 Automatic Half Barrier Protection System Sequence (Network Rail, 2011).

Protection system failures can be categorised as either safe, or hazardous. A safe failure occurs when the protection systems fail in a manner which does not significantly increase the risk to the general public. A typical example of this would be the protection systems operating in the absence of an approaching train. Hazardous failures can dramatically increase risk to the general public, where protection systems remain inactive despite an approaching train.

Complete failures of AHBC protection systems are very rare in literature. Only collisions or near misses are publicly reported, so the true rate remains unknown. One near miss occurred at Ufton, Berkshire where a AHBCs protection systems failed to operate. This resulted in a road vehicle being forced to take evasive action to avoid a passing train. At the time the crossing was being manually operated, with the blame for the fault stemming from personnel not following proper procedure (RAIB, 2012). A similar incident resulted in a road vehicle colliding with a stationary freight train. The freight train had come to a stop within the crossing shared space, but due to a design flaw with an older control system configuration the protection systems were not active (RAIB, 2018). There are no recorded incidents of accidents or near misses due to complete inoperation of protection systems at AHBCs caused by mechanical or electrical failure. There is one report of complete failure of the protection systems at an ABCL crossing due to an electrical failure, however this is a much less robust design of crossing as its barriers do not descend under gravity following a power failure (RAIB, 2013).

Failure of a AHBCs barriers in isolation constitutes a partial failure. Compared to a complete failure this can reasonably be expected to produce a moderate increase in risk, as the lights alone can warn road users of oncoming trains. The increase in collision risk due to such an event may be similar to the general collision risk at an AOC which do not use barriers. Failure of a AHBCs lights in isolation is likely to induce a greater increase in risk than an isolated barrier failure. Road users may be surprised by the boom descent as the lights have not

forewarned them. This could lead to a collision with the boom. The boom begins to descend around 13 seconds after the lights illuminate, giving road users less time to observe and react before the train arrives if the barrier alone is considered.

4.2 Track Circuit

Track circuits signal to the control system when a train strikes in, and when it strikes out. These are a critical component of the level crossing system. Failure to detect an train approaching ensures that the crossing will not cycle, however failures of this type are rare as track circuits feature redundancy and are designed to fail in a safe manner.

Figure 4-6 Directional Treadle used in AHBC Track Circuits.

Whether dual or single line, AHBCs are designed to operate bi-directionally. This allows greater flexibility in line use and improves safety. The track circuit layout for an AHBC is shown Figure 4-7. Three track circuits are used for each rail line: one up the line from the crossing, one down the line from the crossing, and one in the crossing centre. Directionality is determined via a combination of directional strike in treadles, see Figure 4-6, and relay circuits which remember which Track Circuit was occupied first. Strike out treadles are typically non-directional, unless a specific signalling need dictates their use.

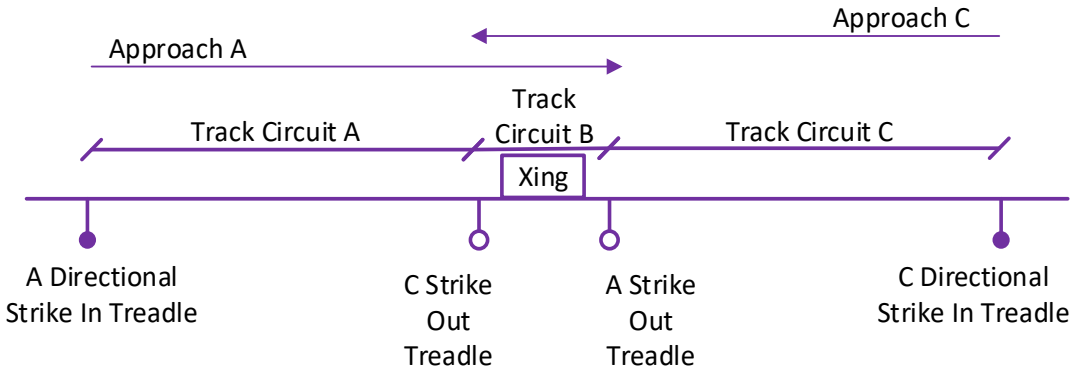

Figure 4-7 Bi-directional layout of Track Circuits at an Automatic Half Barrier Crossing.

The track circuits comprise an electrical current running through rail track, powering a relay. When a train enters a track circuit, its axles short the circuit causing the relay to de-energise. This signals the train's presence on the track segment to the crossing's control system by breaking an otherwise complete and powered electrical circuit. Treadles are fitted at the start of the track circuit which break the circuit when operated. This ensures prompt de-

energisation of the track circuit relay. This protects the integrity of the protection system timings against de-energisation delays due to poor conductivity between the rails and the train's wheels. This could be caused by rusty rails or leaves on the line (Westinghouse Signals, 1999).

The track circuit relay uses special weld resistant silver-graphite contacts. These significantly reduce the risk of track circuit relays welding, though excessive wear may stop the relay contacts from closing the electrical circuit (Kagra, 2013). In practise, this means it is common for a track circuit relay to falsely report the presence of the train, but very rarely do they fail to detect the presence of a train. The former scenario will result in the crossing protection systems becoming stuck on, and in the latter, they will fail to operate. Due to remote monitoring of all track circuits both scenarios can be immediately discovered.

Treadles are spring loaded mechanical switches, actuated by the passage of a train's wheels over a protruding lever (Henry Williams Limited, 2020). In AHBC strike in train circuits, these are wired to cut the circuit between the track circuit relay and the control system when the lever is depressed. This means, that both the track circuit and treadle must fail in a closed circuit manner for track circuit system to fail to report an approaching train to the AHBC's control system. As a mechanical switch, a treadle may fail open circuit where the treadle reports the passage of a train regardless of whether one is present. This could be caused by: the lever becoming stuck down; failure of the spring which keeps the lever raised; or excessive wear on the electrical contacts within the treadle. It could also fail closed circuit, where passage of a train fails to actuate the lever and break the track circuit. This could be caused by: the lever bending; the lever snapping; or metallic debris or distortion short circuiting the electrical switch within the treadle (British Rail, 1993).

Open circuit failure of either track circuit or treadle will result in crossing protection systems staying activated till the fault is resolved. Both relay and treadle must fail closed circuit for a train to fail to be detected. Network Rail's routine inspection protocol does not include individually testing each treadle and track circuit, resulting in only failures which affect signalling or crossing function being discovered. These are summarised in Table 4-1.

Table 4-1 Summary of Track Circuit Component Failure Modes.

Active Failures	Effects
Treadle Closed Circuit	No effect on crossing function. Undiscoverable in absence of other failures.
Treadle Open Circuit	Protection systems stuck on. Self-Reported by crossing.
Track Circuit Closed	No effect on crossing function. Discoverable via remote monitoring of track circuits.
Track Circuit Open	Protection systems stuck on. Discoverable via remote monitoring of track circuits.
Track Circuit Closed + Treadle Closed Circuit	Protection systems stuck off. Discoverable via remote monitoring of track circuits.
Track Circuit Closed + Treadle Open Circuit	Protection systems stuck on. Self-Reported by crossing. Discoverable via remote monitoring of track circuits.
Track Circuit Open + Treadle Closed Circuit	Protection systems stuck on. Self-Reported by crossing. Discoverable via remote monitoring of track circuits.
Track Circuit Open + Treadle Open Circuit	Protection systems stuck on. Self-Reported by crossing. Discoverable via remote monitoring of track circuits.

4.3 Road Traffic Lights

Figure 4-8 Wig-Wag Road Traffic Lights.

At an AHBC there are four sets of 'Wig Wag' road traffic lights, two on each side of the railway, one on each side of the road. The purpose of these traffic lights is to both warn drivers of an approaching train, and to the imminent descent of the barriers, if they have not yet descended. Additional lights may be installed if the crossing is located at a junction.

Modern Wig Wag lights use LED modules for illumination. Cross proving detects whether sufficient LED modules are illuminated when the control system calls for the light to illuminate. One Red LED module per side of the road can fail before it is reported to the monitoring signal box. Amber lights illuminate initially, followed by alternating red lights. Alternation of the red lights is achieved through a flasher unit, operating at approximately 0.8 Hz (Network Rail, 2008). The flasher may fail by flashing at an incorrect rate, or not at all, leaving only one red light per set illuminated. It is not clear from the documentation if the flasher may fail open circuit resulting in a failure of all lights to illuminate, though this cannot be discounted.

Study of the design of the traffic light revealed a robust design. The Lights are wired in parallel, ensuring failure of any LED module will not affect the remaining lights. Each LED

circuit is cross proven by a relay. Failure of two out of the four LED bulbs per crossing approach will leave the cross proving relay unpowered, allowing the control system to self-report the failure. The failure modes which affect LEDs are many and varied, however ultimately result in either dimming or complete failure of the LED to produce light (Chang, et al., 2012). The latter is likely to be self-reported, as complete failure is generally associated with open circuit failures. However, dimming presents a hazard in addition to producing insufficient light – it may not be detected by the cross proving system as it is still a closed circuit.

4.4 Hydraulic Barriers

An AHBC employs two barrier/boom units. One either side of the crossing to block the passage of traffic into the shared area of the crossing. There are several different designs of hydraulic barrier in use on the UKs network, each adhering to a core design specification (British Rail, 1991).

Figure 4-9 Newly Approved Level Crossing Barrier Design with Shroud Removed.

These barriers are designed to be raised using a hydraulic ram, and lower under gravity. A solenoid valve is used to control fluid flow from the ram back to the reservoir. When powered the solenoid mechanism holds the valve closed allowing the hydraulic pump to pressurise the system and extend the ram. When unpowered the solenoid valve opens permitting fluid flow from the ram, releasing pressure and contracting the ram. This design choice ensures that during any power failure the barriers will descend, closing the crossing to traffic. The rate of barrier descent is reduced using a counter weight.

Three electrical wiper contacts report the position of the barrier to the AHBCs control system. These simply report the barrier as raised, lowered, or mid cycle. A fourth raised

position wiper contact powers a relay which cuts power to the hydraulic pump motor, preventing over extension of the hydraulic ram. Should the hydraulic ram sag lowering the barrier from the raised position to the mid position, the pump motor will be powered again until the barrier is in the raised position once more. A strain gauge wire runs along the length of the boom. In the event of a significant impact which bends or shears the boom this wire will snap. This will break an electrical circuit using the wire, triggering a response from the control system which both activates all the crossing protection systems and informs signalling staff of a malfunction. Figure 4-10 shows a simple layout of these components comprising the barrier unit.

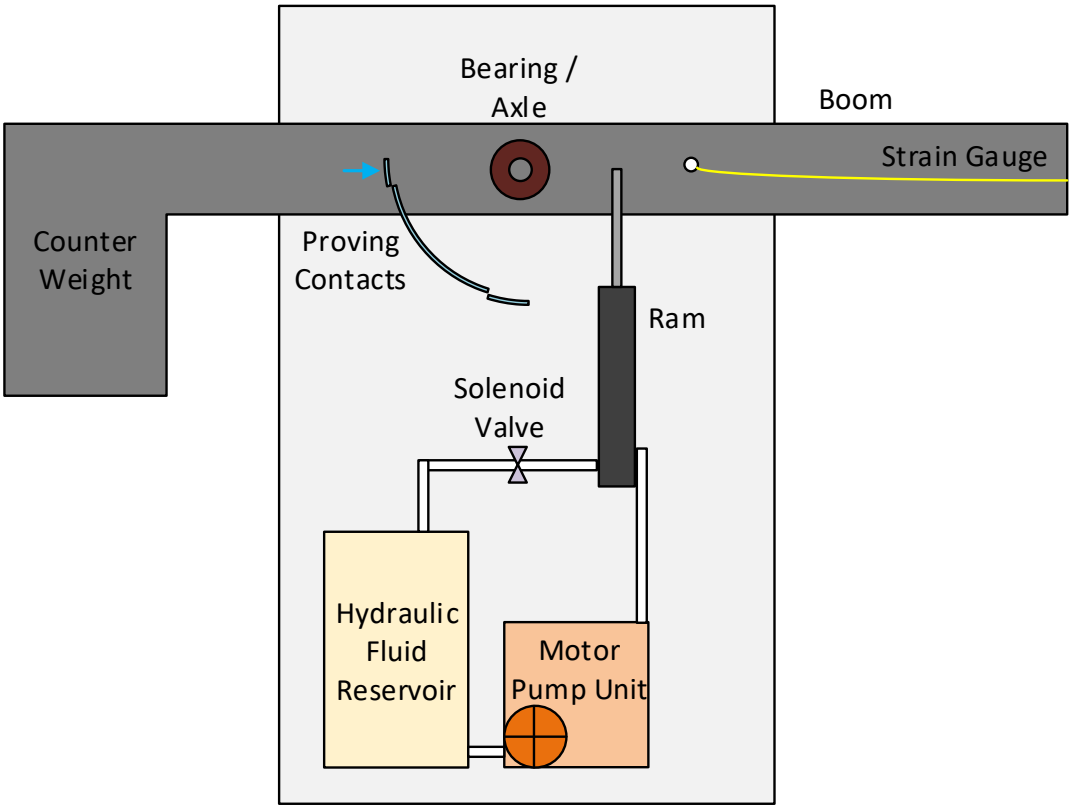

Figure 4-10 Hydraulic Barrier Equipment.

The barrier unit follows a simple duty cycle. After a train enters the AHBC's track circuit the control system breaks the motor control circuit. This cuts power to both the motor-pump unit and the solenoid valve. This allows the hydraulic fluid within the ram to return to the reservoir, forced by the weight of the raised boom. As the barrier descends it reports its position as mid, then lowered via the proving contacts. The barrier position can be monitored remotely by signalling staff. Should the barrier not lower in a timely manner, the control system will report a failure and hold protection systems active until the fault is resolved. After the train has passed the crossing the control system closes the motor control circuit. This powers the motor pump unit pressurising the ram. It also closes the solenoid valve

preventing fluid flow from the ram to the reservoir. When the barrier is fully raised the fourth proving contact completes and the power to the pump is cut.

Wear could cause the hydraulic fixtures to leak or fail. Depending on the location of the leak this may cause the boom to sag. In the short term the system can tolerate this, as the boom sags the raised proving contact will break and the motor-pump will be re-powered, raising the barrier. In the long term, eventually the hydraulic fluid reservoir will empty and the barrier will not raise. Protection circuits within the motor-pump unit will prevent the pump from running dry and burning out. Hydraulic leaks are discoverable during routine inspections. Failure of the barrier to rise when called for will be discovered via the proving contacts and self-reported by the control system. The resting state of the solenoid valve is in the open position. This ensures that during a loss of power to the barrier unit the boom will descend. Should the solenoid become stuck in the closed position the barrier would be unable to descend.

The proving contacts may commonly fail open circuit due to wear or insulating debris. The precise effects on the crossing system and whether it would be self-reported depend on the control system, discussed later. Open circuit failure of the motor control contact will prevent the barrier rising. Closed circuit failure will cause the boom to over extend, the pump motor will continue running until its own protection circuits cut it out. If these fail, the motor will burn out (Roberts, et al., 2010).

Failure of the bearing which supports rotation of the barrier will result in damage to the boom, potentially leading to it separating from the rest of the unit. This would break all proving contacts, allowing the failure to self-reported by the control system. Should the boom be struck by a vehicle and deformed its strain gauge wire will shear. This will be detected by the control system and self-reported also.

4.5 Power Supply

Power supply equipment is used to rectify mains alternating current (AC) electricity to direct current (DC), which powers all crossing systems. This comprises a rectifier and a bank of back up batteries. The rectifier converts incoming 240v mains AC to 24v DC. This is used to both power crossing systems and trickle charge a bank of back-up batteries. The batteries also serve to smooth any errant AC waveforms that have escaped rectification. This is important for reliable operation of the DC relay logic control system described later. The power supply features both a fuse, and a circuit breaker to protect from power surges.

Should either the rectifier fail or a mains AC power cut occur the back-up battery will power the crossing and monitoring signalling staff are alerted. As the batteries are wired in parallel to the rectifiers the batteries will power the crossing immediately after power failure, without need for mechanical or human intervention (Network Rail, 2004). The AHBC specification dictates that the battery back-up bank should have sufficient capacity to power the crossing for 8hrs (Network Rail, 2011). Should the back-up battery run flat monitoring signalling staff are notified when all status indicating lights turn off. The exact capacity required will vary depending on how frequently trains pass the crossing, as the most significant power draw will be from the protection systems cycling. The battery types used in AHBCs may vary, however the specification recommends ALCAD safety critical deep discharge types. These batteries are designed to be operated without maintenance, and to fail in a safe manner (ALCAD, 2020).

Individual batteries may fail in a manner that prevents them from holding a charge, lowering the overall battery back-up time. Though the batteries are designed to fail in a safe manner, it cannot be ruled out that they will fail hazardously, either by exploding or igniting. It is assumed in this scenario that the power will immediately fail and all batteries will be damaged beyond repair.

The fuse or breaker periodically requires resetting, having encountered either a power surge or otherwise tripped/blown spuriously.

4.6 Road Surface

The shared area of a level crossing must safely facilitate the passage of both road and rail traffic. Multiple different types of road surface have been installed at ABHCs. Older crossings may use tarmac or wooden sleepers. Modern installations installed after 1991 use a modular block system referred to as BOWMAC, shown Figure 4-11. The blocks are constructed from steel with a concrete surface to provide grip for road vehicles. Blocks are lain in between rails and modular concrete channels, with rubber wedges used to ensure a tight fit. They are primarily kept in place by virtue of this tight fit and their considerable weight. When the BOWMAC surface becomes worn they require replacement.

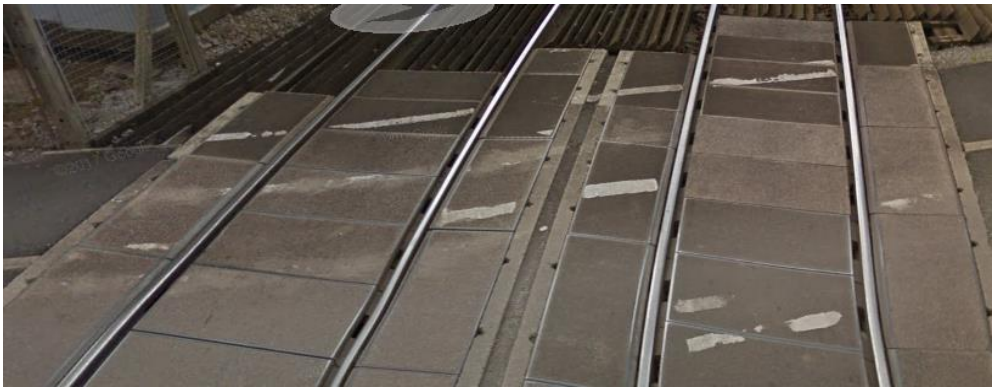

Figure 4-11 Modular Road Surface at a Level Crossing

4.7 Cabin

AHBCs use secure cabins to house sensitive control system equipment. These cabins appear reinforced and intended for long term use, however clearly resemble PortaCabins. The relay based control system housed within these cabins is highly sensitive to humidity and moisture. Some cabins have active climate control equipment. The cabins are not mentioned in the AHBC's specifications therefore their properties can only be speculated. As they are flat roofed structures they will likely need periodic refurbishment of their roof structure. Figure 4-12 shows a cabin at an AHBC that appears to have had its roof replaced, evidenced by the mismatched paintwork. Should the failure of the roof allow water ingress, it is possible that the working life of the sensitive equipment inside may the reduced. However there is no evidence of this effect occurring or its magnitude.

Figure 4-12 Road side cabin containing the control system - mismatched paint indicating a repair has taken place.

4.8 Control System

A AHBCs control system's primary purpose is to cycle the crossing protection systems in response to an oncoming train. As outlined in Figure 4-5, the protection systems operate following a defined sequence with the control system responsible for this. In addition to this the control system also performs various cross checking functions. Cross proving serves to detect equipment failures. Should the control system's cross proving circuits discover that a protection system has not cycled in a prompt manner it will lock the operational protection systems in an active state and notify signalling staff monitoring the crossing remotely (British Railways Board, 1992).

AHBCs use a relay logic based system. These use relays wired into circuits which emulate Boolean logic. The relays used are BR930 miniature signalling relays. The fundamental design philosophy of a relay is to use an electromagnet to break or complete a circuit. Normally relays are used in electronics to break or complete circuits which are of a considerably higher power output that the controlling circuit. In relay logic control systems they are used to emulate Boolean logic, as such the design of the signalling relays differs from normal relays. This type of relay typically has many different sets of contacts, allowing its integration into many different logic circuits. The BR930 signalling relays can contain both front and back sets of contacts. When powered, the electromotive force from an electromagnet within the relay pulls front contacts together completing a circuit. Back contacts behave in an opposite manner, when the electromagnet powers, these contacts are separated breaking the circuit (British Railways Board, 1981). This is shown in Figure 4-13.

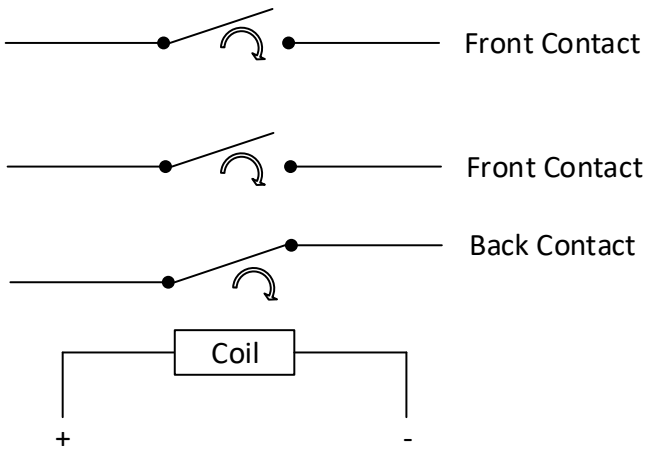

Figure 4-13 Signalling Relay, arrows indicate nature of movement when the electromagnet is powered.

Normal relays can exhibit two different failure modes. Relay contacts may become worn to such an extent that they are no longer able to permit an electrical connection between them.

If relay contacts carry excessive current they may weld together, leaving the circuit unable to be broken. Signalling relays are built with graphite impregnated silver contacts to resist any welding (Kagra, 2013). Due to this, signalling relay contacts may only fail open circuit due to excessive wear. This allows the effects of relay failure to be predicted and managed via an informed logic circuit design resulting in an AHBC control system which generally fails in a safe manner. For the purpose of later modelling, closed circuit relay welding events will be assumed possible, albeit extremely rare.

Timed functions are created by placing a capacitor in parallel with the electromagnetic coils of a signalling relay, shown Figure 4-14. When powered, the relay operates normally and the capacitor is immediately charged. After the circuit powering the relay is broken the capacitor keeps the electromagnet powered until it is discharged. The capacitance of the capacitor determines the length of time the electromagnet is powered for. This allows the integration of short timed functions into the relay logic control system which control protection system activation timings. Due to the capacitance required electrolytic capacitors are used. Evaporation of the electrolyte within these capacitors will result in a progressive loss of capacitance (Gallay, 2015), and hence a reduction in the length of time the electromagnet can be powered for. This will correspond to timed functions completing faster than intended.

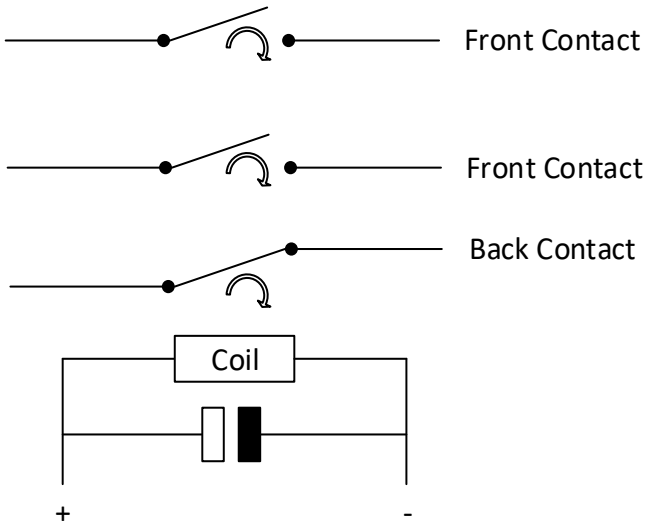

Figure 4-14 Timed Relay.

Electrolytic capacitors enable accurate timing, however only for short lengths of time. Long timed functions required for cross checking use a different design of relay, employing a heating element and heat sensitive switch. When the circuit powering this relay type is completed the heating element begins to heat the relay. After an approximate period of time spanning several minutes the heat sensitive switch will close, powering the relay's

electromagnet. This form of timing is imprecise, however it is sufficient for circuits which check the state of a crossing several minutes after a train has passed. Failure of the heating element within this relay will result in the heat sensitive switch never closing, and the electromagnet remaining unpowered.

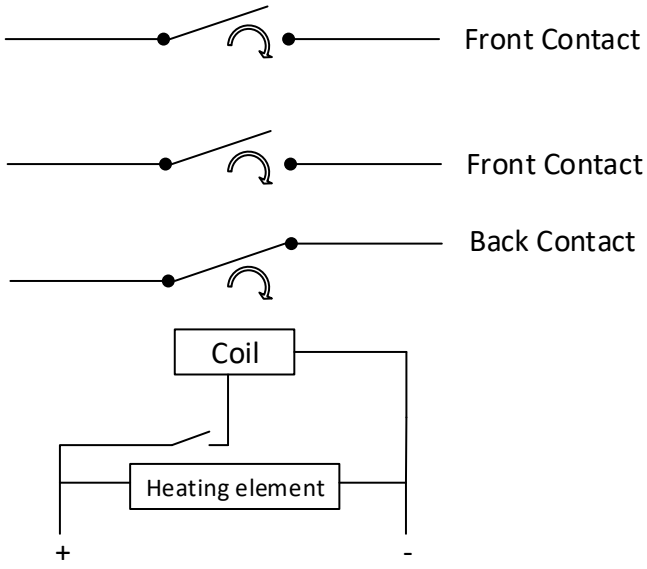

Figure 4-15 Long Timed Relay.

Figure 4-16 shows an example circuit used in AHBCs relay logic control system. This circuit determines when a train is occupying the C track circuit. When the circuit is complete, the B TPSR relay is powered. Two example relay contacts have been labelled: A QRR, circled in green, shows a front relay contact whose contacts close when A QRR is powered; A TPSR, circled in blue, is a back relay contact whose contacts break when A TPSR is powered. The states of these relays are determined by their own circuits (not shown). The complexity of the control system becomes apparent when it is considered that all of the relays shown on the B TPSR circuit diagram have their own circuit of similar complexity.

The B TPSR circuit is designed such that when AC TPR de-energises, indicating a train is in the C track circuit, B TPSR will also de-energise. The path to energising B TPSR labelled 1 is broken when a train approaches the crossing from the C track circuit. The path labelled 2 is used to re-energise B TPSR if a cross checking circuit determines the B TPSR was falsely de-energised. Should a train approach the crossing from the A direction, it will cross track circuit C after it passes the crossing. Path 3 re-energises B TPSR immediately after the train has exited the AHBC track circuits via track circuit C. Path 4 shows a stick circuit. This allows B TPSR to keep itself powered providing AC TPR is powered. This ensures B TPSR only de-energises when a train is occupying track circuit C.

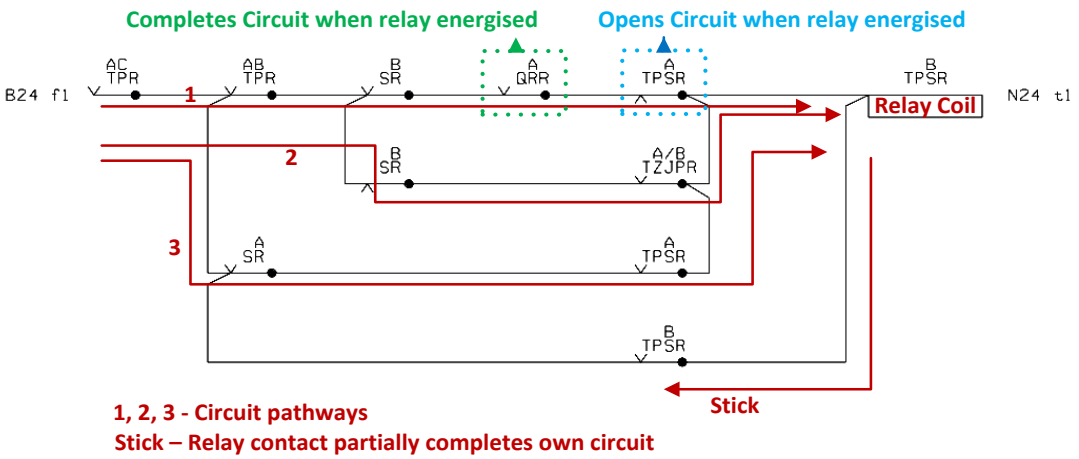

Figure 4-16 Example AHBC Control System Circuit (Westinghouse Signals, 1999).

Relay logic circuits can be expressed in terms of Boolean logic. The example circuit presented above is shown in its Boolean form in Figure 4-17. This form can be used to structure analysis, but traditional Boolean analysis methods cannot be applied. This is because of the temporal element of AHBC control. Due to the movement of trains and timed relays the state of many relays will change as the crossing cycles. Due to the complexity of these logic circuits it is not possible to determine how the crossing will function with concurrent failures by inspection alone. Work presented later in this thesis will determine the effects of specific relay contact failures on crossing function.

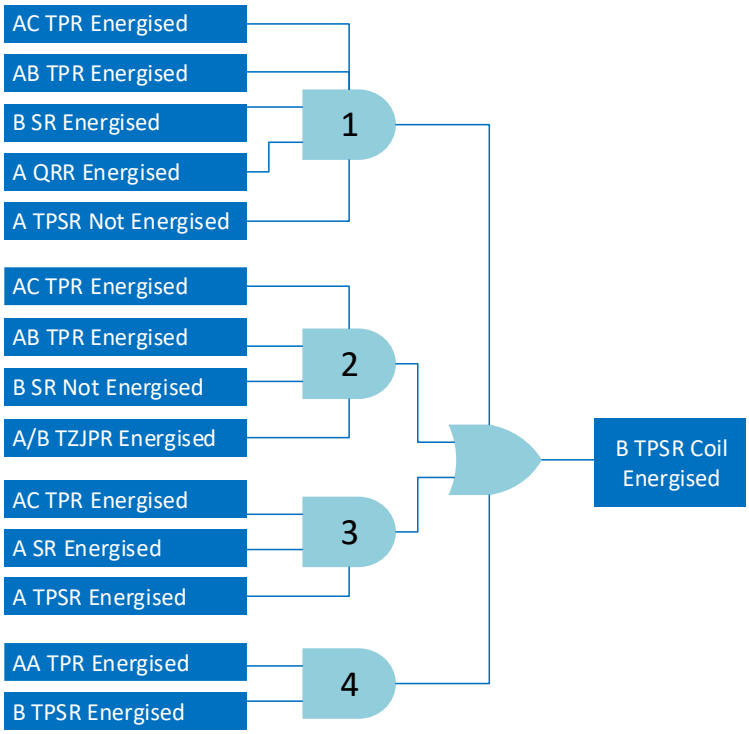

Figure 4-17 Boolean Logic Diagram for Example Circuit.

4.9 Asset Management of Level Crossings

Network Rail maintenance depots are responsible for the maintenance of level crossings. Each AHBC must be inspected no less frequently than once per seven weeks. The exact maintenance inspection frequency can be dictated by the territory manager. The engineering standard developed by Network Rail for level crossing maintenance includes a specification for inspections, however these are cursory. It comprises a visual inspection of the condition of the crossing, and confirmation that all protection systems are operating within acceptable timings only (Network Rail, 2004). Annually, each level crossing has its maintenance record and its condition reviewed by a group of Network Rail engineers who were not involved in its maintenance. This to ensure appropriate standards are being adhered to.

4.10 Human Crossing Interactions

The previous chapter reviewed level crossing risk models. It was found that human behaviour is a significant contributor to collision risk at AHBCs. Barrier weaving is commonly cited as a cause of collisions, where road drivers deliberately drive around lowered half barriers. Sometimes this places both road and rail vehicle on course to collide. What was not clear from the studies found, was the effect of concurrent level crossing protection system failures on this risk.

Factors which could be explored include: Barriers stuck raised/lowered; lights stuck on/off; late activation of barriers/light/both. If these factors were to significantly affect collision risk, it would create a link between crossing condition and collision risk. This would result in maintenance actions having the capacity to change collision risk.

4.11 Summary

This review has shown the AHBC system to contain a diverse range of systems and components. These systems have generally seen no prior modelling in literature. The complexity of the AHBCs control system creates an obstacle. It is not possible to determine the effect of component failures within or external to the control system on the crossing function by inspection. Nor is any documentation or study available which explores or lists failure cutsets for the protection systems. A model of the system is presented later in this thesis which determines the effects of single and concurrent component failures on the integrity of the crossing's protection functions and self-reporting system.

Review of risk factors for AHBCs found miss-use to be the greatest cause of collisions. No models or statistics consider the effect of specific crossing protection system malfunctions on collision risk at ABHCs. These deficiencies will be addressed in the following chapter on

modelling through further development of the available models. Should this model show that protection system failures increase collision risk, it will demonstrate a potential link between asset management of an AHBC and collision risk.

5. Collision Risk Model

The purpose of this model is to estimate the probability of a collision between a road vehicle and a train at a level crossing for given usage parameters and operational states, such as hazardous breakdowns. These collision probabilities are calculated per each train passing the crossing. This allows estimation of the number of collisions at a level crossing as a product of the number of trains expected to pass the crossing whilst it is experiencing a hazardous breakdown. The number of hazardous breakdowns expected is predicted by the asset management model described in a later chapter.

The model has been created using a Coloured Petri net, and analysed using a Monte Carlo method. In many studies reviewed in chapter 3, regression models were used to estimate collision frequencies at level crossings. These are insufficient for the purposes of this work because collisions are rare events occurring in the order of once per decade or century at a given level crossing (Evans, 2011). As such, insufficient data is available to create accurate regression models which include multiple factors such as vehicle arrival rate at the crossing, crossing geometry or crossing operational state. By using models which lean heavily on the well-studied fundamentals of road vehicles and driver characteristics, models can be developed which make predictions based on these factors. It enables risk modelling of level crossing accident risk to move from the network or route level, to individual crossings. Though direct verification of the model itself is difficult for the same reason that regression models are inappropriate.

A further hindrance to regression modelling of historical collisions is the changing nature of driving as new technologies are adopted, and perhaps even changing attitudes towards driving. Studies conducted several decades ago are much less valid today in light of continual improvements in vehicle stopping distances and general handling. The same studies will be even less valid in the coming decades as automatic braking systems become more prevalent and sophisticated. Models based on the fundamental processes of driving such as the one presented here will be able to adapt to these changes much faster, and even predict their effects.

5.1 Literature Review

Many level crossing collision models and analyses were identified in Chapter 3, however only a limited selection concerned Petri nets and Monte Carlo analyses. Professor Stott conducted the first study into accidents at UK automatic open crossings (1987). These unsupervised crossings, which use only road traffic lights, had experienced accident frequencies in the

order of once per 20 years since their introduction. Stott found this excessive accident frequency to be due to non-compliance with the crossing's road traffic lights. When the frequency of road vehicles using the crossing was considered a non-linear relationship was found. This relationship was the product of the number of road and rail vehicles using the crossing and hence likelihood of both a train and road vehicle being on a collision course, and the opportunity for drivers to pass the crossing at danger, unobstructed by drivers who had heeded the lights. Stott demonstrated this with a simple Monte Carlo model.

The British Standard for Petri net modelling (BSI Group, 2012) includes the development of an example model concerning collisions at level crossings. Despite being used as an example, and not the focus of the published article, the model is well developed and applicable to modelling automatic crossings. The model uses the Stochastic Petri net methodology with high level extensions such as inhibitor arcs. Car headway is modelled using an exponential transition, regarded as valid for modelling traffic flow where there is a large headway between vehicles (Knoop, 2017). Whilst not valid for all crossings, this is generally valid for automatic crossings which are typically deployed at low trafficked sites. Vehicles are randomly assigned to one of three levels of compliance: full compliance, non-compliance, and non-compliance despite a visibly passing train. Vehicles spend an exponentially distributed length of time in the shared area of the crossing before leaving. Trains are modelled arriving and leaving using exponential transitions. The model has some facility to model both hazardous and safe crossing breakdown scenarios. Collisions are recorded when both road vehicles and trains occupy the shared space of the crossing at the same time.

The last example of a Petri net to model collisions at level crossings was published by Ghazel (2009). This model uses a Stochastic Petri net to model the effects of 'blocking back' at level crossings, where drivers enter the shared area of the crossing without any clear exit route due to traffic queues originating after the crossing. In this model vehicle arrival rate is again modelled exponentially. The time for the vehicle to clear an area is modelled using a truncated normal distribution. Traffic jams randomly occur blocking the exit to the crossings, drivers may then either enter the shared space of the crossing or safely wait at the entrance. Train arrival is modelled similar to the BSI model. Drivers are modelled as always complying with crossing protection systems, however may become trapped in the shared area should a train approach after they entered the shared area.

Neither of the Petri net models reviewed are sufficient for the purposes of this work. Whilst the BSI model covers many aspects of crossing interactions, it does so in an overly simple

manner in places. The BSI model considers crossing protections systems as operating, or failing in unison, whereas the analysis in the previous chapter shown that the barriers of an automatic half barrier crossing (AHBC) may fail independently of the lights and vice versa. Given that crossings with only road traffic lights have much higher collision frequencies than half barrier crossings (Evans, 2011) it seems reasonable to assume that compliances with barriers will be different and superior to that of compliance with the road traffic lights. As a consequence, a collision model may benefit from considering more than one level of driver compliance at an AHBC. The model also suffers from a high computational burden, as trains are modelled arriving randomly there may be long periods where road vehicles are simulated during which no collision is possible regardless of driver behavior.

The model published by Ghazel focuses on 'blocking back' behavior, however this is more applicable to crossings in urban areas where traffic lights and junctions are common. In contrast, automatic crossings tend to be situated in rural areas with little traffic and thus there appears to be little value in including such a feature in the model developed in this work. As no suitable models are available, a model will be developed and presented in this chapter for the purpose of estimating collision risk at AHBCs.

5.2 Model Description

The model functions by simulating road driver's behavior at a level crossing as a train approaches and the crossing protection systems operate. Should a vehicle and the train be determined to occupy the same area of the crossing simultaneously a collision is recorded. Train behavior is modelled simply: the model begins with the train 20 seconds from entering the crossing track circuit. When the train reaches the track circuit the level crossing lights operate, 7 seconds later the barriers begin to descend, and 5 seconds after they have completed their descent. After a further 26.5seconds the train reaches the shared area of the level crossing where a collision may occur. The train takes 1.5seconds to clear the crossing. Trains do not react to simulation events. The simulation resets after the train has passed the crossing, this ensures the model does not waste computational time simulating road vehicles when a collision is impossible.

Road vehicles arrive randomly with a constant arrival probability with time, modelled using an exponentially distributed random variable to generate headway between vehicles. Any headways generated below 1second are set to 1second to prevent unreasonably close headways. This assumes road vehicle headway is sufficiently large such that the headway is random, and not influenced by the capacity of the road (Knoop, 2017). Vehicles spawn on

either side of the crossing. When a road vehicle enters the simulation, its vehicle type, speed and driver compliance level are randomly determined. A further assumption regarding headway is required, it is assumed that the headway between drivers is large enough that the speed of each vehicle is independent of the vehicles preceding them.

Vehicle type is determined randomly as either a car or a heavy goods vehicle (HGV). HGVs and cars differ within the model by length and speed, but retain the same stopping distances and levels of compliance with the crossing protection systems. Cars are modelled as being 4m long, and HGVs 15m long. A traffic survey at a local AHBC found that 7% of the traffic over this crossing were heavy goods vehicles (Network Rail, 2016), which has been used to configure the model.

Vehicle speed upon entering the simulation is modelled based on government survey data (Department For Transport, 2018). Annually, the UK government publishes the aggregate data from surveys of driver speed at multiple sites. The data used in this model is from roads where the speed limit is 60mph as this best represents the rural roads automatic crossings are deployed on. A normal distribution was fitted to this data, yielding a mean = 50MpH and standard deviation = 8.57MpH for cars specifically. This distribution is shown in Figure 5-1. This was repeated for heavy good vehicles, finding mean = 47.6MpH, standard deviation = 8.76MpH.

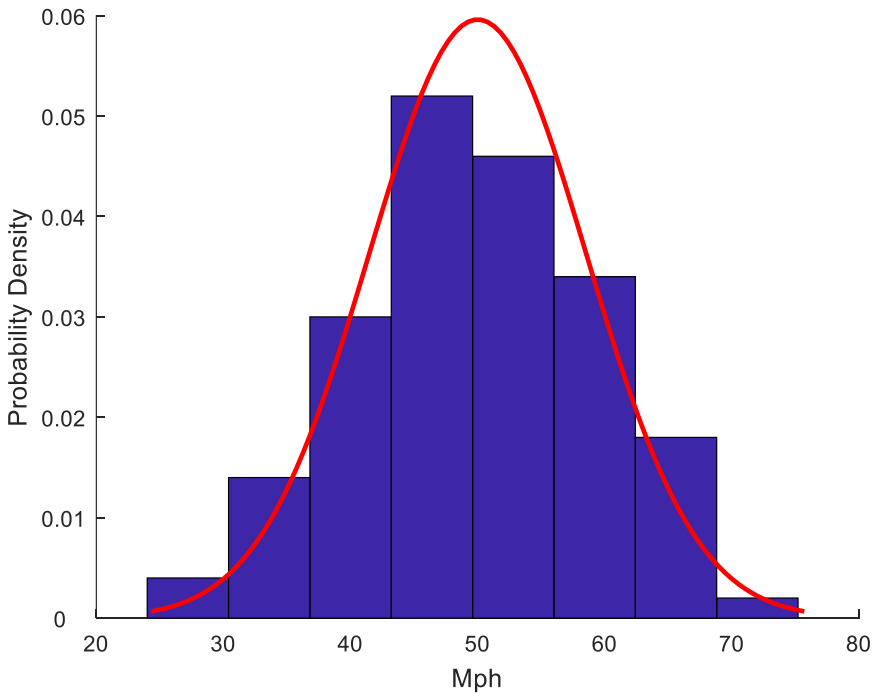

Figure 5-1 Vehicle Speed Distribution for Cars on 60MpH Speed Limit Roads (Department For Transport, 2018).

Road vehicle stopping distances have been taken from the UK's Highway Code. This comprises a base 0.7second reaction time, and a braking distance which varies with initial speed in discrete increments. These discrete increments have been linearly interpolated to obtain a continuous stopping distance estimate as a function of speed. The equations of motion used to model stopping assume the vehicles exhibit constant deceleration with time. The model does not at any point permit a vehicle to accelerate. This extremely simple modelling of vehicle braking is somewhat justified later by the model's insensitivity to braking distances and reaction times.

As level crossing events unfold a driver may react depending on their compliance level, this is illustrated in Figure 5-2. Vehicles always spawn 200meters from the crossing. Drivers travel towards the crossing at the speed previously determined. As they travel towards the crossing they may react to either the crossing's lights, barriers or a visibly passing train. Entirely compliant drivers will brake upon seeing any of these. Partially non-compliant drivers will disregard the lights but brake for the barriers. Entirely non-compliant drivers will only brake for a passing train. Should a second vehicle be present at the crossing as either a witness or an obstruction, both types of non-compliant driver will obey both lights and barriers. This has been modelled to match the behavior described by Stott (1987). All drivers have a 33% chance to observe a train before it reaches the crossing, if the crossing geometry has line of sight. This is in keeping with the results of a German study on road driver attention at level crossings (GrippenKoven & Dietsch, 2016). All drivers will brake upon observation of an approaching train regardless of compliance level.

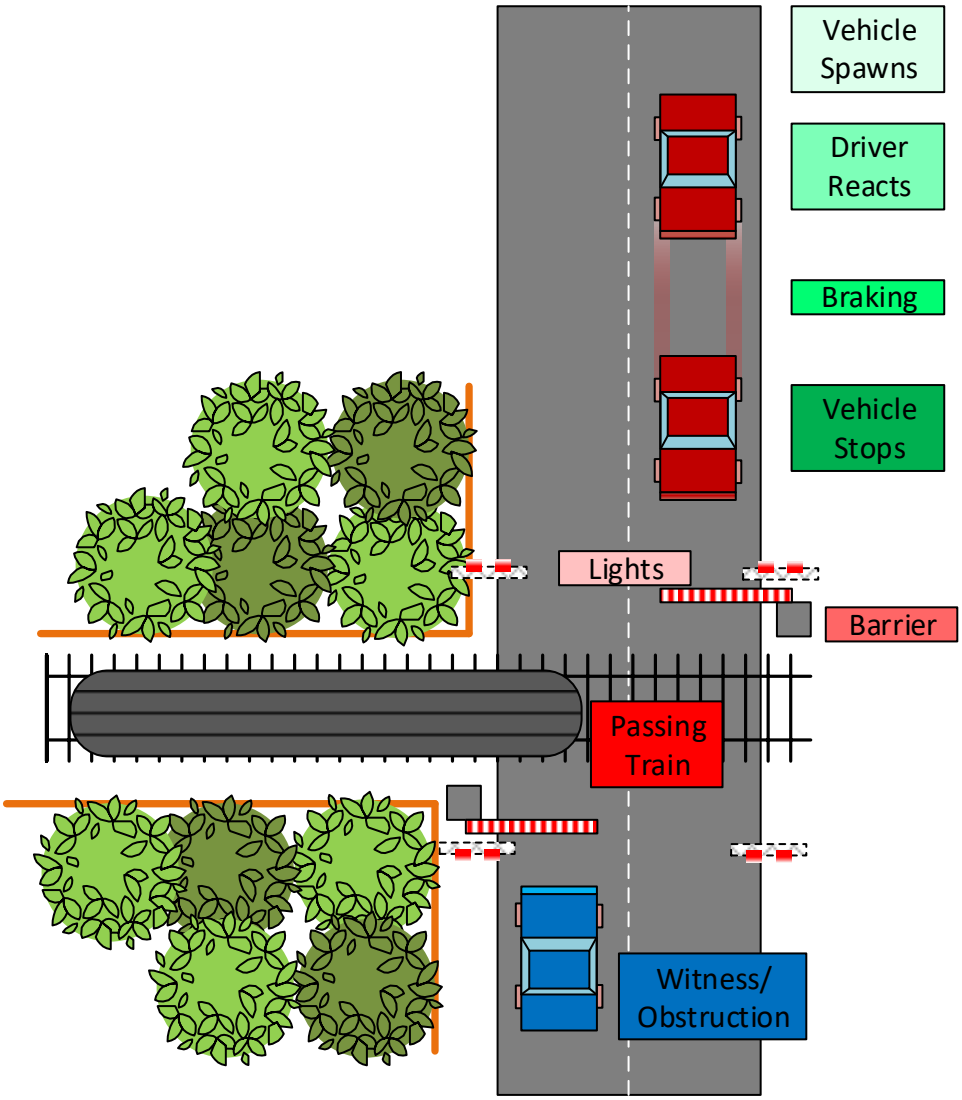

Figure 5-2 Driver Behaviour

No information is available on these proportions so they have been estimated. Stott predicted that 1% of drivers will disobey the lights of an Automatic Open Crossing which uses only lights. Thus, in this model 1% of drivers will be willing to ignore the lights, but not the barriers (partially non-compliant). The proportion of the drivers which will are entirely non-compliant has been estimated. These proportions are listed in Table 5-1.

Table 5-1 Driver Compliance Proportions

Compliance Level	Percentage
Compliant	98.977%
Partial Non Compliance	1%
Non Compliance	0.023%

Should a driver not obey the crossing's protection system warnings they will have to weave around the barriers, if required, as depicted in Figure 5-3. However, the model does not extend to 2-dimensional movement and instead assumes vehicles move in a straight line through the barrier. This may result in a modelled vehicle spending slightly less time in the shared area of the crossing compared with reality. However, extension of the model to 2-dimensions would introduce further difficulties and inaccuracies. To compensate for this, it is assumed that vehicle speed reduces by 1.5 as they traverse the crossing to account for a longer path.

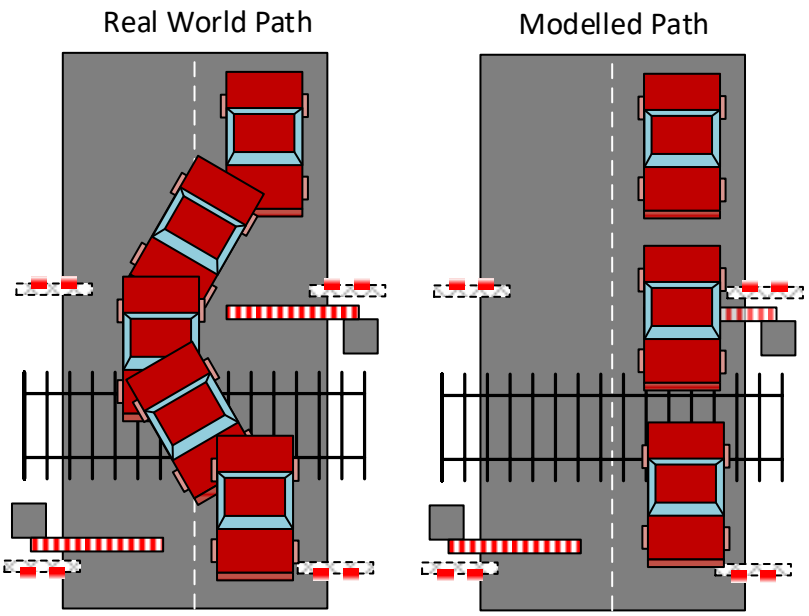

Figure 5-3 Barrier Weaving

Collisions may occur by the train hitting a road vehicle as it traverses the crossing, or a road vehicle may hit the side of the train as it passes, depicted in Figure 5-4. This could be due to non-compliant behaviour, or a failure of the protection systems. If the model is configured so that trains are not visible to road drivers as they approach the level crossing, a train may strike the side of a road vehicle without any reaction from the driver prior to the collision. However, in the scenario where the driver hits the side of the train, the train must be visible to the driver prior to the collision. Here, the driver may begin to brake if reaction time permits and may avoid hitting the passing train, depending on their distance from the crossing and current speed.

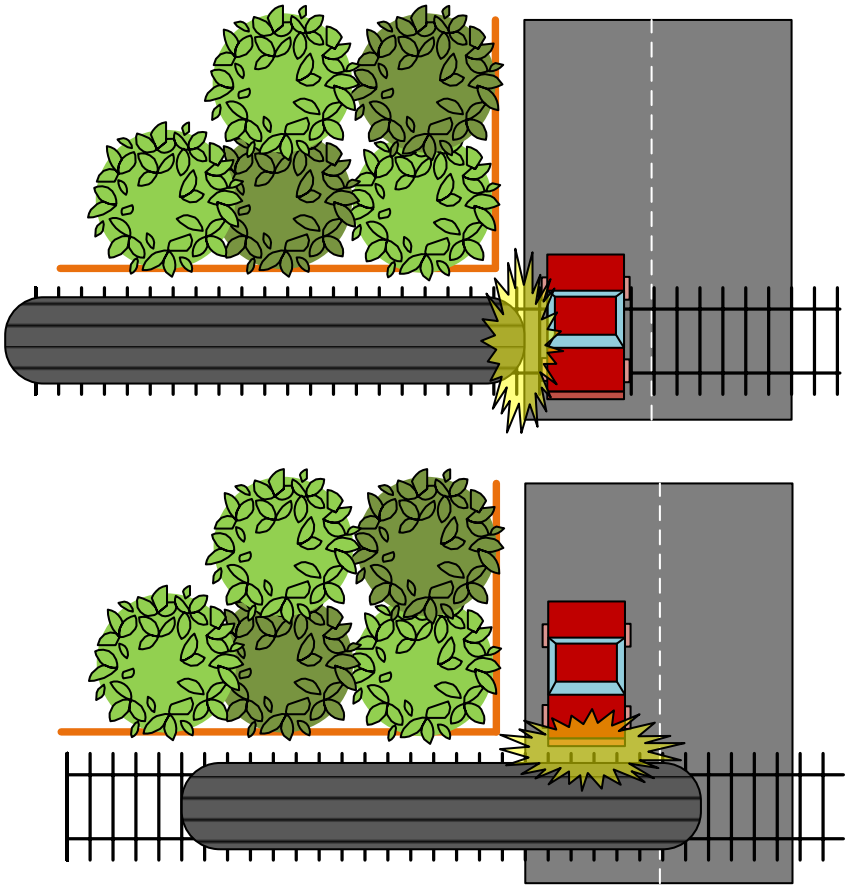

Figure 5-4 Possible Vehicle Positions during a Collision.

The model is simulated repeatedly and any collisions recorded as part of the Monte Carlo solution method. The model produces a probability of a train colliding with a road vehicle for the parameters input. Any combination of failures of the protection systems can be tested. This is a particularly valuable aspect of the model, as determining the probability of a collision for reduced levels of crossing operation would be impossible via a data driven method owing to the rarity of such events.

5.3 Petri Net Structure

The structure of the model comprises four parts: the train module, the road traffic light module, the barrier module and the road traffic module.

5.3.1 Train Module

The train simulation module is simple, shown Figure 5-5. The simulation begins with the train approaching the start of the level crossing's track circuits. Transition T1 controls the time till the train reaches the strike in treadles. The simulation starts here to give the traffic module time to generate vehicles approaching the crossing. Transitions T2 and 3 control the length of time till the train reaches a point where it is visible to road traffic passing the crossing for

each side of the crossing independently. Transition T4 then controls the time the train is visible to vehicles on both sides of the crossing before the train reaches the shared area of the crossing. The values of transitions T2, T3 & T4 can be altered to model different train speeds, track circuit lengths and visibility. Unlike previously discussed models, there is no randomness in the train characteristics. This is because the model estimates risk on a per train basis, where the total risk estimation is obtained by summing the risk from each train which passes the crossing. When the train exits the crossing, transition T6 fires resetting the entire Petri net to its starting state to simulate the next train.

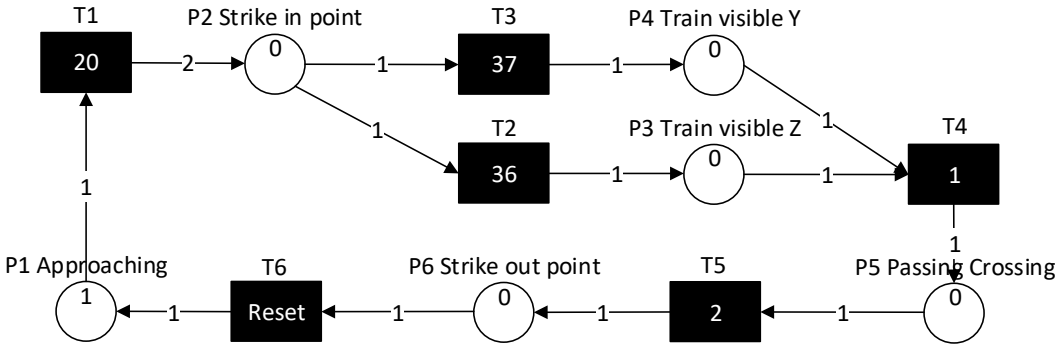

Figure 5-5 Train Simulation Module

5.3.2 Road Traffic Light Module

The road traffic light module operates on a simple lights on – lights off basis, shown Figure 5-6. At the start of each simulation, and whenever the simulation is reset by transition T6, there is one token in P7, indicating that the lights are off. Once the train reaches the strike in point it enables T7 via test arc. T7 fires removing the token from P7 and adding a token to P8 indicating that the lights are now on. To model a scenario where the lights activate late, a single token in P10 inhibits T7 and enables T8 via test arc which fires after a delay. A token is placed in P9 to model a scenario where the lights do not function. This inhibits both T7 & T8.

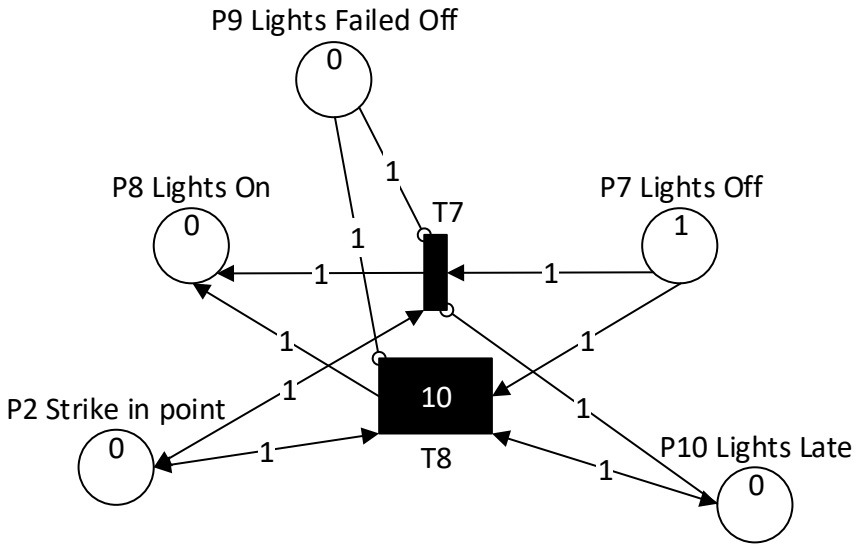

Figure 5-6 Road Traffic Light Module

5.3.3 Barrier Module

The barrier module follows a similar structure to the lights, however includes an additional state indicating the barriers are descending, shown Figure 5-7. The module starts when the train strikes in, achieved via test arcs from each transition to place P2. There is an initial delay, modelled via T9, where the lights operate for a period before the barriers begin to descend. When transition T9 fires a token is removed from place P11 and a token added to place P12. This signifies that the barriers are now descending. Failures which prevent the barriers from descending can be modelled by adding a token to place P14, this will inhibit transition T9.

Transition T10 models the normal descent of the barriers using a 5 second delay. This transition removes a token from place P12 and adds a token to place P13, indicating that the barriers have descended. Slow descent of the barriers may be modelled by the addition of a single token to place P15. This has the effect of inhibiting transition T10 and enabling the slower transition T11, which otherwise behaves the same.

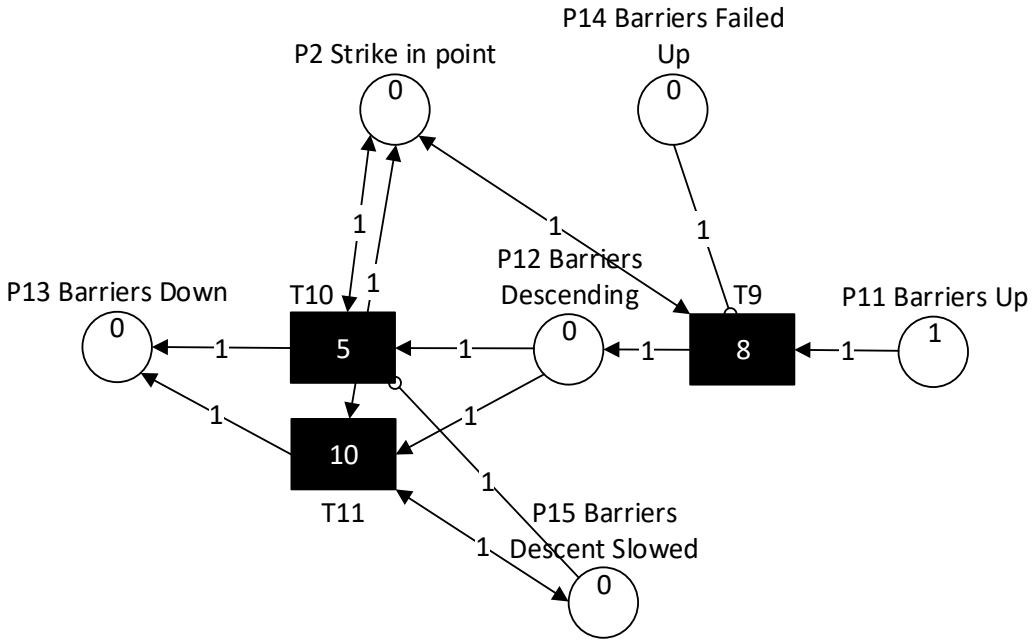

Figure 5-7 Barrier Module

5.3.4 Road Vehicle Module

The first element of the road vehicle module controls the arrival of road vehicles, Figure 5-8. Transitions T12 and T13 model the arrival of vehicles at each side of the crossing. The arrival rate is assumed to exponentially distributed as automatic crossings are generally deployed at low traffic sites and therefore unaffected by vehicle to vehicle interactions and headways. To prevent these transitions generating impossibly small vehicle headways transitions T14 and T15 have delays of 1 second. This prevents more than one vehicle per second per side of the road entering the simulation.

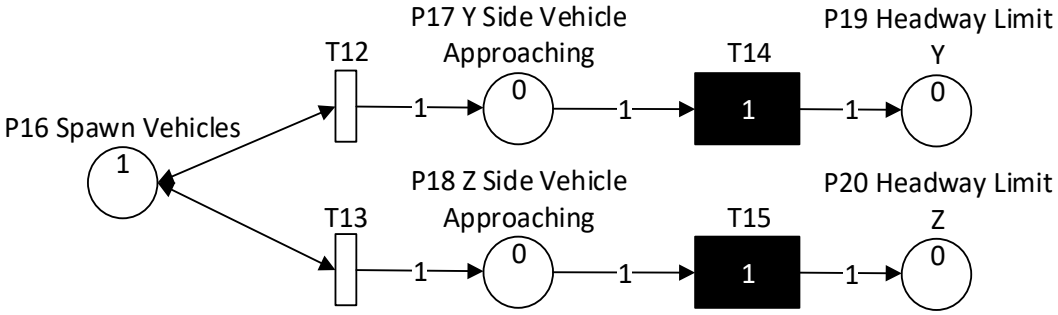

Figure 5-8 Road Vehicle Module Part 1

Two alternative designs for the rest of the road vehicle module are presented below. The first continues as a SPN, the second uses a CPN. As the road vehicle module is somewhat complex this provides a meaningful comparison between the two methodologies.

5.3.4.1 SPN Road Vehicle Module

The first task of the SPN road vehicle module is to determine what type of actor the driver is. This is achieved in the Petri net shown in Figure 5-9. When a token is present in place P19, six transitions are enabled, T16a-21a. One of these transitions is chosen to fire based on a weighting held by each transition, in a similar manner to how compliance is chosen in the BSI model (2012). When one of these transitions fires a token is added to the corresponding place from P20a-25a and a second token added to place P26a indicating that a vehicle is approaching the entrance to the crossing. Inhibitor arcs from place P26a to each transition ensure that should the vehicle chose to wait at the crossing no other vehicles will overtake.

Each of the places from P20a to P25a corresponds to a specific level of driver compliance. Place P20a is for actors who are entirely compliant and are aware of approaching trains if visible. Place P21a is the same as P20a, but corresponds to drivers who are not aware of approaching trains. Places P22a and P23a are for partially non-compliant drivers who may disobey the lights but not the barriers, again with P22a representing drivers with situational awareness and P23a those without. Finally places P24a and P25a represent entirely non-compliant drivers who will not stop at the crossing unless they see a passing train, or notice an approaching train (P24a only).

This Petri net, and the others to follow are duplicated for each side of the crossing, however for brevity only one side is shown.

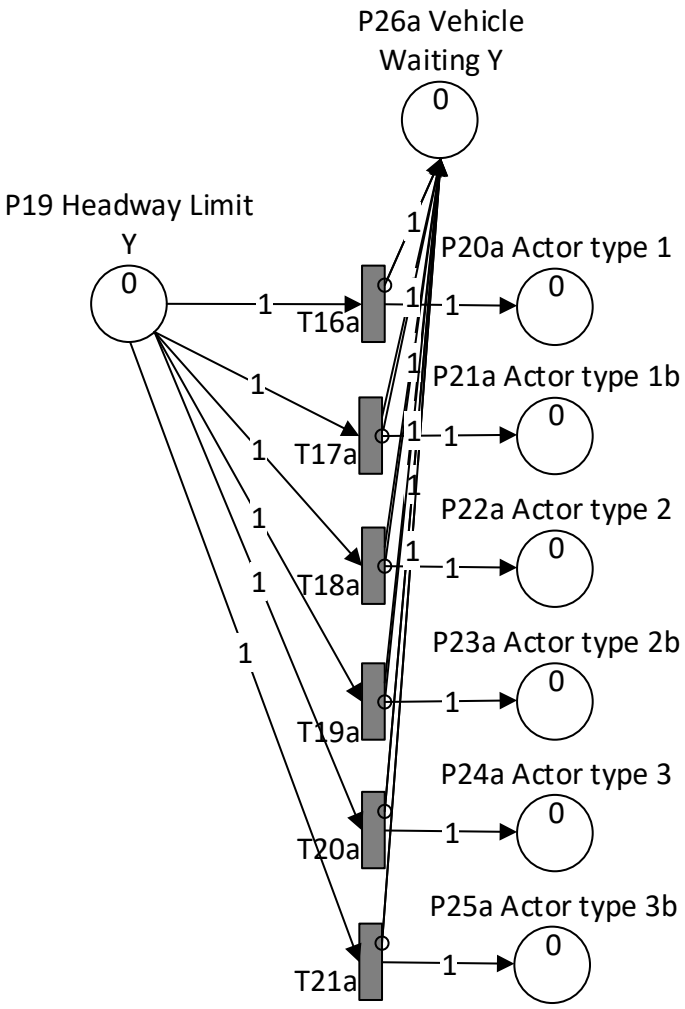

Figure 5-9 SPN Road Driver Type Selection

The next section of the Petri net contains the logical structures which form the behaviours described above, shown Figure 5-10. Transitions T22a to T27a control the behaviours for the corresponding places P20a-25a. This is achieved with inhibitor arcs from five other places: P30a representing if any vehicles are queuing on the other side of the crossing; place P8 describing if the lights are on; place P13 describing if the barriers are lowered; P4 describing if an approaching train is visible from the Y side of the crossing; and P5 describing if a train is currently passing the crossing.

Transition T22a specifically controls the decision making process of the entirely compliant drivers with situational awareness, as such this transition is inhibited by all the controlling places described. As transition T23a controls drivers who are entirely compliant but not situationally aware, this transition is inhibited by every place described except P4. The same logic applies to the remaining transitions, with transition T27a inhibited only by places P30a and P5. When any of these transitions fires the token is removed from the input place and a

token added to P27a indicating that the road vehicle has reached a distance from the crossing where they no longer have time to react should a train appear. The token is also removed from P26a, allowing another driver to approach the crossing.

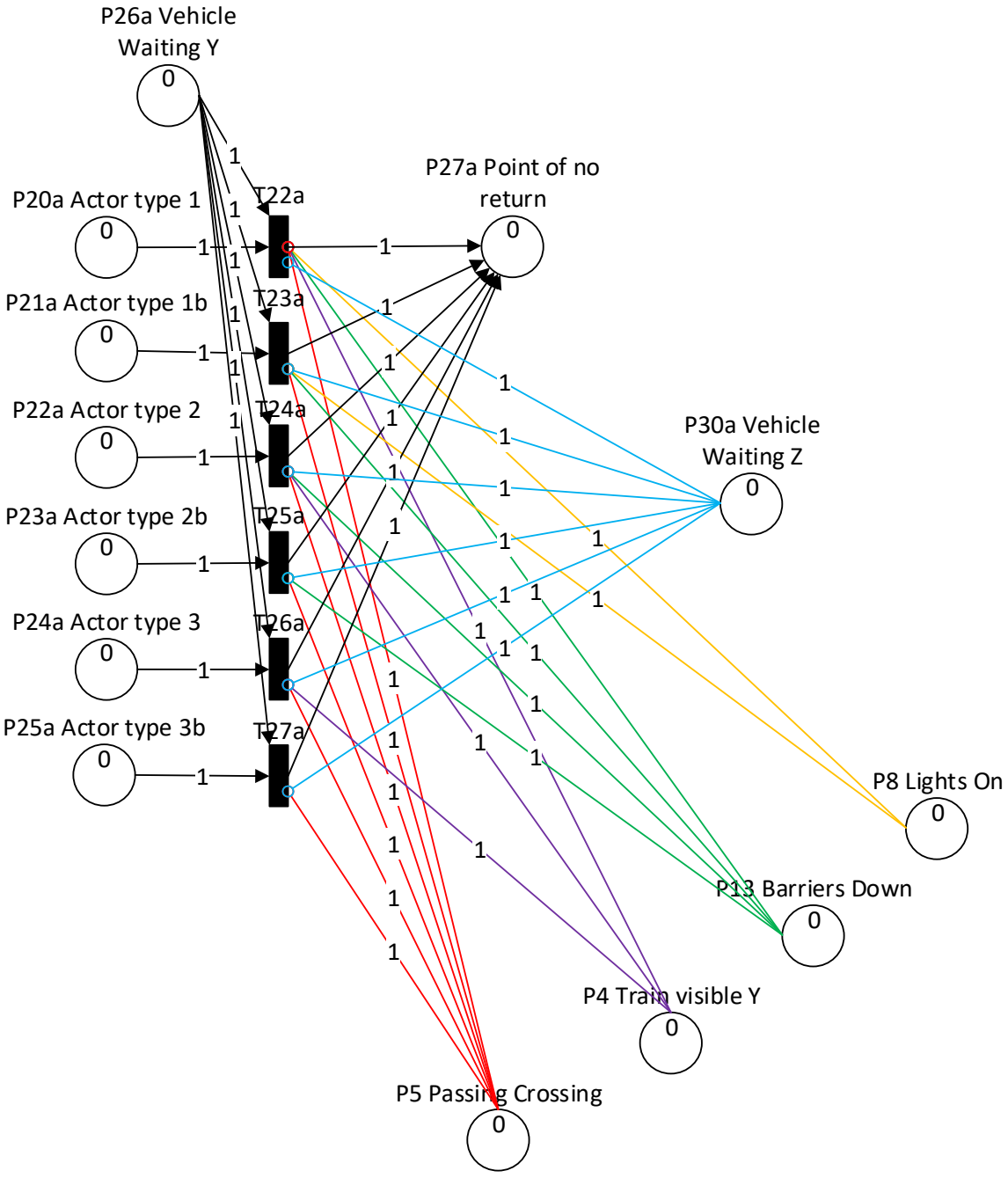

Figure 5-10 SPN Road Driver Type Behaviour Control

Finally, the driver travels the remaining 2 seconds to enter the crossing, and 0.5 seconds to clear the crossing itself. This is shown in Figure 5-11. The 2 second travel time is achieved with two 1 second delay transitions, T28a and T29a, this allows a minimum headway of 1 second to be maintained. Had a single 2 second delay transition been used the minimum headway would have been forced to 2 seconds.

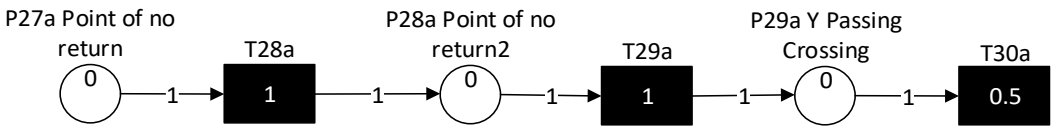

Figure 5-11 SPN Road Drivers Passing Crossing

An obvious limitation of this SPN version is its inability to consider a vehicle's specific speed, braking distance and driver's reaction time. These aspects are approximated in the three delay transitions T28a-30a. These could be replaced with stochastic transitions to allow some variation, with a transition for reaction time and a second for braking time, illustrated Figure 5-12. Here, transition T28b controls normal passage towards the crossing and T29b over the crossing. Should a driver decide to brake due to seeing a passing train Transition T30b will be enabled, and fire with random delay representing reaction time. Transition T31b can then fire with another random delay modelling time for the vehicle to stop. All the random variables are conditional upon the vehicles speed, obtained from the number of tokens in place P31b. Should the vehicle stop in time it will be safe, otherwise it will pass the crossing and risk collision.

At first glance this may seem to be sufficient, but there are many issues with this approach. Consider the scenario where a driver begins to brake, but does not stop in time to avoid entering the crossing. Due to the braking the driver will take much longer to clear the crossing, transition T29b cannot account for this unless place P31b, whose token count represents speed, is altered. Issues such as this can be solved by assuming constant deceleration and removing tokens from place P31b at a constant rate whilst there is a token in P28b.

An unresolvable limitation with this method however is that in an SPN only one token may interact with a transition at any given time. As such the actions of only one vehicle may be modelled at a time per side of the road. At crossings with low levels of traffic this may be accurate for most of the time, but it cannot account for randomly occurring busy periods when multiple vehicles may approach the crossing simultaneously. In a CPN these transitions could fire concurrently allowing multiple vehicles to approach at once, such as that described next.

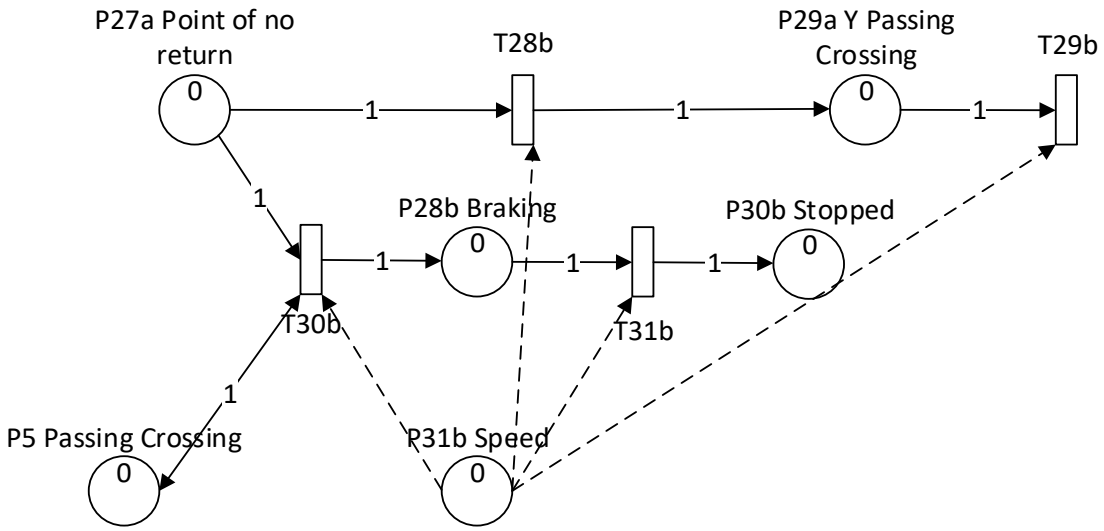

Figure 5-12 SPN Illustration of Complex Road Driver Passage

5.3.4.2 CPN Road Vehicle Module

The following CPN road vehicle module resolves the issues encountered with the SPN equivalent. The first portion of the CPN is shown in Figure 5-13. Transitions T16 and T17 set driver behaviour and vehicle characteristics.

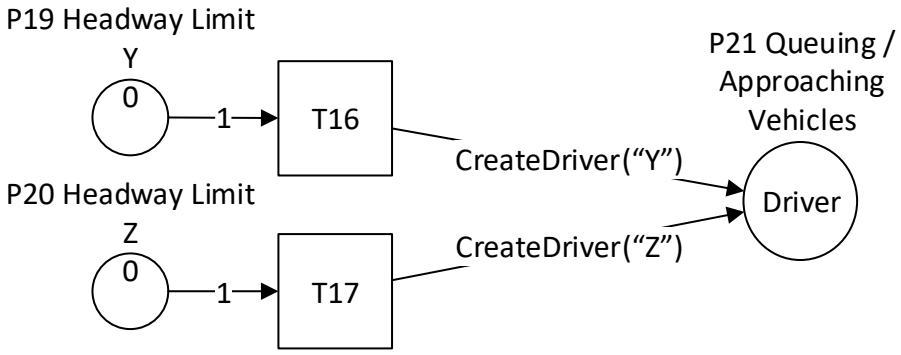

Figure 5-13 CPN Road Vehicle Module Part 1

When either transition fires a token is placed in P21 of the colourset 'Driver'. This contains information on: compliance, speed, vehicle type, spawn time and the side of the road the vehicle approached from. The definition of the 'Driver' Colourset is shown in Figure 5-14.

```
23. Colour Driver As ComplianceLevel x Speed x VehicleType x RoadSide x Deceleration x Position x
    SpawnTime
24. Colour ComplianceLevel As Integer
25. Colour Speed As Float (m/s)
26. Colour VehicleType As Binary [Car or HGV]
27. Colour Roadside As Binary [Y or Z]
28. Colour Deceleration As Float (m/s/s)
29. Colour Position As Float (m)
30. Colour SpawnTime As Float (s)
```

Figure 5-14 Colourset Driver

These values of the Colours in Colourset ‘Driver’ are determined within the output arc function ‘CreateDriver’. This function has been expanded in Figure 5-15. The first task of CreateDriver is to decide the type of vehicle, whether it is a car or a HGV. To achieve this, a uniformly distributed random number between zero and one is generated, shown line 34. Here, 7% of vehicles using the crossing are HGVs and the rest are cars. Thus, if the random number drawn is less than 0.07 the vehicle type is set as an HGV, otherwise it is set as a car, shown line 36.

Road speed for HGVs is determined over lines 38-42. On line 39 a normally distributed random variable is generated to be used as the HGVs speed. Lines 40 and 41 truncate the speed between 5 and 100Mph. This is repeated for cars over lines 44-48, using slightly different parameters to generate the normally distributed variable. Finally, line 51 converts the speed from mph to meters per second.

Next the function determines the compliance level of the driver. A third random variable is generated on line 53, uniformly distributed between [0,1]. The value of this random number is used to determine the compliance level over lines 55-57. Lastly, the function sets the time the vehicle spawned into the simulation to the current time, and sets the position of the vehicle at 0 m, and creates the token to be added to P21.

```
CreateDriver(InputRoadSide){
        String Type
        Float RV
        RV = uniform_distribution(0,1)
        //set type of vehicle
        If (RV < 0.07 ) { Type = "HGV" } else { Type = "Car" }
        //if HGV, determine and set speed
        If (Type == "HGV" ) {
                RV2 = Normal_Distribution(47.6, 8.76)
                if ( RV2 < 5) { RV2 = 5 }
                if ( RV2 > 100 ) { RV2 = 100 }
        }
        //if Car, determine and set speed
        Else {
                RV2 = Normal_Distribution(50, 8.57)
                if ( RV2 < 5 ) { RV2 = 5 }
                if ( RV2 > 120  ) { RV2 = 120 }
        }
        Float Speed
        //convert speed to m/s
        Speed = RV2 * 0.44
        //Set compliance level
        RV3 = uniform_distribution(0,1)
        ComplianceL As Integer
        If ( RV3 < 0.989772352 ) { ComplianceL = 1 }
        Elseif ( RV3 < 0.999670076 ) { ComplianceL = 2 }
        Else {ComplianceL = 3 }
        //Create output token
        Output 1'Driver(Vehiclespeed = Speed, VehicleType = Type, Roadside = InputRoadSide,
SpawnTime = SimulationTime, Position = 0, Deceleration = 0, ComplianceLevel = ComplianceL)
}
```

Figure 5-15 CreateDriver Function.

The third element of the road vehicle module models the passage of road vehicles who do not react to crossing events, either because they are non-compliant or there are no perceptible events. The Petri net structure is shown in Figure 5-16.

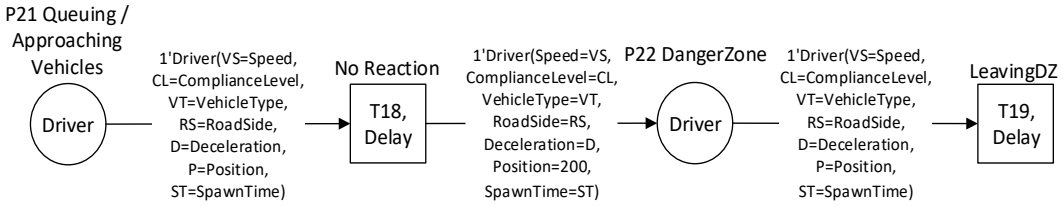

Figure 5-16 CPN Road Vehicle Module Part 2

Transition 18 models the movement of a road vehicle from the spawn point to the crossing. It is enabled by tokens in P21, when it fires it moves these tokens to P22. Transition 18 calculates its delay time using a simple distance travelled equation: d = vt, where d = distance to the crossing, v = vehicle speed, and t = time till the vehicle arrives at the crossing, solved for t, shown Figure 5-17. No effects of deceleration are considered here as this transition models the scenario where a driver has no cause for braking. The position field held within the DriverToken is also updated to 200m.

```
61.  T18 Delay( VS ) {
62.          Delay = 200 / (VS)
63.          Return Delay
64.  }
```

Figure 5-17 Transition 18 Delay Function.

The function which calculates the delay for transition 19 is shown in Figure 5-18. This is similar to that of Transition 18, however this delay must account for the vehicle length as the entirety of the vehicle must clear the crossing to be safe. It is assumed that HGVs are 15m long, and cars 5m long. The width of the passing train is assumed to be 3m.

```
65.  T19 Delay( VT, VS, P ) {
66.          //set vehicle length from type
67.          if (VT == "HGV") { VehicleLength = 15 }
68.          else { VehicleLength = 5 }
69.          //calc time till vehicle leaves crossing
70.          Delay = (3 + VehicleLength) / (VS)
71.          //update position
72.          P = 200 + 3 + VehicleLength
73.          Return Delay
74.  }
```

Figure 5-18 Transition 19 Delay Function (Partial).

Transition T20 competes for any tokens present in P21. This transition models the scenario when an approaching driver reacts to either the level crossing's protection system or an approaching/passing train. Transition T20 has a delay to account for the driver's reaction time, this is taken to be 0.7seconds from the UK Highway Code. The structure of the Petri net is shown in Figure 5-19. Test arcs to the crossing protection system places and other places in the traffic module allow T20's guard function to evaluate whether a driver will react given the circumstances and their level of compliance.

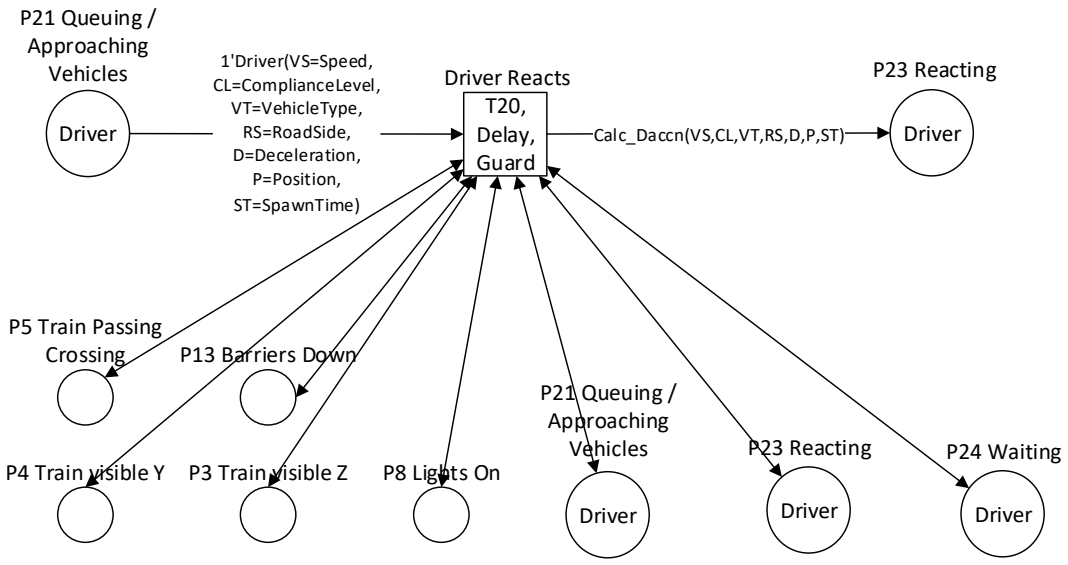

Figure 5-19 Road Vehicle Module Part 3

If a driver is fully compliant and the crossing operating, lines 94-103 will enable the transition allowing the driver to react. Should a train be approaching, but the crossing protection systems not be operating, line 100 allows the driver to react if both the train is visible and they are a driver who will visually check the crossing. Line 98 allows the driver to react if the train is passing the crossing directly in front of the driver.

The behaviour of drivers who may ignore lights, but not barriers is determined over lines 105-116. Line 107 forces them to react to the protection system if there is a potential witness to their hazardous driving. Otherwise lines 109-111 force the driver to react if either the barrier is down, or a train is currently passing the crossing. If the driver is aware of a visibly approaching train, line 113 will model a reaction.

Lastly, the most hazardous drivers who will disobey both barriers and lights are modelled via lines 118-127. This is same as the previous hazardous driver category, except the line which enables the transition when a token is present in Place 13 has been removed. As such this driver will not react to any protection systems unless there is a witness present. On line 129 the position of the vehicle is updated to reflect its position when braking begins.

```
T20 Guard (CL, RS, P3, P4, P5, P8, P13, P21, P23, P24, SimulationTime ) {
        //Find state of protection system operation
        bool ProtectionActive = false
        if ( TokenPresent(P13) ) { ProtectionActive = true }
        if ( TokenPresent(P8) ) { ProtectionActive = true }
        //Is a witness present?
        bool Witness = false
        if ( TokenPresent(P24) ) { Witness = true }
        elseif ( TokenPresent(P23) ) { Witness = true }
        elseif ( TokenPresent(P21) ) { Witness = true }
        //Is an oncoming train visible? And on which side?
        bool TrainVisible = false
        if (TokenPresent(P3) & RoadSide == "Y" ) { TrainVisible = true }
        if (TokenPresent(P4) & RoadSide == "Z" ) { TrainVisible = true }
        //Does driver check for visibly approaching train?
        bool DriverAware = false
        RV = UniformDistribution(0,1)
        If (RV < 0.333) { DriverAware = true }
        //Driver modelling for compliance level 1
        if (CL == 1) {
                //driver reacting to protection system if active
                if (ProtectionActive == true) { Transition = Enabled }
                //driver reacting to train passing directly ahead
                elseif ( TokenPresent(P5) ) { Transition = Enabled }
                //driver reacting to approaching train in distance if checked for
                elseif ( TrainVisible == true & DriverAware == true) { Transition = Enabled }
                //otherwise no reaction
                else { Transition = Disabled }
        }
        //Driver modelling for compliance level 2
        elseif (CL == 2) {
                //driver reacting to protection system if active and witness present
                if (ProtectionActive == true & Witness == true) { Transition = Enabled }
                //driver reacting to barriers if lowered
                elseif ( TokenPresent(P13) ) { Transition = Enabled }
                //driver reacting to train passing directly ahead
                elseif ( TokenPresent(P5) ) { Transition = Enabled }
                //driver reacting to approaching train in distance if checked for
                elseif ( TrainVisible == true & DriverAware == true) { Transition = Enabled }
                //otherwise no reaction
                else { Transition = Disabled }
        }
        //Driver modelling for compliance level 3
        elseif (CL == 3) {
                //driver reacting to protection system if active and witness present
                if (ProtectionActive == true & Witness == true) { Transition = Enabled }
                //driver reacting to train passing directly ahead
                elseif ( TokenPresent(P5) ) { Transition = Enabled }
                //driver reacting to approaching train in distance if checked for
                elseif ( TrainVisible == true & DriverAware == true ) { Transition = Enabled }
                //otherwise no reaction
                else { Transition = Disabled }
        }
        //Update vehicle position
        P = VS * (SimulationTime - ST)
}
```

Figure 5-20 Transition 20 Guard Function.

When transition T20 fires it signifies that a driver is now reacting by braking. Arc function 'Calc_Daccn' calculates the rate at which the vehicle will decelerate. The contents of this

function are shown in Figure 5-21. Line 133 determines the stopping distance by interpolating the values form the Highway Code. This calculation has been omitted for brevity. The deceleration value is calculated on the following line, using the formula derived from equations of motion. The derivation of this formula can be seen in appendix A.

```
131. Calc_Daccn(VS, CL, VT, RS, D, P, ST) {
132.         //Find stopping distance from highway code
133.         StoppingDistance = LinearInterpolationHighwayCode(VS)
134.         //Calculate deceleration
135.         D = VS^2/(2*StoppingDistance)
136.         //create output token
137.         Output 1'Driver(VehicleSpeed = VS, VehicleType = VT, Roadside = RS, SpawnTime = ST,
     Position = P, Deceleration = D, ComplianceLevel = CL)
138. }
```

Figure 5-21 Function for Calculating Deceleration Rate.

The final part of the road vehicle module is shown in Figure 5-22. This models drivers stopping at the crossing. When a token representing a vehicle is placed in P23, transition T21 is enabled. This models the vehicle braking. Should the driver stop before the crossing a token is placed into P24, signifying the driver is now waiting at the crossing. However, should the driver fail to stop in time, as may occur if a driver is surprised by a train, they may still enter the crossing's danger zone. In this case the token is placed in P22, rather than P24. Should this occur, the driver may either continue decelerating as they leave the danger zone, or come to rest in the danger zone. When this occurs it is assumed each vehicle will take an additional 10 seconds to accelerate from stationary out of the crossing.

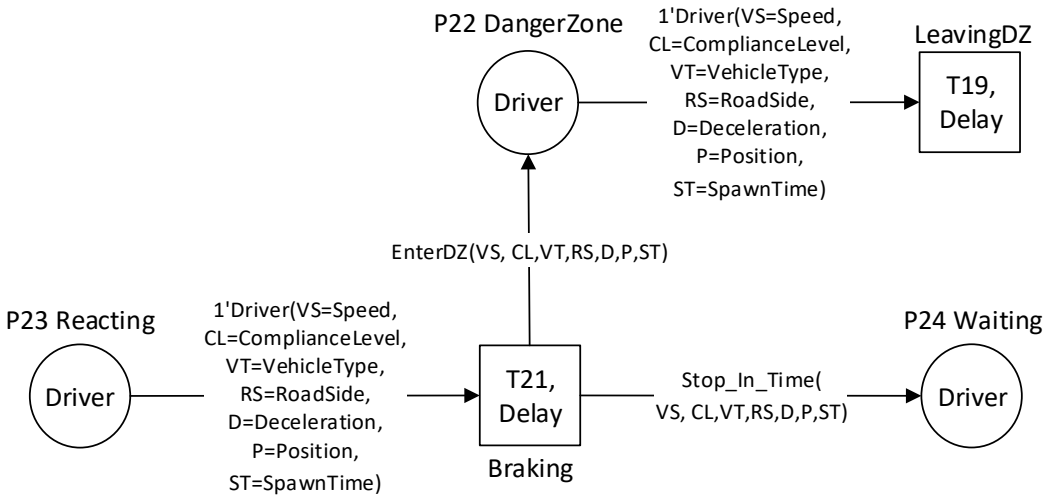

Figure 5-22 Road Vehicle Module Part 4

The function which calculates both the delay for transition T21 and the vehicle speed when it fires is shown in Figure 5-23. Lines 142-143 determine the vehicles length from whether it is a car or a HGV. Line 144 calculates the distance from the vehicles current position to the

shared area of the crossing. Over lines 147 – 151 it is determined if the vehicle will come to a stop before it reaches the shared area of the crossing. This is done using the equations of motions derived in Appendix A. If the discriminant from equation A-6 is less than 0, there are no real roots of the formula for the variables used as the vehicle does not reach the crossing. Therefore there is no real value for the time for the vehicle to reach the crossing. Where this is the case, Line 149 calculates the time for the vehicle to come to a stop and uses this as the delay for transition 21. The time to reach the crossing for the scenario where the road vehicle enters the shared area of the crossing whilst braking is calculated over lines 152-155, using formula A-6 from Appendix A.

```
139. T21 Delay(VS, VT, P, D ) {
140.            Bool StoppedInTime = false
141.            //Set vehicle length from type
142.            if (VT == HGV) { VehicleLength = 15 }
143.            else { VehicleLength = 5 }
144.            DistanceToDZ = 200-P
145.            //Calc Discriminant from equation of motion formula
146.            Discriminant = VS^2 + 2 * D*(3+VehicleLength)
147.            if (Discriminant < 0) {
148.                        //Vehicle stopped before entering crossing
149.                        Delay = (VS / D)
150.                        VS = 0
151.            }
152.            else {
153.                        //Calc time till vehicle enters crossing and speed when it does
154.                        Delay = (VS + sqrt(VS^2  + 2 * D*(3+VehLength))) / D
155.                        VS = VS – VS*D*Delay
156.            }
157.            Return Delay
158. }
```

Figure 5-23 Transition 21 Delay Function.

There are two arcs leaving transition 21, however the Driver tokens are conserved. The arc functions EnterDZ and Stop_In_Time determine whether place P22 or P24 receives the driver token. If the vehicle's speed is 0 m/s when the transition fires the token is moved to P24, showing the vehicle has come to a stop in front of the crossing, but not within it. If the vehicle is still moving when the transition fires the token is moved to P22, modelling the scenario where the vehicle enters the crossing whilst braking. These functions are displayed in Figure 5-24.

```
159. EnterDZ(VS, CL, VT, RS, D, P, ST) {
160.            If (DriverToken(Speed) != 0) {
161.            Output 1'Driver(VehicleSpeed = VS, VehicleType = VT, Roadside = RS, SpawnTime = ST,
     Position = P, Deceleration = D, ComplianceLevel = CL)
162.            }
163. }
164. Stop_In_Time(VS, CL, VT, RS, D, P, ST) {
165.            If (DriverToken(Speed) == 0) {
166.            Output 1'Driver(VehicleSpeed = VS, VehicleType = VT, Roadside = RS, SpawnTime = ST,
     Position = P, Deceleration = D, ComplianceLevel = CL)
167.            }
```

Figure 5-24 Transition 21 Arc Functions.

Reviewing Figure 5-22 it can be seen that transition 19 must model both the passage of vehicles traveling at constant speed through the crossing, but also vehicles who are braking. This requires an expansion to the delay function presented above. The expanded function is shown in Figure 5-25, covering lines 180-186. This section of the function operates using the same formulas and structure as that of transition 21. The discriminant of formula A-6 is calculated to determine if the road vehicle clears the shared area of the crossing, or if it stops within the crossing. If the vehicle stops within the crossing the transition delay is the time for the vehicle to stop plus an additional ten seconds for the vehicle to accelerate out of the crossing.

```
168. T19 Delay(VS, VT, P, D) {
169.            //set vehicle length from type
170.            if (VT == HGV) { VehicleLength = 15 }
171.            else { VehicleLength = 5 }
172.            //if vehicle travelling at constant speed
173.            if ( D == 0 ) {
174.                        //calc time till vehicle leaves crossing
175.                        Delay = (3 + VehicleLength) / (VS)
176.                        //update position
177.                        P = 200 + 3 + VehicleLength
178.            }
179.            //if vehicle braking as it enters crossing
180.            if ( D != 0 ) {
181.                        //calculate discriminant from equation of motion
182.                        Discriminant = VS^2 + 2 * D*(3+VehicleLength)
183.                        //if less then zero vehicle does not clear crossing before stopping
184.                        if (Discriminant < 0) { Delay = (VS / D) + 10 }
185.                        //vehicle clears crossing
186.                        else { Delay = (VS + sqrt(VS^2  + 2 * D*(3+VehLength))) / D }
187.                        Return Delay
188.            }
189. }
```

Figure 5-25 Transition 19 Delay Function Expanded.

5.4 Analysis

The model was simulated using the CPN road traffic module for the passage of 10^{10} trains for each average traffic flow rate, for each crossing operational state and alterations in other parameters. The number of trains which collided with vehicles was divided by the total number of train passages to obtain the probability of a collision occurring per passing train. This is shown for normal crossing operations in Figure 5-26. When varying the traffic flow rate, a non-linear relationship is revealed, consistent with the results of data analysis by Stott (1987). Stott found that as the frequency of vehicles passing a crossing increased, so did the probability of a collision occurring, but that with further increases the collision probability fell. Stott hypothesised this was due to a reluctance among driver to be witnessed breaking traffic laws.

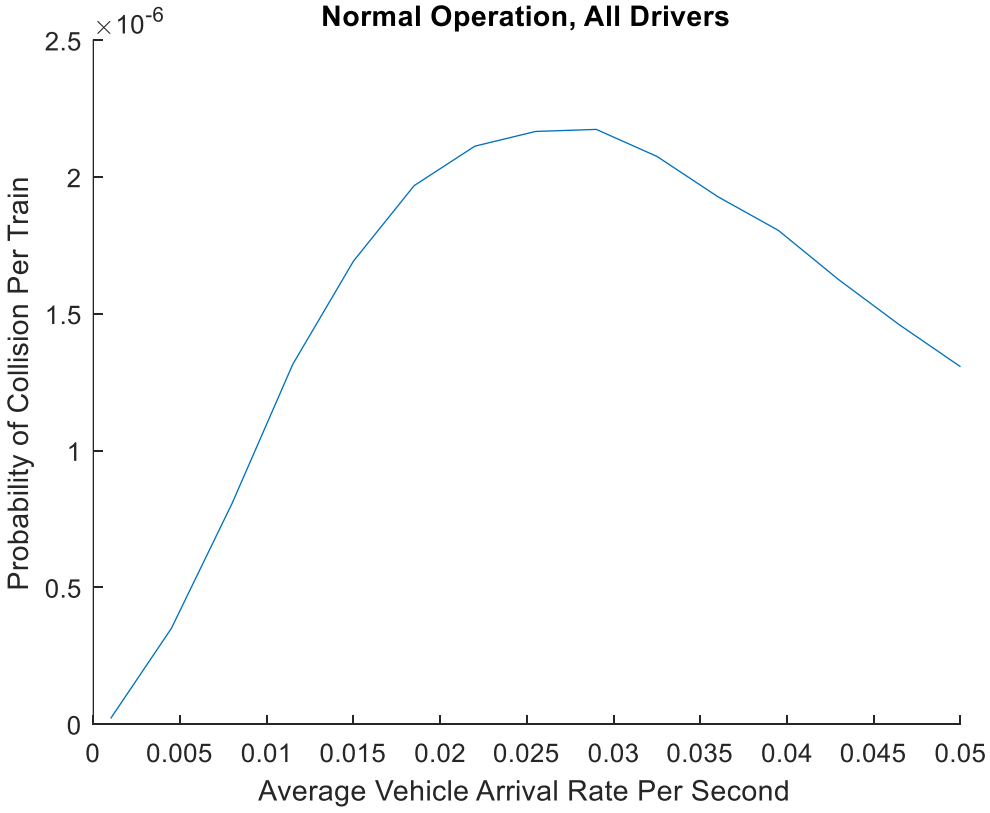

Figure 5-26 Collision Probability Vs Average Traffic Arrival Rate

This nonlinear relationship is due to the balance between probability of vehicle and train to be on a collision course, and the opportunity for a driver to disobey the rules of the road without being witnessed or impeded. The increase in collision probability from 0 vehicles per second to 0.025 stems from a greater chance of a non-compliant driver being on course to collide with a train. The decline from 0.025 vehicles per second onwards reflects how the now quite regularly trafficked crossing will often provide witnesses or obstructions to any

driver who might consider disobeying the crossing's warnings of an oncoming train. The simulated collisions which occur when the crossing is operating normally are all caused by the entirely non-compliant drivers. This is because the partially compliant drivers who only obey the barriers still have approximately 20 seconds to traverse the crossing if they enter just as the barrier comes to rest in the lowered position.

Figure 5-27 shows the expected number of collisions per year for a crossing which does not fail hazardously. This assumes 80 trains pass per day, 365 days per year.

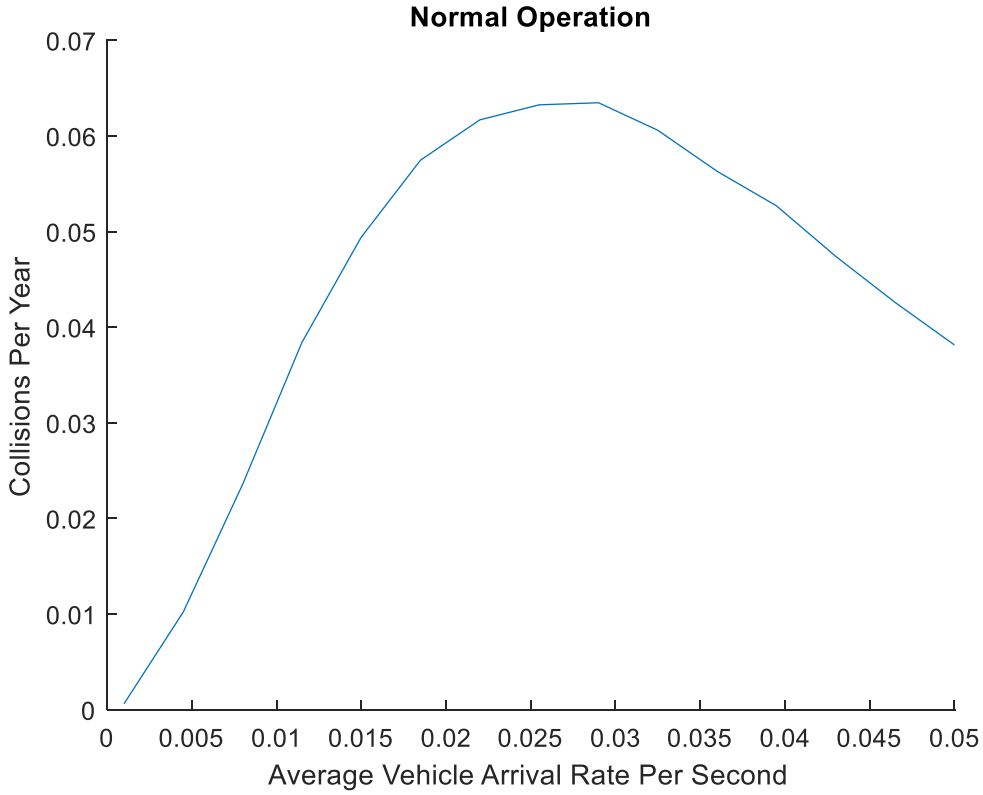

Figure 5-27 Collisions per Year Vs Average Vehicle Arrival Rate

5.4.1 Road Traffic Light Failure

The model was used to test the scenario where the road traffic lights fail to operate on either side of the crossing, leaving only the descent of the barriers to warn traffic of the approaching train. This produced collision probabilities very similar to that of normal operation, shown Figure 5-28, note that the scale is different for this graph and those that follow. The peak collision probability has increased, however this is the only observable difference. As per normal operation, only entirely non-compliant drivers are responsible for collisions. The increase occurs as there is less chance of a non-compliant driver encountering a complaint driver waiting at the crossing, as now compliant drivers will not stop until the barrier begins to descend.

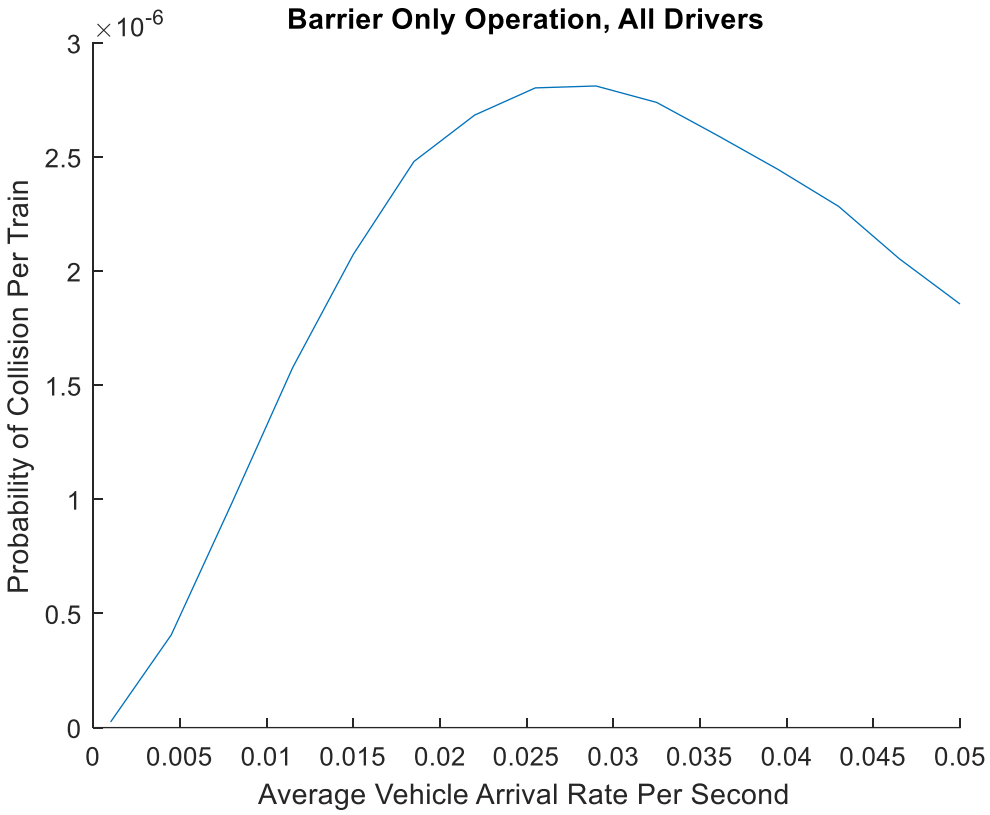

Figure 5-28 Collision Probabilities for Barrier Only Operation.

5.4.2 Road Traffic Barrier Failure

The model was also used to explore the scenario where the road traffic barriers have failed in the raised positon, leaving only the road traffic lights to warn drivers of the approaching train. Shown in Figure 5-29, this resulted in a considerable increase in the probability of a collision occurring, with the peak collision probability approximately 30 times higher than normal operation. This stark increase occurs because now partially compliant drivers will behave the same as non-compliant drivers as the lowered barrier no longer causes them to stop. The 30 fold increase in risk occurs because the non-compliant fraction is 30 times larger, as now 1% of all drivers will not comply with the crossing's warning systems. The proportions of driver types involved in collisions reflects this, with 30 out of every 31 collisions being caused by partially compliant drivers, and the remaining collisions caused by entirely non-compliant drivers. As the road traffic lights are still operating, no fully compliant drivers are the cause of a collision.

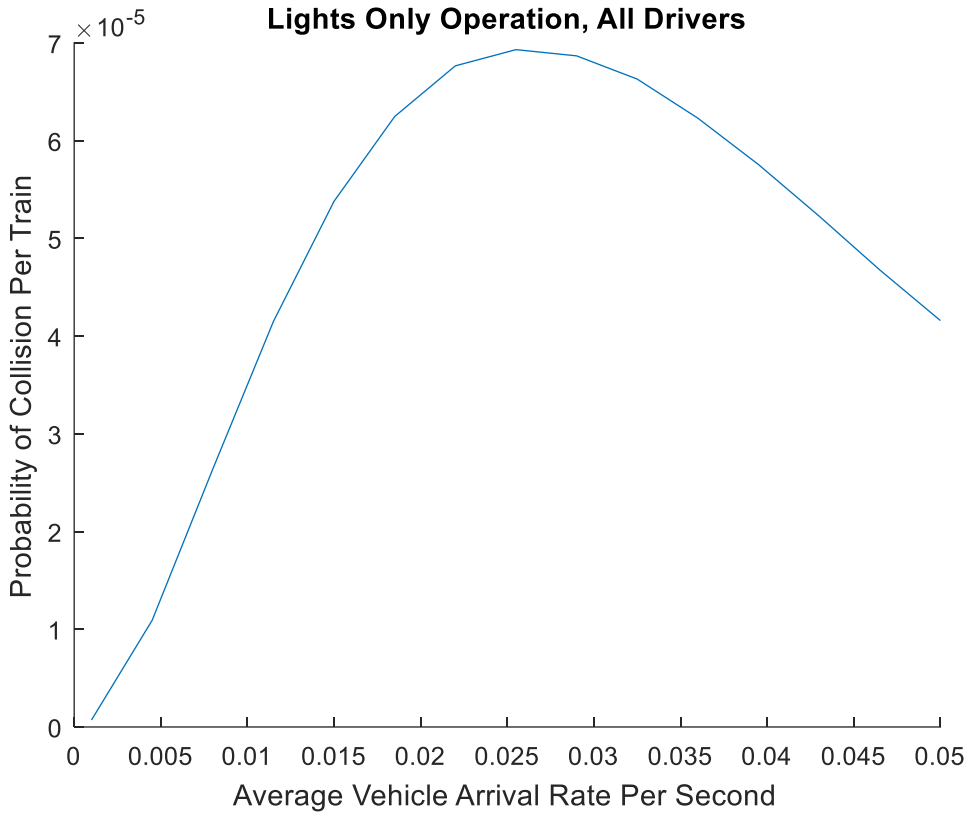

Figure 5-29 Collision Probabilities for Lights Only Operation.

5.4.3 Complete Protection System Failure

Finally the model was used to estimate collision probabilities should all the crossing's protection systems fail to operate as a train approached. Unsurprisingly, this produced the most adverse probabilities of collisions between road vehicles and trains. Figure 5-30 shows a linear increase in collision risk as the average arrival rate of vehicles increases. In this scenario all driver compliance levels have an equal likelihood of being involved in a collision, as there are no active protection systems to comply with.

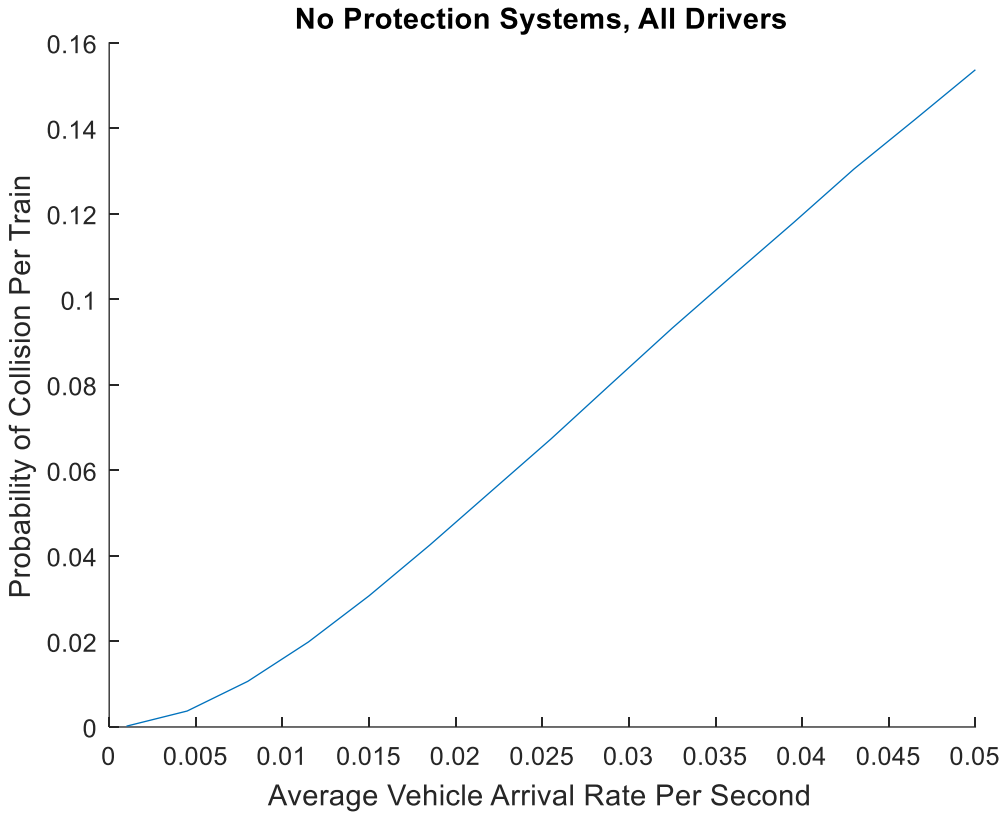

Figure 5-30 Collision Probabilities for Complete Failure of Protection Systems.

The graph depicted in Figure 5-31 uses a Semi Log plot to facilitate clear comparison of the changes in collision risk between each level of protection system operation.

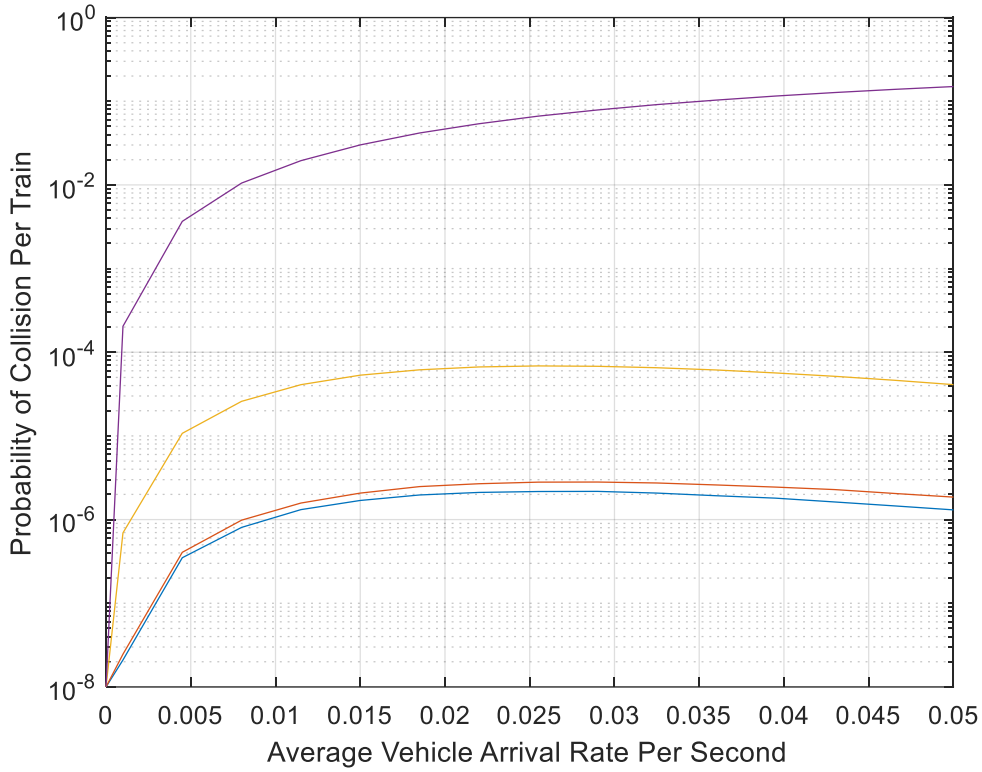

Figure 5-31 All Collision Probabilities Plotted Using a Semi Log Scale for Comparison.

5.4.4 Sensitivity Analysis

Several sensitivity analyses were performed to determine the extent to which the variables used in the model could affect the collision probabilities the model produces.

5.4.4.1 Reaction Time Sensitivity

The simulations were repeated a further five times with linearly varying reaction times for each driver from 0.1 seconds to 1.3 seconds. This revealed the collision probability to be somewhat sensitive to reaction times, however less than other parameters outlined later. The range of reaction times chosen yielded a 20% difference in collision rates for normal crossing operation. Increases in reaction time lead to corresponding increases in collision probability. Similarly, decreases in reaction time reduced collision probability, though the effects were much more pronounced. This is unsurprising, as faster reaction times allow a driver to brake sooner should they observe a passing train, increasing the chance they will stop short of the train. This effect can be seen in Figure 5-32, showing the effects on reaction time during a scenario where no protection systems are operating. Altering reaction times did not change the proportions of collisions being caused due to driver compliance level.

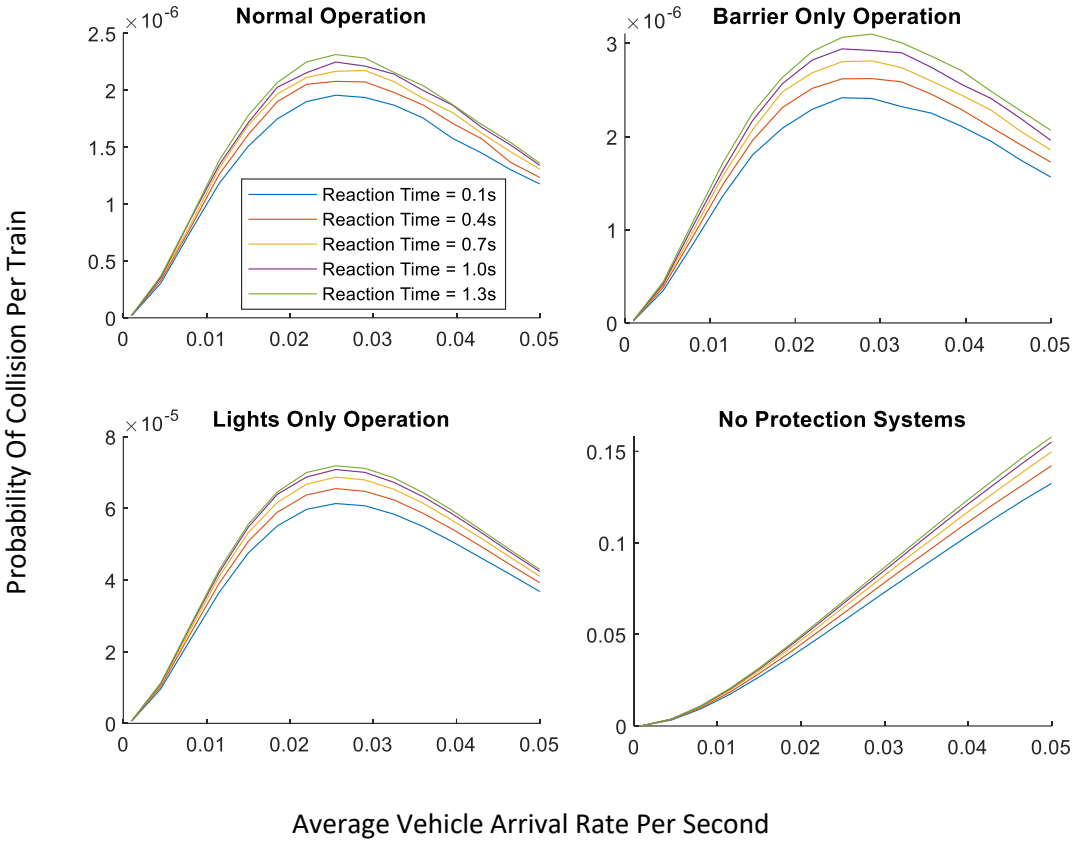

Figure 5-32 Reaction Time Sensitivity Analysis Results

5.4.4.2 Stopping Distance Sensitivity

The sensitivity of the model to stopping distance was tested by applying a scale factor to all the stopping distances generated for the road vehicles simulated. The factor applied varied between 0.4 and 1.6. This produced very similar results to the reaction time variation shown previously. This is unsurprising, as both ultimately determine the time it takes for a vehicle to stop, and the distance travelled doing so. Figure 5-33 shows the effects of varying stopping distance on collision probability. Comparing this Figure 5-32 shows just how similar the effect of varying stopping distances is compared to varying reaction times. As with the reaction time tests, all other crossing levels of operation yielded similar differences.

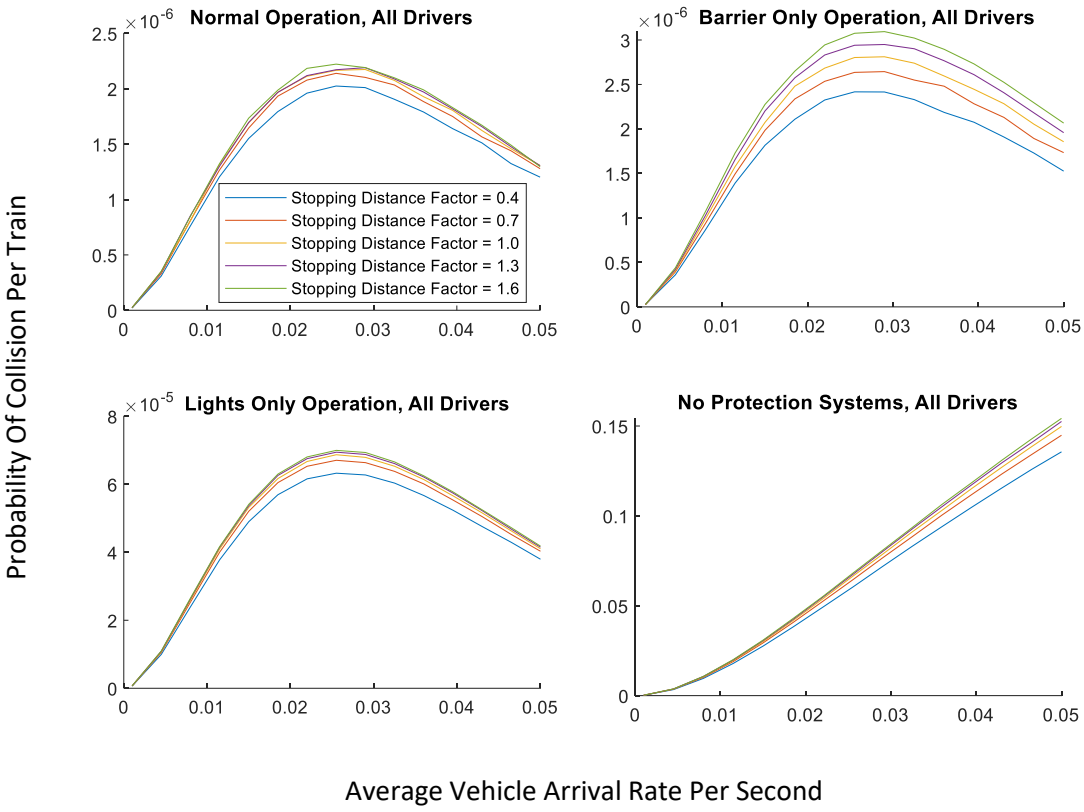

Figure 5-33 Stopping Distance Sensitivity Analysis Results

5.4.4.3 Sensitivity to Proportion of Heavy Goods Vehicles

The proportion of Heavy Goods Vehicles in the model was varied from 3% to 11% to determine its effect. There are two differences between cars and HGVs in the model, HGVs are longer and drive slightly slower on average. This had little effect on the probability of a collision, shown Figure 5-34. The results show increasing number of HGVs increases collision probability slightly. This is due to the increased length of HGVs creating a larger window where train and HGV can be on a collision course, however the effect is very slight. This conflicts with the historical observations at level crossings, where HGVs collisions occur disproportionally to normal cars (Davis Associates Limited, 2005). This could be due to HGV drivers displaying higher levels of non-compliance than the typical road user, not featured in this model.

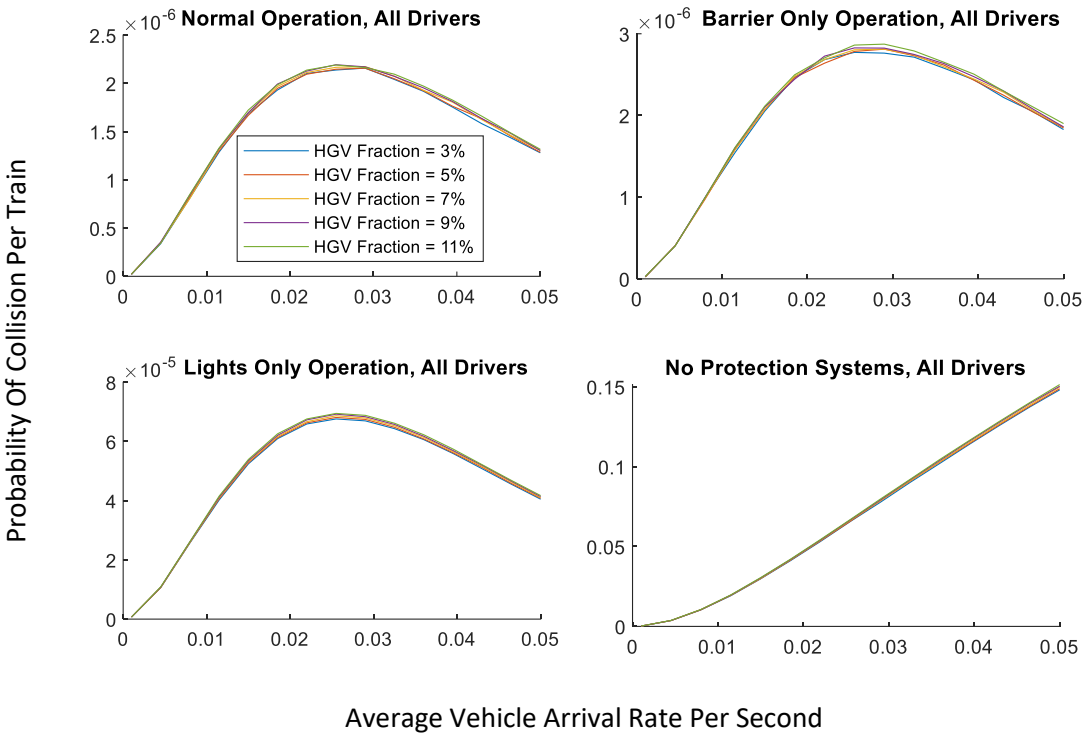

Figure 5-34 HGV Proportion Sensitivity Analysis Results.

5.4.4.4 Car Length Sensitivity

As vehicle length increases, so does the time window where that vehicle and a train can be on course to collide. As such, the effect of changing the length of road vehicles has been investigated to determine how it affects the probability of a collision. This was carried out by varying the length of all road vehicles in the simulation from 2 to 6 metres. The results of the analysis, shown Figure 5-35, displays an even but slight increase in collision probability for each additional 1m vehicle length.

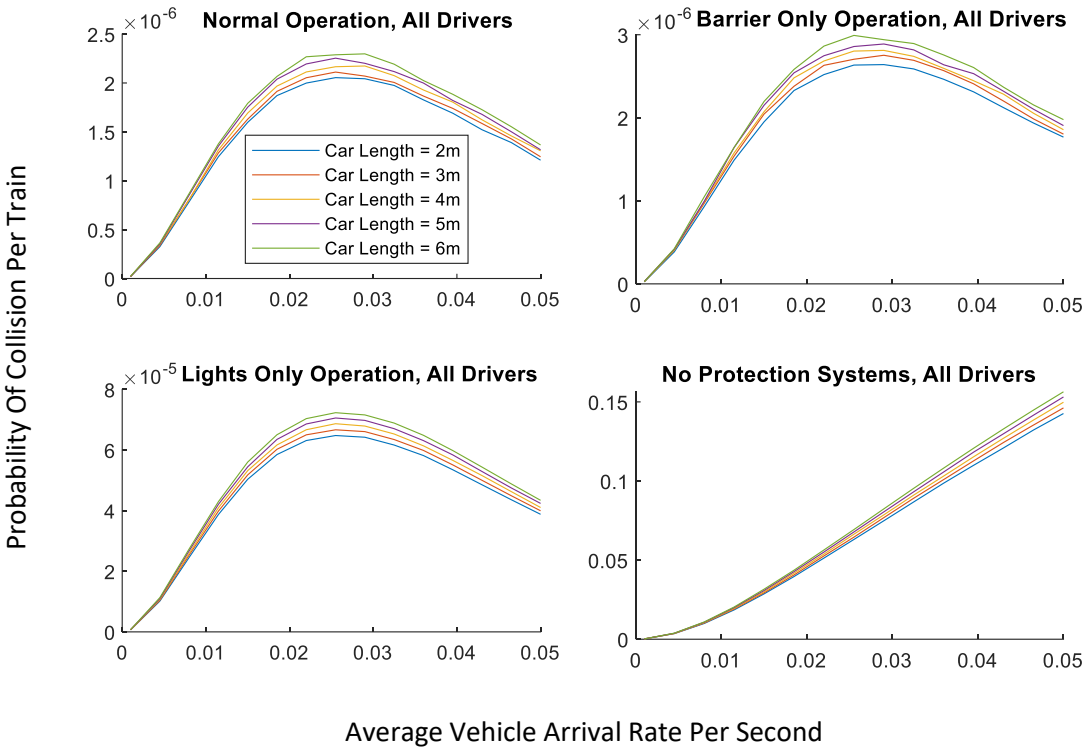

Figure 5-35 Car Length sensitivity Analysis Results

5.4.4.5 Driver Compliance Fraction Sensitivity

Driver compliance controls which level crossing protection signals a driver will respond too. To determine the model's sensitivity to the quantity of partially, or entirely non-compliant drivers, these values were varied to determine the extent of their effects on collision probability. Varying the percentage of partially non-compliant drivers in the simulation from 0.55% to 1.27% produced striking results, shown in Figure 5-36.

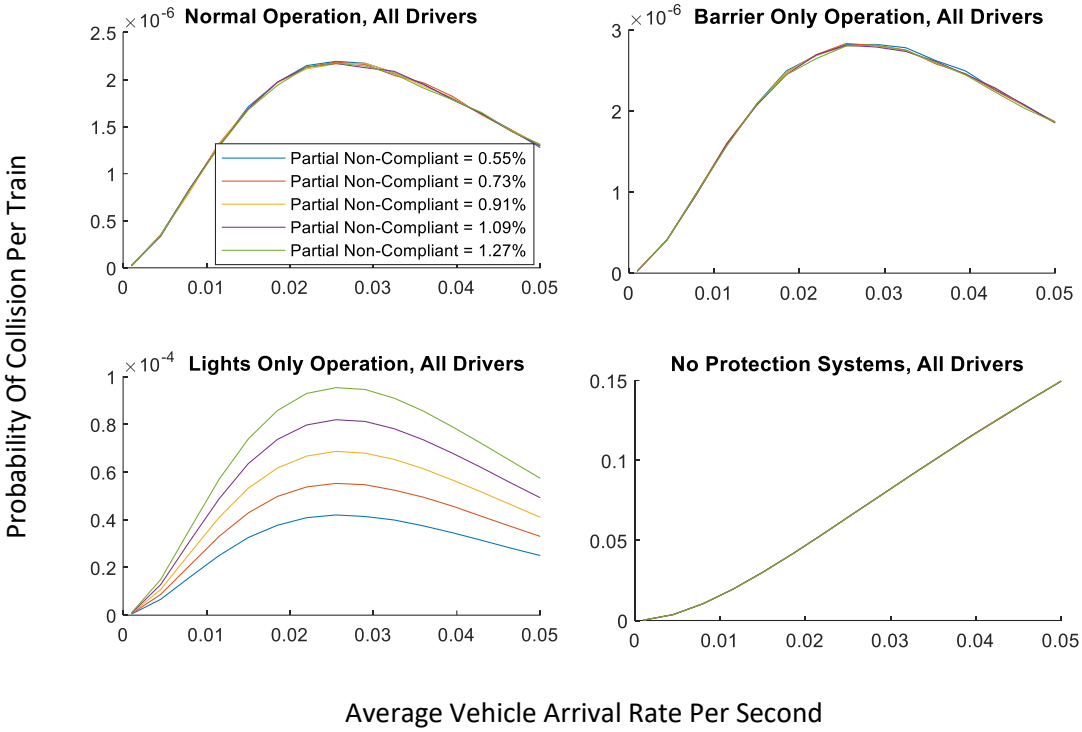

Figure 5-36 Partial Compliance Proportion Sensitivity Analysis Results.

The results show that varying the fraction of partially compliant drivers which pass the crossing has no effect on collision probability when the crossing is operating normally, when the barriers are operating only, or when no protections systems are active. However, there is a large effect when the crossing is operating with only the road traffic lights functioning. This effect occurs because there are only two scenarios when partially compliant drivers can be involved in collisions: When no protection systems are active at the crossing, and when the barriers are not operating. The partially compliant driver fraction has no effect when no protection systems are active as all drivers effectively have the same behaviour in this scenario. Whereas, when no lights are active the majority of collisions are caused by partially compliant drivers, as the only protection system which can cause them to comply is no longer active.

A further analysis was conducted on the effects of varying the entirely non-complaint driver fraction from 0.02% to 0.046%, shown Figure 5-37. This had a considerable effect on the probability of a collision occurring when the crossing was operating normally and during barrier only operation, this was expected as in these circumstances only entirely non-compliant drivers are involved in collisions. There was a slight effect on collision risk when only the road traffic lights were operating, as the slight increase in entirely non-compliant drivers is dwarfed by the large number of partially compliant drivers causing collisions. Again,

there was no effect when the crossing's protection system were not operating as each driver has the same behaviour in this scenario.

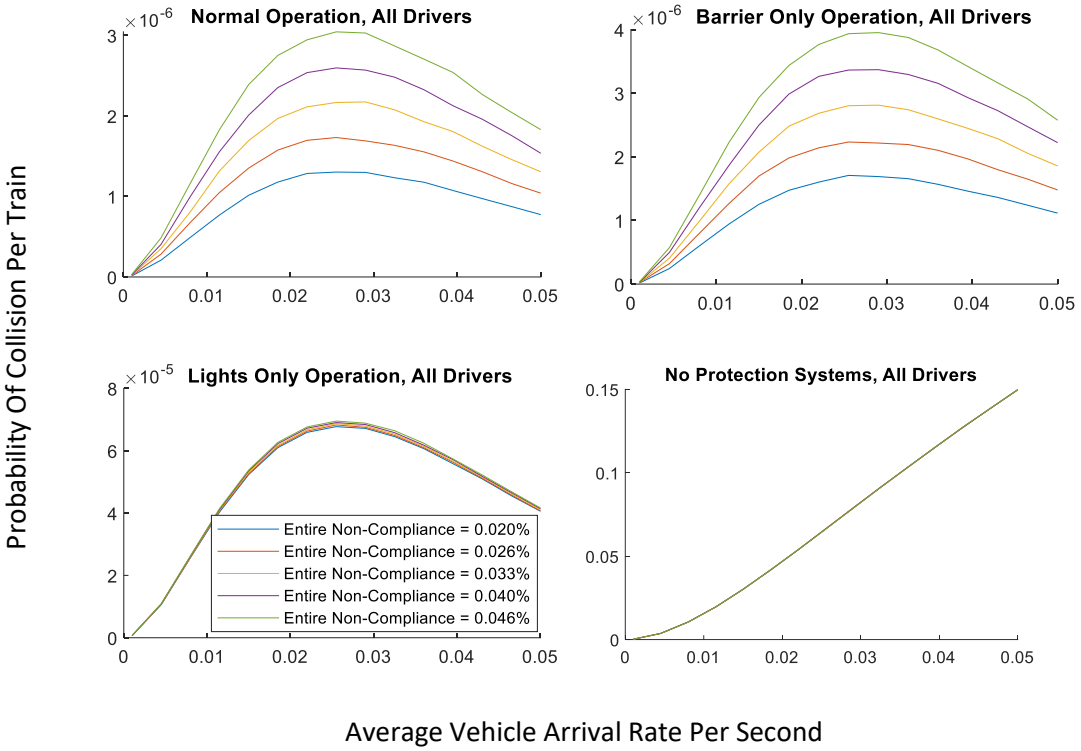

Figure 5-37 Entirely Non-Compliant Driver Fraction Sensitivity Analysis Results.

5.4.4.6 Sensitivity Analysis of Train Passage Time

In the model, the speed and length of the train are not directly modelled. These characteristics are instead combined by modelling the duration of time it takes for the train to pass the crossing. This duration was varied to find out the extent to which it will affect the collision probability results. The results presented in Figure 5-38 show that train passage duration has a considerable effect on the probability of a collision occurring. Simply, this is due to a longer train creating a larger window in which car and train can be on a collision course. Interestingly, the effect lessens with progressive increases in duration. Consider a train which takes 5 seconds to pass. A vehicle may collide with the front portion of the train, but a driver on course to hit the rear most portion of the train will have at least 5 seconds to react and brake. Thus increasing train passage durations yield diminishing increases in collision probability. The magnitude of the effects of train length reflect that the majority of collisions occur by the road vehicle hitting the side of the train rather than vice versa.

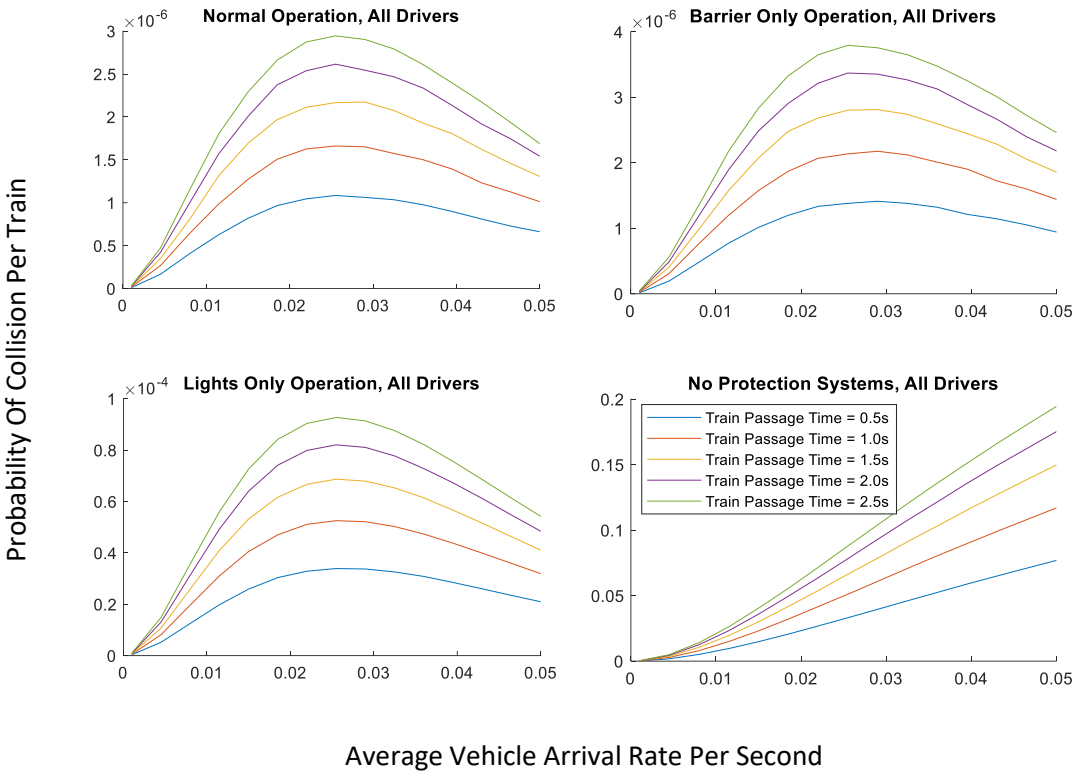

Figure 5-38 Train Passage Duration Sensitivity Analysis Results.

5.4.4.7 Summary of Sensitivity Analysis

The sensitivity analysis has shown which variables have the most influence on the outcome of the modelling. The most pronounced effect came from the number of road vehicles trafficking the crossing, whose increase generally corresponded to increases in risk between 0 and 0.025 vehicles per second, and slightly decreased between 0.025 and 0.05 vehicles per second.

Road vehicle driver reaction times and stopping distances have a moderate effect on collision risk. The model is largely insensitive to sensible ranges of HGV vehicles, though this conflicts which historical evidence suggesting HGVs are a larger collision risk than other road vehicles. Driver compliance fractions had striking effects, often made worse by concurrent crossing protection system failures. Finally, the time required for a train to pass the crossing also had a striking effect on collision risk.

From this, it can be concluded that the most important factors when using this model to predict collisions are road vehicle arrival rate and the entirely non-compliant driver fraction.

5.5 Chapter Summary

This chapter has reviewed Petri net models in literature for predicting level crossing collision risk and found then unsuitable to fulfil the aims and objectives of this thesis. Specifically, they were not able to either predict risk as a consequence of level crossing asset operation state, nor were they effective in determining the most suitable framework modelling methodology.

A CPN model was developed to meet this need and found to be superior to an equivalent SPN due to the flexibility CPN modelling allows. In the absence of sufficient historical crash data, and an expected continued absence of data, the model was designed around simple core tenants of driving which can be understood in isolation in the hope that the model may prove useful despite not being directly verifiable. It was found that failure modes which affected crossing protection system function could have significant effects on the probability of collisions occurring between road and rail vehicles.

A sensitivity analysis was conducted on this model to determine the most important parameters, whose acquisition or estimation should be prioritised. This revealed road traffic volume to be an extremely important factor, already collected and used in contemporary modelling efforts (Turner, 2008), but also driver compliance fraction. Driver compliance has not been directly surveyed at UK automatic level crossings, however Network Rail's increasing use of CCTV at automatic level crossings will allow these parameters to be obtained in the future.

In order to calibrate and validate the model, video from the crossings could be analysed using computer vision software to count the number of vehicles passing the crossing, and determine whether they were passing despite lights or barriers being active. However, the purpose of Network Rail installing these cameras is to dissuade drivers from ignoring the crossing protection systems with the threat of fines and driving licence endorsements. If effective, this will reduce the number of non-compliant drivers and save lives, however the data produced will only be representative of crossings where there are CCTV cameras deployed.

Covert CCTV may be deployed to obtain this data without affecting the data obtained, but this poses ethical issues, as disclosure of the covert CCTV may save lives by preventing non-compliant behaviour. Similar problems are faced when considering using driving simulators. When drivers are aware they are being observed and making simulated journeys, their decision-making processes may no longer be representative their driving behaviour in real

life conditions. Whilst there are different methods available to collect the data required to calibrate the model, it is difficult to avoid changing the nature of the system by observing it.

Through modern technology, there is the future potential to capture data on user behaviour at level crossings with minimal effect on the quality or outcome of the data caused by its collection. Many smart devices routinely capture GPS tracking data which can be used for mobility research (Ruktanonchai, et al., 2018). Such data has been used during the Coronavirus pandemic to quantity changes in transport and mobility. Hypothetically, this data could be used to identify road traffic volumes passing a level crossing. It could be used in combination with train timetables to detect vehicles which have probably passed the crossing whilst the protection systems were active. Though this type of data is routinely collected automatically, and often without awareness of the device owner, it is unlikely to be a feasible source of data for this research for some time, if at all, as many ethical and anonymity questions regarding its use have yet to be resolved. However, should these questions be resolved in the future, it represents the lowest cost and highest quality solution.

6. Level Crossing System Simulation Model

Automatic Half Barrier Level Crossings (AHBCs) are mechanically and electrically complex. Models are required to determine system behavior when components have failed as the system is too complex for this to be determined by inspection alone. The review of the AHBC system in a previous chapter found no study or documentation on the effects of failures of specific components. The absence of this information limits the accuracy of any asset management model, as it is difficult to model an appropriate response to a component failure without knowing the effect of that failure. To remedy this, a model to simulate an AHBC was created using a Timed Petri Net (TPN). This model simulates the mechanical and electrical operation of an AHBC during both normal operation, and with component failures. A further Coloured Petri Net (CPN) model was created using identical logic to allow comparison between CPN and TPN for this application. This is an example of a functional model within the In2Rail framework, as it relates degradation state to a level of service of the crossing asset.

The simulation model determines the state of the system prior to, during, and after a train passes the crossing. It also shows the operation of elements of the AHBC which are not directly related to the protection system operation. Components can be altered within the simulation to discover the effects of their failure on protection systems, and if the failure produces aberrant behavior that might lead to the discovery of the failure by inspecting engineers. By determining the effects and discoverability of each component failure mode, the scope of the asset management model can be expanded to include these factors. This furthers an aim of this thesis, to create an asset management model for automatic level crossings.

6.1 Simulation Model Framework

The simulation model comprises four major modules, representing the four major mechanical and electrical systems within an AHBC. Many of these modules interact with the others as per a real AHBC. The first module is the Train Simulation Module. This simulates the interaction of a passing train with the AHBCs track circuits. AHBCs use three track circuits per line and four sets of treadles. This module operates the treadles and track circuits at the appropriate times and durations for a given train length and speed.

The next module simulates the operation of the track circuits and relay based control system. This is the largest module due to the complexity of the control system. Using Boolean logic it determines the state of each relay over time. It models the effects of failure of relay

contacts via welding or high resistance failure modes. It also models the failure of the capacitors which create the timed functions of an AHBC via electrolyte leakage reducing capacitor capacitance.

The road traffic lights and barriers are each modelled using a separate module. Barrier and light failure modes are modelled using a black box approach where specific component failure modes are not considered explicitly, only their effects. This approach is much simpler to model, made possible by the comparatively simple barrier and light systems. For the barriers this is a reduced rate of rising/falling or a failure to rise or fall entirely. For the lights, failure of individual bulbs can be specified. The electronic device which causes the lights to flash is modelled as operating either too fast or slow due to failure.

These modules were created using timed Petri nets. No examples of modelling relay logic using Petri nets were found in literature, however examples were found where Petri nets were used to model Ladder Logic Diagrams. Ladder Logic is generally applicable to discrete event control system modelling (Twiss, 1996). This can include older relay logic based control systems such as that used in automatic half barrier crossings, however it is much more commonly applied to modern Programmable Logic Controller (PLC) software design. As such the papers published on Ladder Logic tend to focus more PLC applications. With some development, these methods have been used to translate the level crossing control system logic into a form which can be modelled via TPN.

6.2 Timed Petri Net Simulation Model Structure

The simulation model evaluates the effects of component failures in a deterministic manner, therefore no stochastic transitions are required. The specific type of regular Petri net used is therefore a Timed Petri Net with high level extensions. The only high level extensions used are inhibitor arcs.

6.2.1 Train Simulation Module

The train simulation module simulates the interaction of a passing train with the track circuits of an AHBC. The Petri net which models this is shown in Figure 6-1. The Petri net may start with an initial marking of a single token in Place P-TM1. When simulated, this enables the transition T-TM1 which fires after a delay of 1,000seconds. This is to allow the control system time to reset itself if the specified failed components trigger this action.

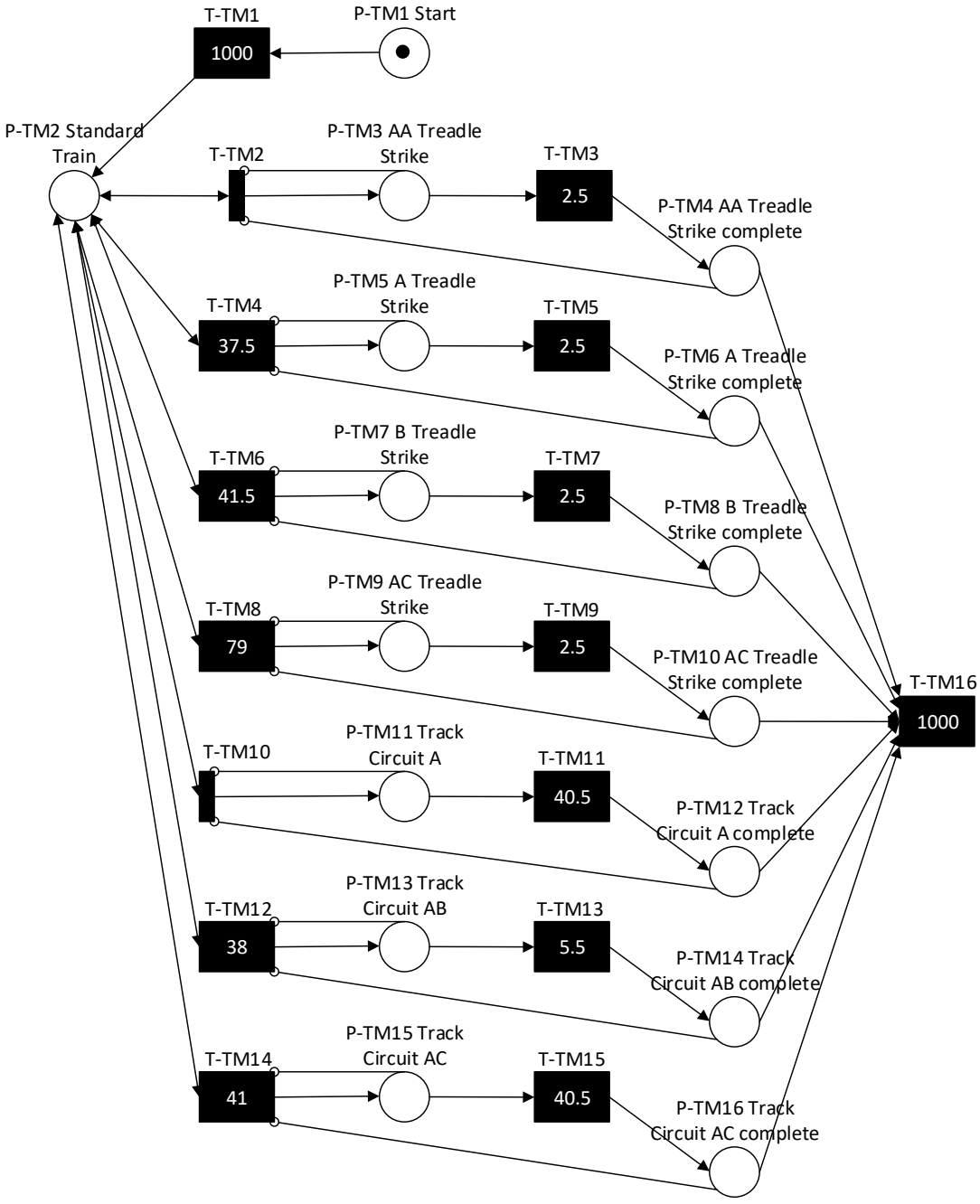

Figure 6-1 Train Simulation Petri Net Module.

The presence of a token in P-TM2 enables many transitions via a test arc [T-TM2, T-TM4, T-TM6, T-TM8, T-TM10, T-TM12, T-TM14]. These transitions control the time at which the train is expected to reach a given track circuit. T-TM2 and T-TM10 are immediate transitions as this is the first track circuit / treadle the train arrives at. Using track circuit A as an example: Immediately after the train arrives at the crossing T-TM10 fires, placing a token in place P-TM11. This signifies that the train has reached the A track circuit. This is connected to the track circuit and relay module via test and inhibitor arcs (not shown) allowing this module to simulate the appropriate responses. An inhibitor arc from place P-TM11 to transition T-TM10

prevents the transition from firing again. This place also enables transition T-TM11, which models the time for the train to exit this track circuit. After a further 40.5seconds has elapsed transition T-TM11 fires, removing a token from place P-TM11 and placing it in P-TM12. This place signifies that this track circuit event has completed, an inhibitor arc from this place to transition T-TM10 keeps it from firing again. All track circuit events follow this standard structure.

After all track circuit events have completed, transition T-TM16 is enabled. This transition models the headway between trains using the line. When it fires it removes tokens from all the track circuit event complete places. This removes the inhibition on the track circuit event initiating transitions allowing a subsequent trains passage to be simulated. The structure of this Petri net allows it to simulate trains continually until the simulation of the Petri net model is ended.

6.2.2 Relay Based Control System

In their PhD thesis Kurapati (1995) explores the application of Petri net modelling to discrete event control systems. The main thrust of the thesis is demonstrating that Petri nets can be used instead of conventional Ladder Logic Diagram software and demonstrating their superiority. The thesis includes some examples of conversion from Ladder Logic Diagrams to Petri nets for manufacturing control systems. The examples presented are best applied to linearly progressing control systems, where event initiating conditions are not required to be maintained for the event to continue.

This is in contrast to how level crossing control systems function, which often require the initiating signals to be maintained in order for the actions they trigger to be maintained. Lee and Lee (2009) present a similar, but better defined method for conversion where Ladder Logic Diagrams are first converted to Boolean logic, and then replaced with defined Petri net structures which model these logic gates. Like Kurapati their work requires any control action which requires sustained initiating signals to be maintained to be explicitly added to the Petri net structure. Individually considering each potential route through which the termination of an initiating signal may occur is likely to prove time consuming, and may introduce errors when applied to modelling the complex relay logic control system of an automatic level crossing.

In general, such methods like that presented by Lee and Lee, and Kurpati require a clear understanding of how the system behaves both under normal and failure conditions, in order to explicitly programme those behaviours. But the behaviour of the AHBC's system when

components have failed has not been documented. Therefore, a modelling methodology is required which faithfully models the behaviour of each component under any input scenario. This will allow errant behaviours to emerge which were not explicitly programmed. Fortunately, this is achievable for a relay based control system, as each relay may only be either powered or unpowered, relay contacts either open or closed, and the volumes of both are manageable. This requires a Petri net structure which can both evaluate Boolean gates to true when the appropriate conditions are met, and also re-evaluate them to false should the conditions no longer be met after time has elapsed.

A paper by Quezada et al (2014) presented a solution to this by developing a conversion method which enables modelling of control systems which require sustained input signals to continue an event or process. Quezada et al define Petri net transitions which can be used to model logical AND and OR gates using the same Petri net structures as Lee and Lee. These transitions include provision to re-evaluate the state of the gate should the initiating states change. Should the state of the places which enabled these special transitions change such that the transition would no longer be enabled, the effect of the transition firing is reversed by removing the tokens added and restoring any tokens removed. The use of this transition simplifies the presentation of the Petri net, though its complexity is beyond what would often be considered a regular Petri net transition. It may be better described as a Coloured Petri net transition which is used within an otherwise regular Petri net.

For use in this thesis, the Petri net logic gates presented by Quezada et al have been expanded. Whilst they are logically equivalent, the gates now no longer require a non-standard transition to re-evaulate the state of the logic gate. This has been done to minimise the number of Petri net extensions required to simulate the model, and help maintain general compatibility with Petri net software and understanding.

6.2.2.1 Boolean Form of Relay Circuits

The relay logic circuit B TPSR described in Chapter 4 will now be revisited to provide an example of the conversion of a relay circuit diagram to a Boolean form. The diagram has been reproduced in Figure 6-2, all four pathways through which the circuit may be completed have been overlain. The completion of any pathway will power the relay, and is dependent on the state of the relays on that pathway.

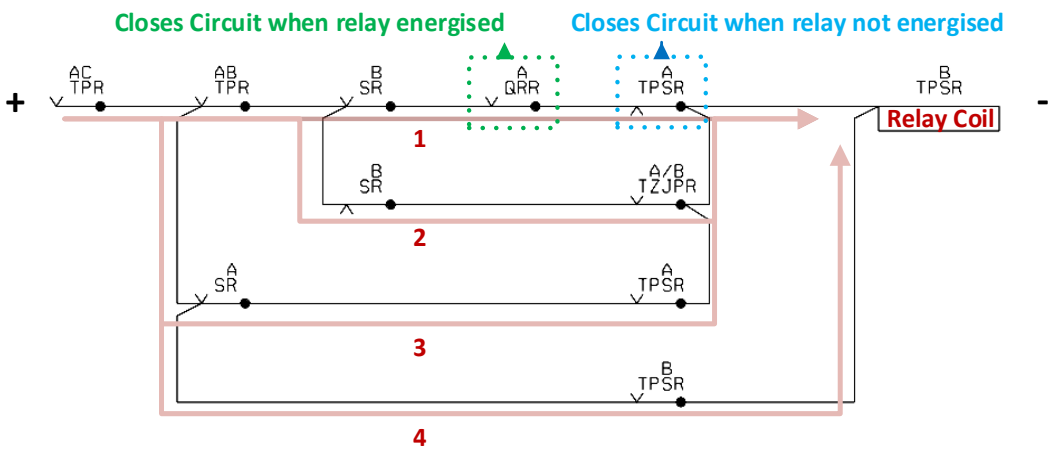

Figure 6-2 Example AHBC Control System Circuit (Westinghouse Signals, 1999).

Using this, the relay circuit can be redrawn as a Boolean logic diagram, shown in Figure 6-3. Here, each AND gate evaluates the state of the relay contacts comprising a single circuit pathway to determine if it is complete or open. The OR gate then evaluates if any of the circuits are complete, and therefore if the relay is powered or not.

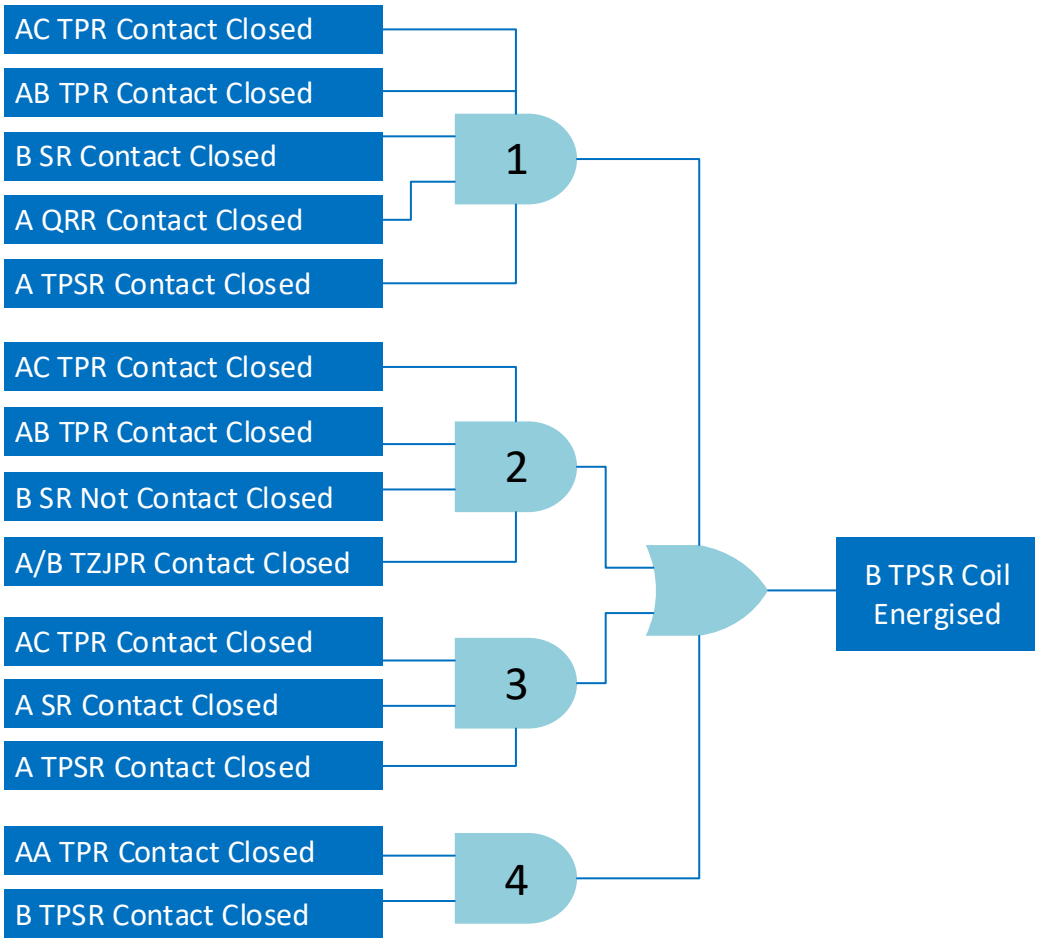

Figure 6-3 Boolean Logic Diagram for Example Circuit.

6.2.2.2 Petri Net AND Gate

A Boolean AND gate Petri net has been developed for this work which can be applied to all circuits. The general form is shown in Figure 6-4.

Both Places P-A1 and P-A2 can be used as outputs of the Boolean gate. A single token in P-A1 indicates the gate is outputting True, no tokens indicates false. This is reversed for place P-A2. This simplifies the structure of the later model, described later. Places P-A3 and P-A4 represent the Boolean inputs to the gate, where a single token in the place indicates the input condition is true. These represent the state of the relay contacts within a circuit, with true equating to a closed contact.

Should both conditions be true, transition T-A1 will be enabled by test arcs from both P-A3 and P-A4. When T-A1 fires it will remove a token from Place P-A2 and add a token to Place P-A1. This represents the AND gate evaluating to true, and the circuit completing. Transitions T-A2 and T-A3 control the evaluation of the Boolean condition from true to false. This achieved with inhibitor arcs. Should either condition no longer be true it will enable either T-A2 or T-A3, which removes the token from P-A1 and adds it to P-A2, breaking the circuit.

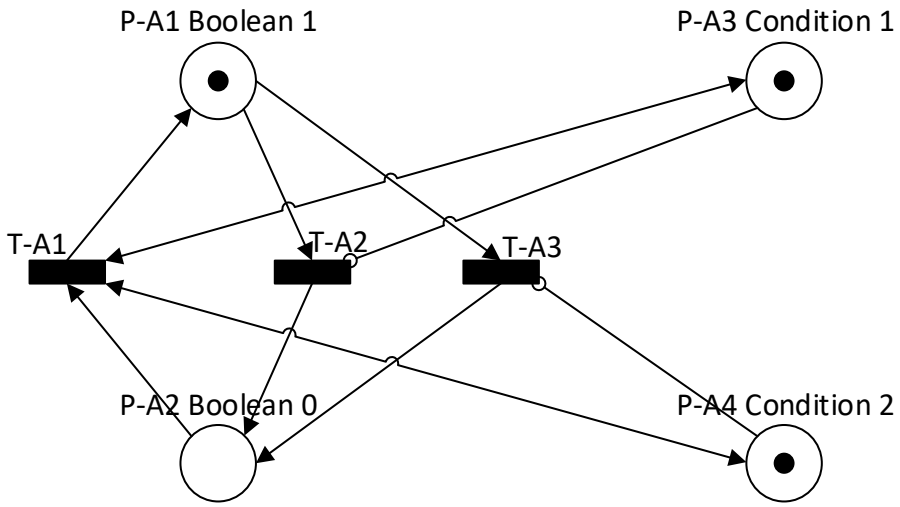

Figure 6-4 Petri Net Boolean AND Gate

The gate can be expanded to include additional conditions. This requires an additional transition per condition place added with a similar arc arrangement to transitions T-A2 and T-A3. The new condition places must be connected to the transition T-A1 via test arc. This general AND gate layout can be seen applied to circuit 1 in Figure 6-5. The state of the five relay contacts which comprise the circuit are represented by places P3-7. The lack of tokens in these places shows that none of the relay contacts are closed. The state of the circuit itself

is indicated by Places P1 and P2. The presence of a single token in P2 shows that the circuit is open.

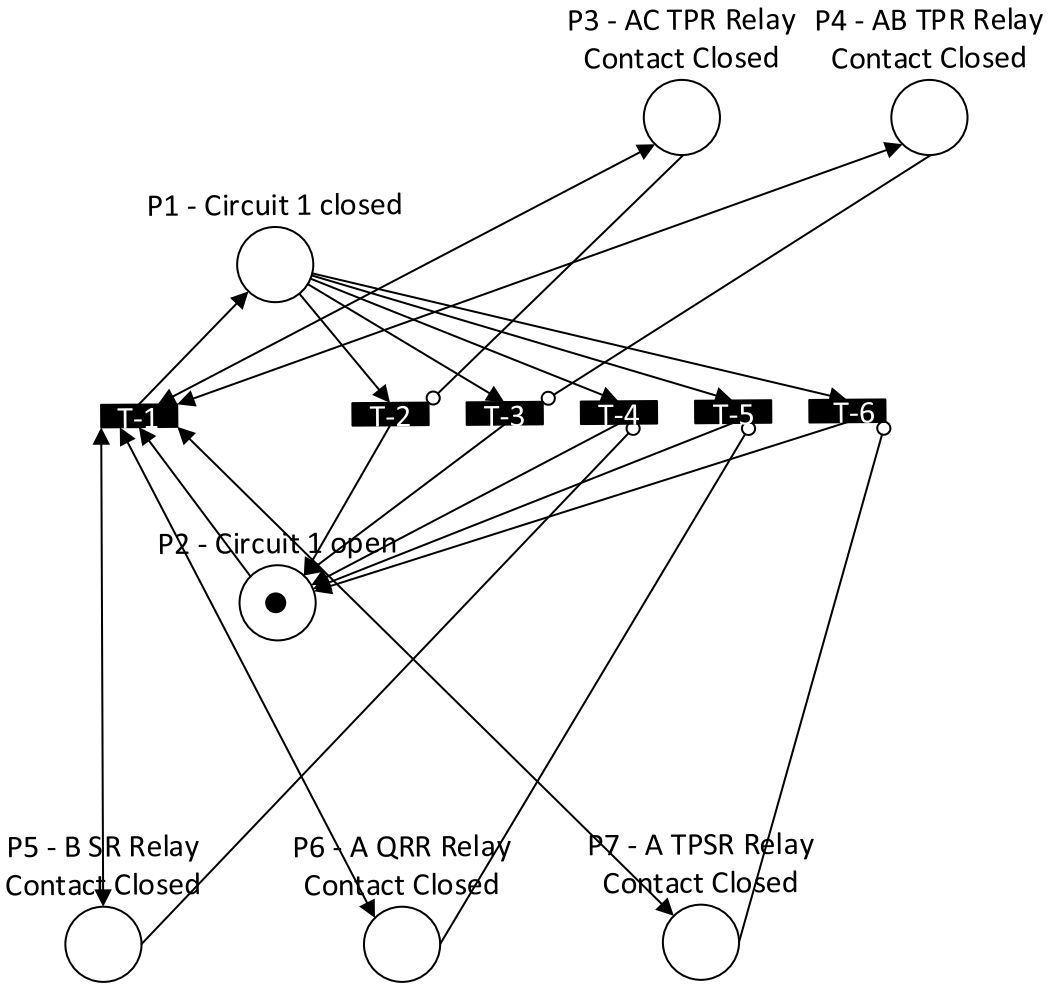

Figure 6-5 Petri Net AND Gate for Circuit 1 of B TPSR

6.2.2.3 Petri Net OR Gate

Boolean OR gates are used to determine if any of the multiple relay logic circuit pathways are complete. Figure 6-6 shows the Petri net structure for the OR gate, like the AND gate, this has also been developed as part of this thesis and was not found in prior literature. Places P-OR3 and P-OR4 are input conditions for the OR gate. Places P-OR1 and P-OR2 are the outputs of the gate, operating in a similar manner to the AND gate. Transitions T-OR1 and T-OR2 both evaluate the Boolean state to true when either condition is true. Transition T-OR3 evaluates the Boolean condition to false when neither conditions are true. Like the AND gate, this gate can be readily expanded to include additional conditions.

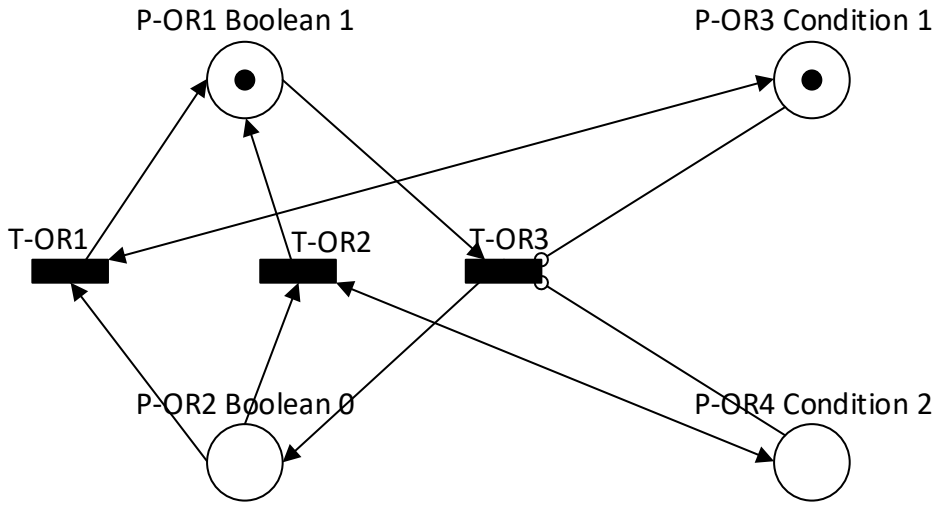

Figure 6-6 Petri Net Boolean OR Gate.

Applying this general form to B TPSR yields the net shown in Figure 6-7. This net determines if any of the 4 circuit pathways are complete, and energises the relay if so, de-energising it if not. The presence of a token in Place P1 indicates that the relay is energised, whereas a token in P2 indicates the relay is de-energised. The design of the net permits only one token between these two places. The output places of the AND gate Petri Nets which evaluate the state of individual circuits now provide the input to this gate. Places P3 to P6 indicate with the presence of a token if circuits 1 to 4 are complete. Any completed circuit will then enable a transition to move the token from P2 to P1.

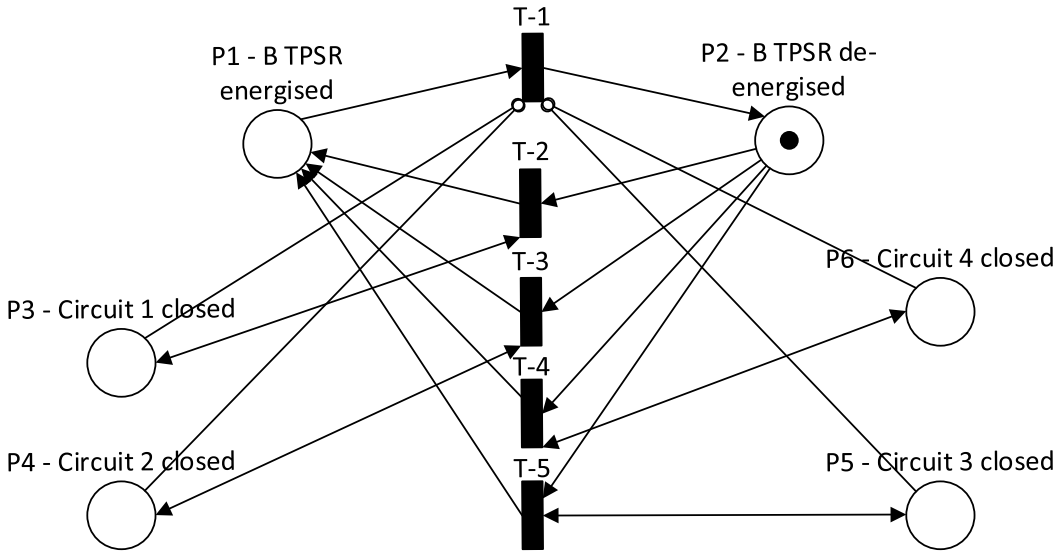

Figure 6-7 Petri Net OR Gate for Relay Logic Circuit B TPSR.

The entirety of the Petri net modelling the logic of circuit B TPSR can be seen in Figure 6-8. This Petri net uses the gates outlined above and the Boolean logic representation of the B

TPSR circuit. Places P11 to P20 represent the conditions of the relay contacts on which the B TPSR circuit's state depends on. Places P3 to P10 represent the state of the four possible circuit pathways through which the B TPSR relay can be powered. Places 1 and 2 represent the state of the B TPSR relay itself.

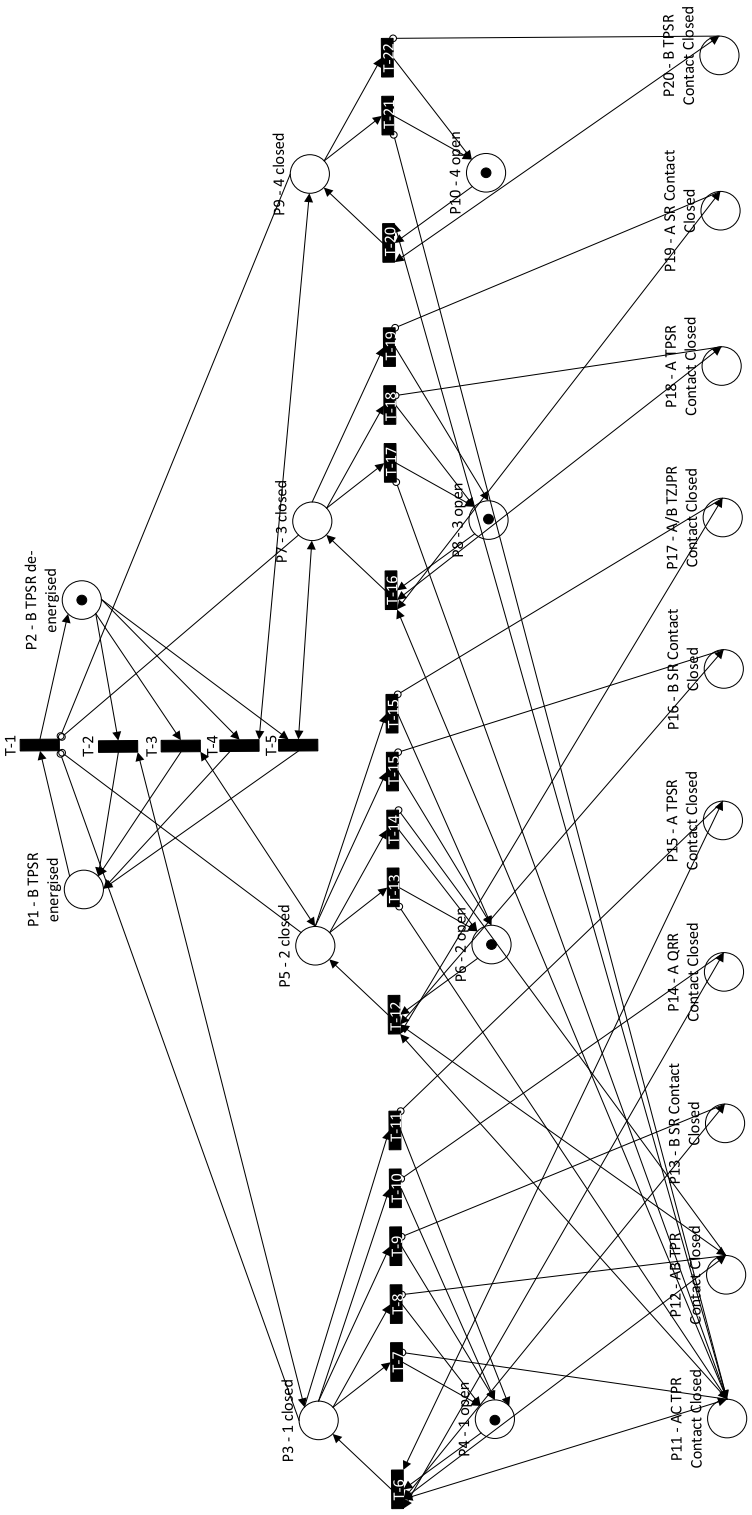

Figure 6-8 Petri Net Structure of B TPSR circuit

6.2.2.4 Relay Contact Movement and Failure Modes

So far, the Boolean diagrams presented have not considered failure modes or relay contact movement time. Direct current relays such as those used in the AHBCs control system can fail open circuit. This is where excessive wear to the relay contacts does not allow electrical current to flow. Relay contacts may also fail closed circuit, where excessive current causes the contacts to weld together. The Boolean form will now be expanded to include the possibility that a relay contact within a relay may not match the energisation state of that relay. This expansion is shown in Figure 6-9.

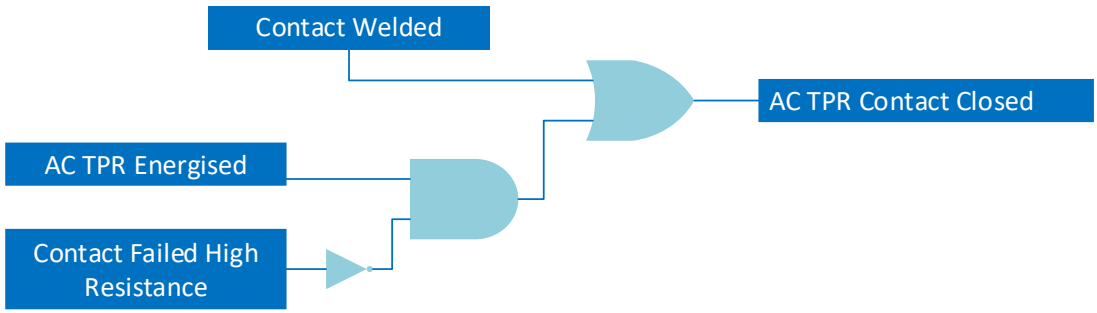

Figure 6-9 Relay State Boolean Diagram

Now the contact closed state reflects the state of the relay, and whether the contact has welded or failed in high resistance. The example shown is for a front-contact relay where energisation of the relay coils closes the contact. The structure is identical when applied to a back contact – one which closes when the relay is not energised. An example is shown in Figure 6-10, with the only difference being the addition of a NOT gate after the relay state, indicated by the triangle symbol.

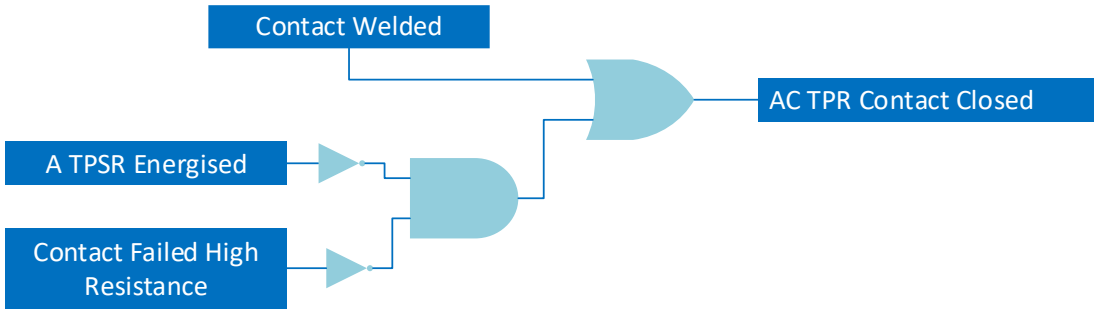

Figure 6-10 Relay State Boolean Diagram - Back Contact

A small Petri net module was created to simulate this, shown Figure 6-11. The number of tokens in Place P-CF1 controls its behaviour. Zero tokens allows the contact to open and close normally. When there are zero tokens in P-CF1, place P-CF3 will always be the same as the place P-CF2 which represents the state of the controlling relay coil. When there is a token in place P-CF2, transition T-CF1 is enabled via test arc, after firing a single token is placed in P-

CF3. When there are no tokens in P-CF2, transition T-CF2 will no longer be inhibited allowing it to remove the token from P-CF3 ensuring both places have the same number of tokens. All transitions have a 0.02second delay to model the physical movement time of the relay contacts when opening or closing a circuit.

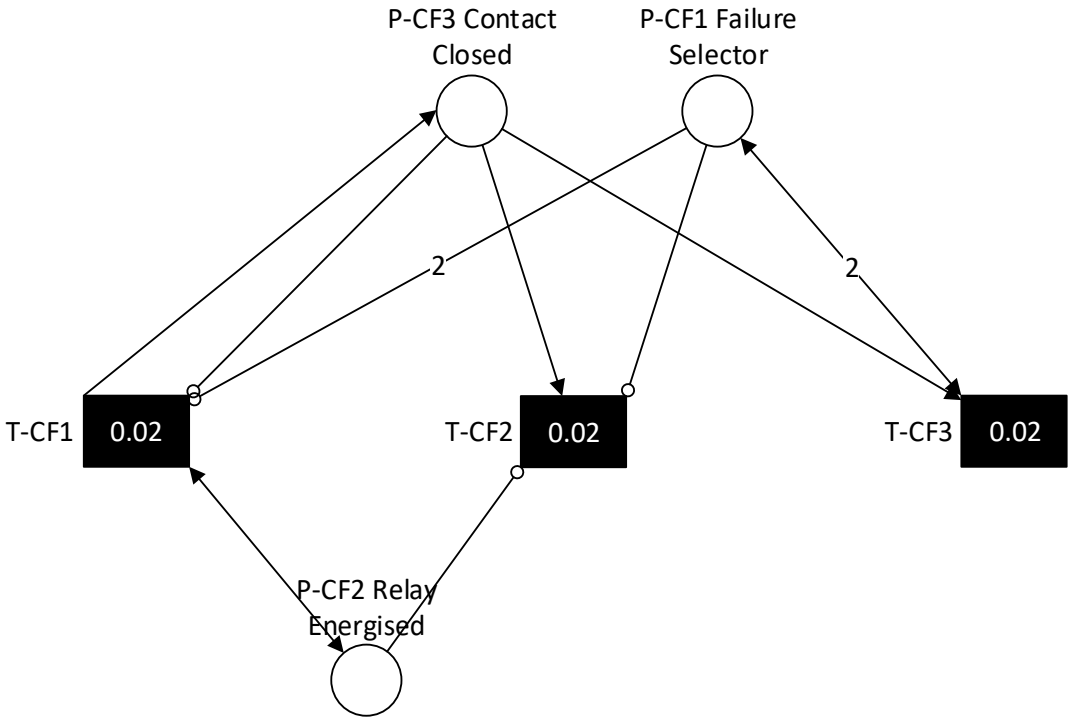

Figure 6-11 Petri Net Module for Front Relay Contact Failure.

The behaviour of this Petri net changes when there are tokens in place P-CF1. A single token in P-CF1 allows the net to model a closed circuit failure where a pair of relay contacts has failed by welding together. This single token inhibits transition T-CF2. Therefore, any token within place P-CF3 can no longer be removed. The placement of a single token in place P-CF1 is in effect an instruction for the contacts to weld together immediately if they are currently closed, or the next time they close. The model has been designed this way to prevent modelling the welding of contacts which never close in normal use.

Two tokens within place P-CF1 models an open circuit failure, where excessive wear prevents a pair of relay contacts from conducting electrical current. These two tokens inhibit both transitions T-CF1 and T-CF2. They enable a third transition T-CF3, this transition removes any tokens which may be present in place P-CF3, ensuring that the contacts remain open within the model.

This structure works for both front and back relay contacts. Recall that the OR gates which evaluate the energisation state of a relay are designed with two places outputting the state of the relay. Where the 'Relay Energised' place contains a token if energised, and no tokens

if not energised, and the ‘Relay De-energised’ place which behaves in the opposite manner. This allows a single Petri net to model both front and back contacts by selecting either the ‘Relay Energised’ or ‘Relay De-energised’ place as P-CF2.

If the relay contact modelled is a normal front contact, where powering the relay closes the contact, place P-CF2 should be the place representing the ‘Relay Energised’ place. If a back contact is to be modelled, where the contacts close if the relay is unpowered, place P-CF2 should be the place which represents the Relay De-energised’ place. Figure 6-12 shows the circuit B TPSR with these failure modules integrated. There is one instance of the relay contact failure module for each relay contact. Due to the scale of the AHBCs control system it is not feasible to reproduce all other relay logic circuits in their Petri net form in this thesis, with this serving as an example.

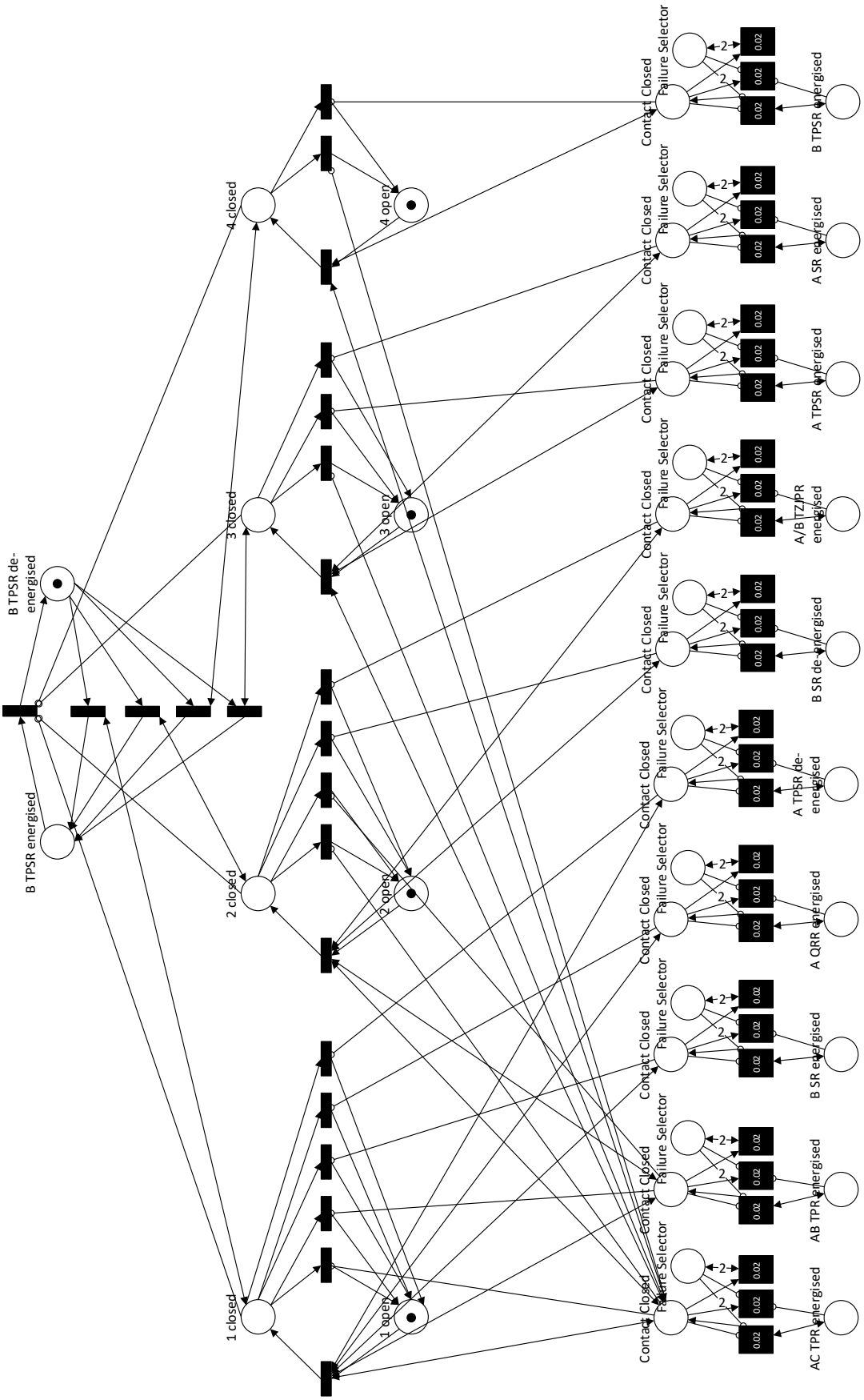

Figure 6-12 Complete Petri Net for Relay Logic Circuit B TPSR.

A further Petri net module was developed to model capacitor degradation. Some relays contain capacitors which hold the relay energised for a specific length of time after the power to the relay has been cut. Loss of electrolyte from a capacitor will progressively reduce its capacitance. Four levels of degradation have been used corresponding to 75%, 50%, 25%, and 0% of the original hold open time. These times were arbitrarily chosen. The Petri net for this module can be seen in Figure 6-13.

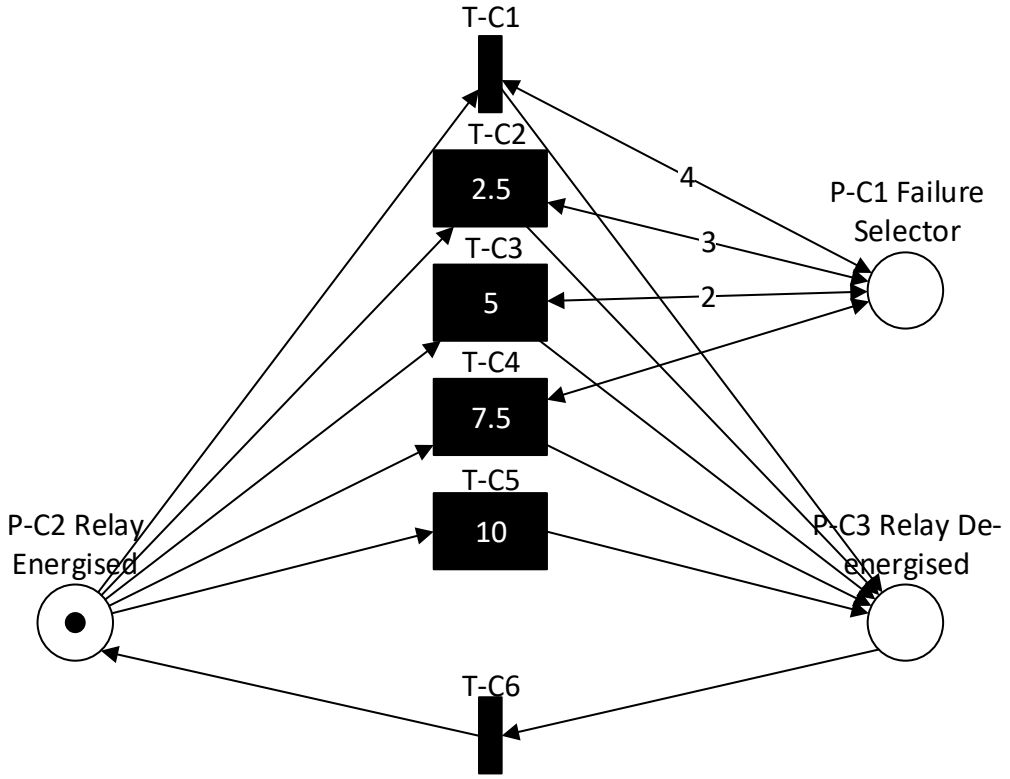

Figure 6-13 Petri Net Module for Capacitor Degradation and Failure.

This module replaces part of the OR gate which models the change of relay state. Using the Petri net in Figure 6-7 as an example, transitions T-C1 to T-C5 replace transition T-1. Each of these five transitions represents a different degradation state of the capacitor. T-C5 represents an undegraded capacitor, in this example with 10 seconds of hold energised time. T-C1 to T-C4 control the four levels of degraded state, from 75% to 0% capacitance. Place P-C1 enables these transitions via test arcs. Zero tokens within this place leaves only transition T-C5 enabled, modelling the full capacitor hold powered duration. When there are four tokens in P-C1, representing complete loss of capacitance, all transitions T-C1 to T-C5 are enabled. However the fastest transition T-C1 will fire first, leaving the other transitions to be unenabled.

6.2.3 Road Traffic Lights

This model incorporates two functions of the road traffic light system: the flasher which cycles the lights; and the failure of the individual LED bulbs and their cross-proving circuits. The flasher has three possible failure modes. It may cycle the lights either too fast or too slow, it may also fail to cycle the lights at all.

The Petri net structure which models both the operation and failure of the flasher is shown in Figure 6-14. The state of the flasher is represented by three places and a single token: if the token is in place P-FS1 the flasher is unpowered; if the token resides in P-FS2 the flasher output corresponding to the right hand side lights is powered; finally if the token is in P-FS3 then the flasher output corresponding to the left hand lights is powered.

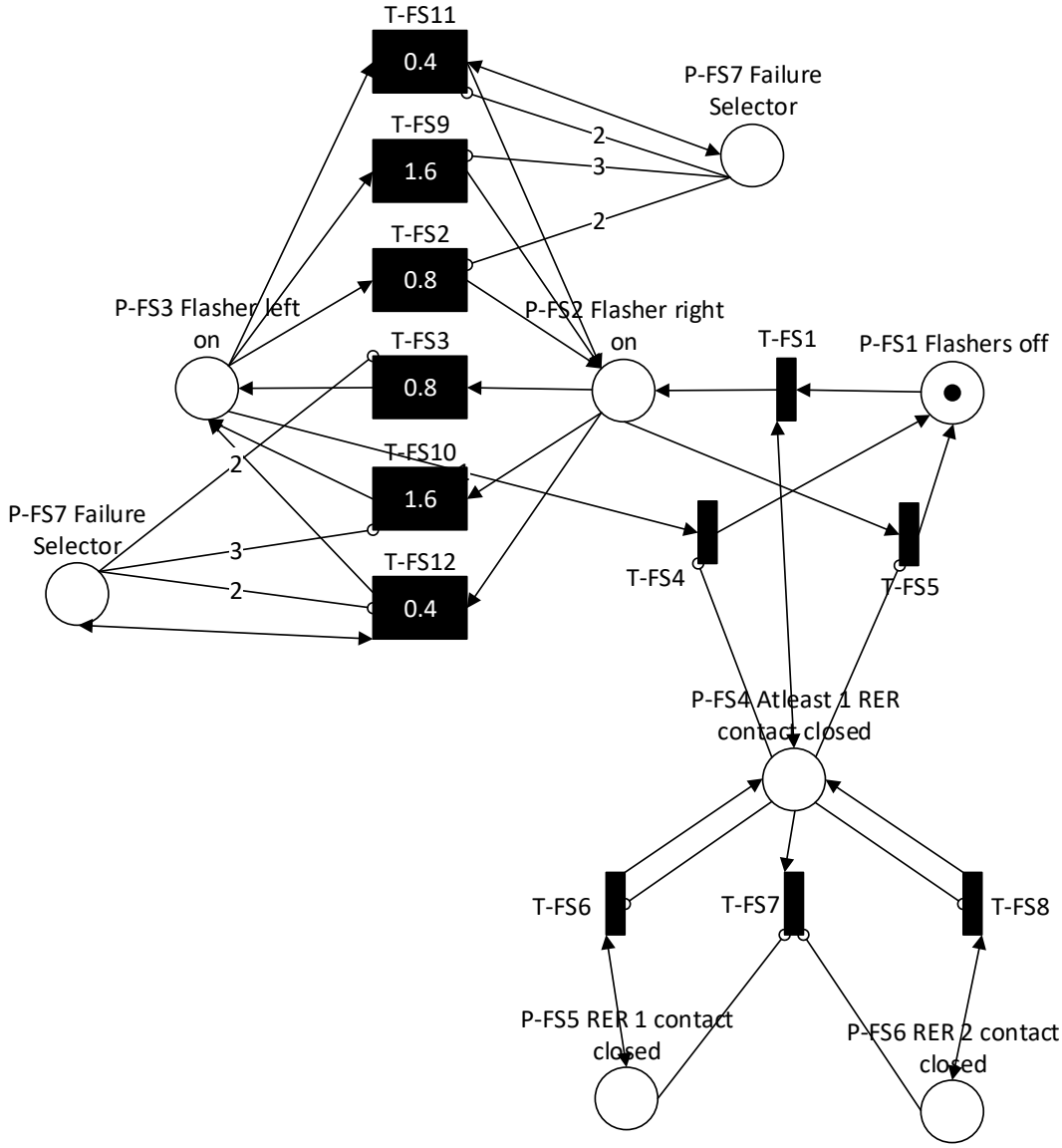

Figure 6-14 Flasher Petri Net Module.

The flasher is controlled by places P-FS5 and P-FS6, these represent the state of two pairs of relay contacts controlled by the RER relay. If one or more of these relay contacts is closed the flasher will power. Place P-FS4 indicates if at least one of the pair of contacts is closed. Transitions T-FS6 and T-FS8 place a single token in P-FS4 if either set of contacts is closed. Transition T-FS7 removes the token from P-FS4 if neither pairs of RER contacts are closed. Place P-FS4 controls initiation and cessation of the flasher cycle via transitions T-FS4, T-FS5 and T-FS1. Transition T-FS1 initiates the flasher when enabled by a single token in place P-FS4. Transitions T-FS4 and T-FS5 end the flasher cycling by removing the token from either P-FS2 or P-FS3 when not inhibited by a token in place P-FS4. Normal cycling of the flasher from right to left output is modelled using transitions T-FS2 and T-FS3, these transitions cycle the single token between left and right places every 0.8seconds.

Four additional transitions (T-FS9 to T-FS12) and many additional arcs model flasher failure, controlled by place P-FS7. Note, place P-FS7 appears twice in the figure to improve readability. When there are zero tokens in place P-FS7, transitions T-FS2 and T-FS3 which model normal 0.8 second cycling operation can be enabled in addition to transitions T-FS9 & T-FS10 which model slow 1.6 second flasher operation, however the faster 0.8second transitions will always fire first. When a single token is present in P-FS7, transitions T-FS11 and T-FS12 are enabled by test arc. These 0.4 second delay transitions fire before the 0.8 second transitions and model the fast cycling failure mode of the flasher. If two tokens are present in P-FS7 then the transitions responsible for normal and fast cycling of the flasher are inhibited. This leaves only the slower 1.6 second transitions enabled, modelling the slow cycling failure mode. If three tokens are present in P-FS7 then all transitions which cycle the flasher are inhibited. This models the failure mode where the flasher is unable to cycle. In this model this leaves the right hand side illuminated, though it is not clear from the technical drawings available if this is the correct side or if it would vary.

Each Road traffic light stalk contains three lights, two flashing red LED lights and a single amber LED light. The Petri net structure controlling LED bulbs is shown in Figure 6-15. This is a simple Petri net where two places and a single token indicate if the LED is illuminated. Place P-LED3 controls LED bulb failure. A single token in P-LED3 inhibits transition T-LED2 preventing the token moving P-LED1 to P-LED2. The additional place/arcs which control the transitions are not shown in this figure.

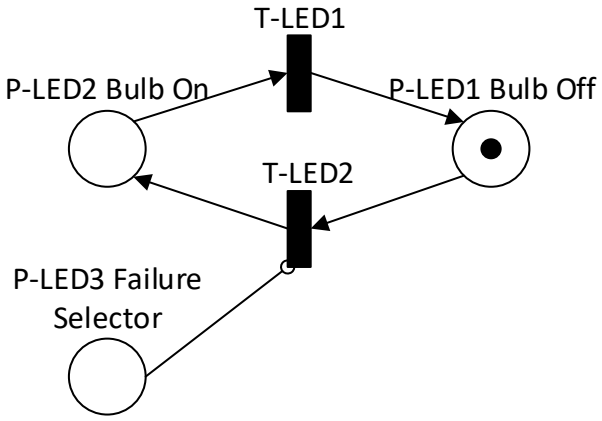

Figure 6-15 General LED Failure Petri Net.

The Petri net structure for the red LEDs is more complex as they are controlled by the flasher, and the presence of a proving relay wired in series with the red LED bulbs which can only be powered continually if neither red LED has failed open circuit. The Petri Net structure for this is shown in Figure 6-16. Places P-L1 to P-L4 represent the state of the 2 LED bulbs. Places P-FS3 and P-FS2 are from the flasher circuit Petri net presented previously. These use inhibitor and test arcs to control when the bulbs illuminate.

Collectively, places P-L7, P-L8 and transitions T-L5 to T-L7 form an OR gate. If at least one bulb is illuminated the relay will power, reporting to the control system that this light stalk has at least one working bulb.

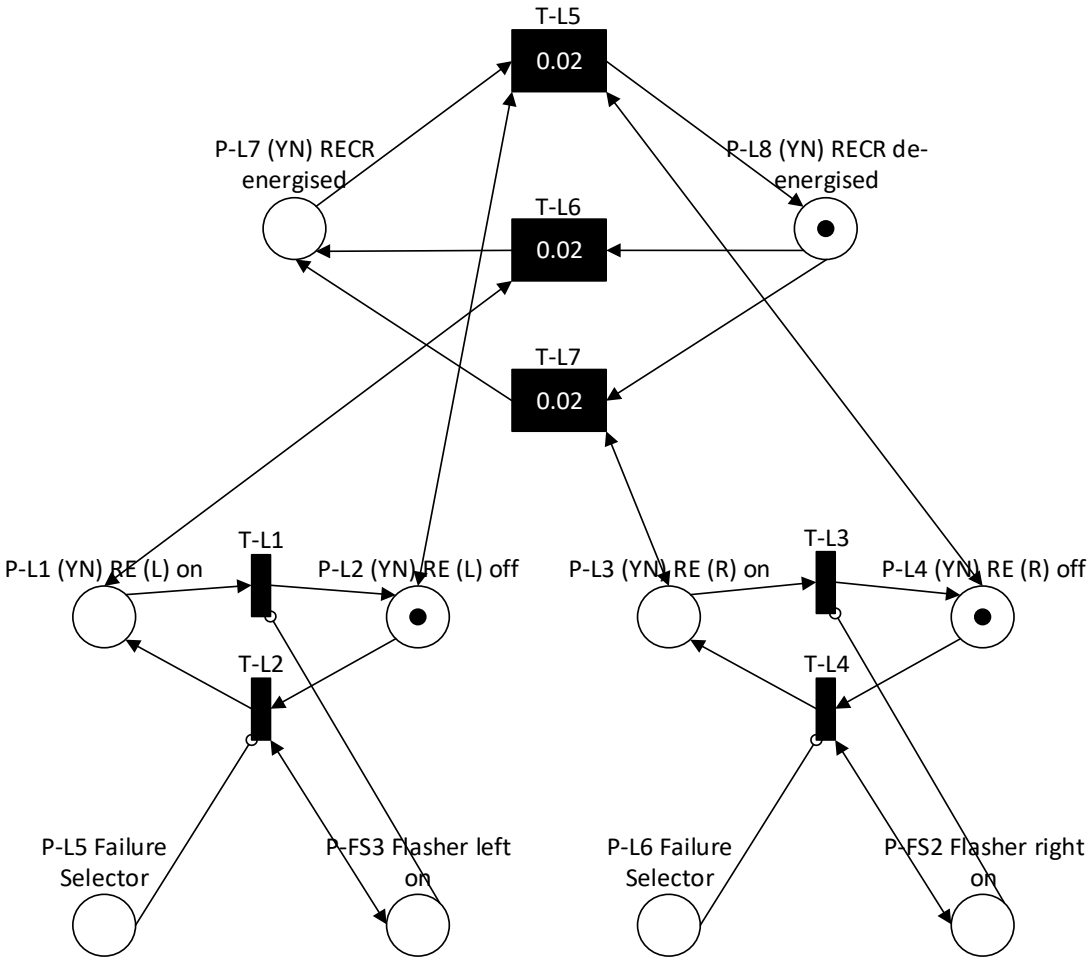

Figure 6-16 Petri Net Module for Red Road Traffic Lights.

6.2.4 Road Traffic Barriers

The first part of the barrier operation module is shown in Figure 6-17. This controls the raising and lowering of the barrier. Place P-B1 holds 83 tokens, each representing 1degrees of rotation from horizontal. Raising or lowering of the barrier is controlled by P-B2 and P-B3 respectively, representing both the state of the solenoid valve and hydraulic pump control circuit which powers the hydraulic system. These places are controlled by an AND gate comprising transitions T-B1 to T-B4 which are controlled by the places P-B4 to P-B6, which represent the state of the three relay contacts which must close to power both the solenoid valve and hydraulic pump to raise the barrier. The 0.4 second time delay of these transitions models the operation time of the solenoid valve. All three relay contacts must be closed for the barrier to rise.

Places P-B7 and P-B8 are used to control any failures which affect the operation of the barrier due to a failure within the barrier system. When there are zero tokens in place P-B7, transition T-B6 will raise the barrier at 20 degrees per second representing a normal rotational rate. An inhibitor arc from P-B1 to T-B6 prevents the barrier from rotating beyond

84 degrees. When there is one token in P-B7, transition T-B6 is inhibited. This models the scenario where the barrier is unable to raise due to failure within the barrier or other fault. The cause of the fault is not modelled, only the failure of the barrier to rise when called. When there are two tokens in place P-B7, transition T-B5 is enabled. This transition is slower than T-B6, representing a reduced rotational rate of 10 degrees per second, modelling the scenario where a barrier component failure has reduced the rate of rotation. Place P-B8 controls failures which affect the descent of a raised barrier. In a similar manner to P-B7: zero tokens models normal behaviour, one token models failure to descend when called; and two tokens models the effect of a reduced rate of descent.

The complete simulation model contains two copies of this structure, to model each barrier, the Y side is shown in the figure.

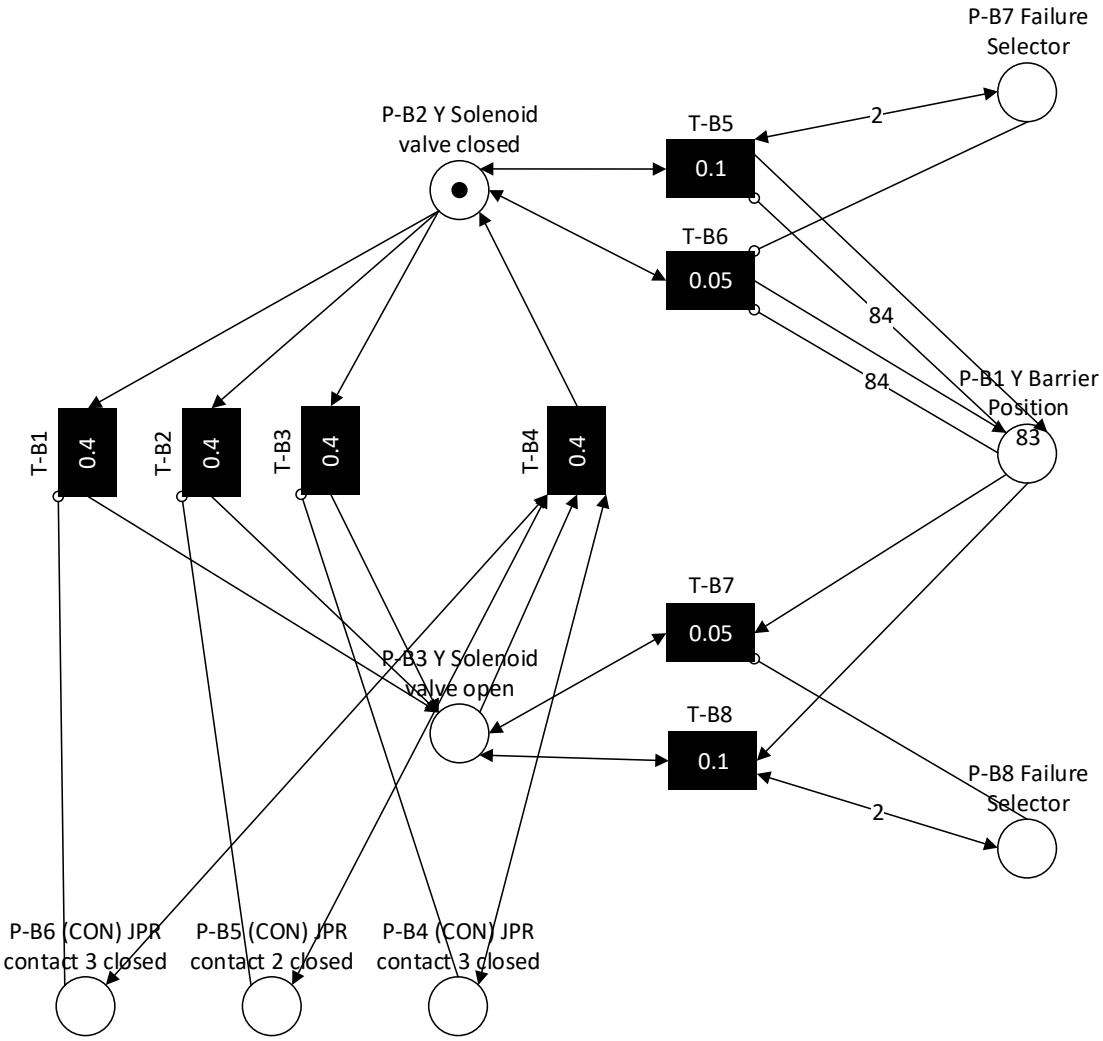

Figure 6-17 Y Barrier Movement Petri Net Module, Part 1.

Each barrier unit contains four sets of wiper contacts which complete at different ranges of angle of the barrier. Three of these contacts perform cross proving duties, the final cuts

power to the hydraulic pump when the angle of the barrier is greater than 83 degrees from horizontal. Each wiper contact is represented by two places and a single token, shown Figure 6-18. Two transitions per contact move the single token to indicate the state of the contact. These transitions are controlled via test and inhibitor arcs from place P-B1.

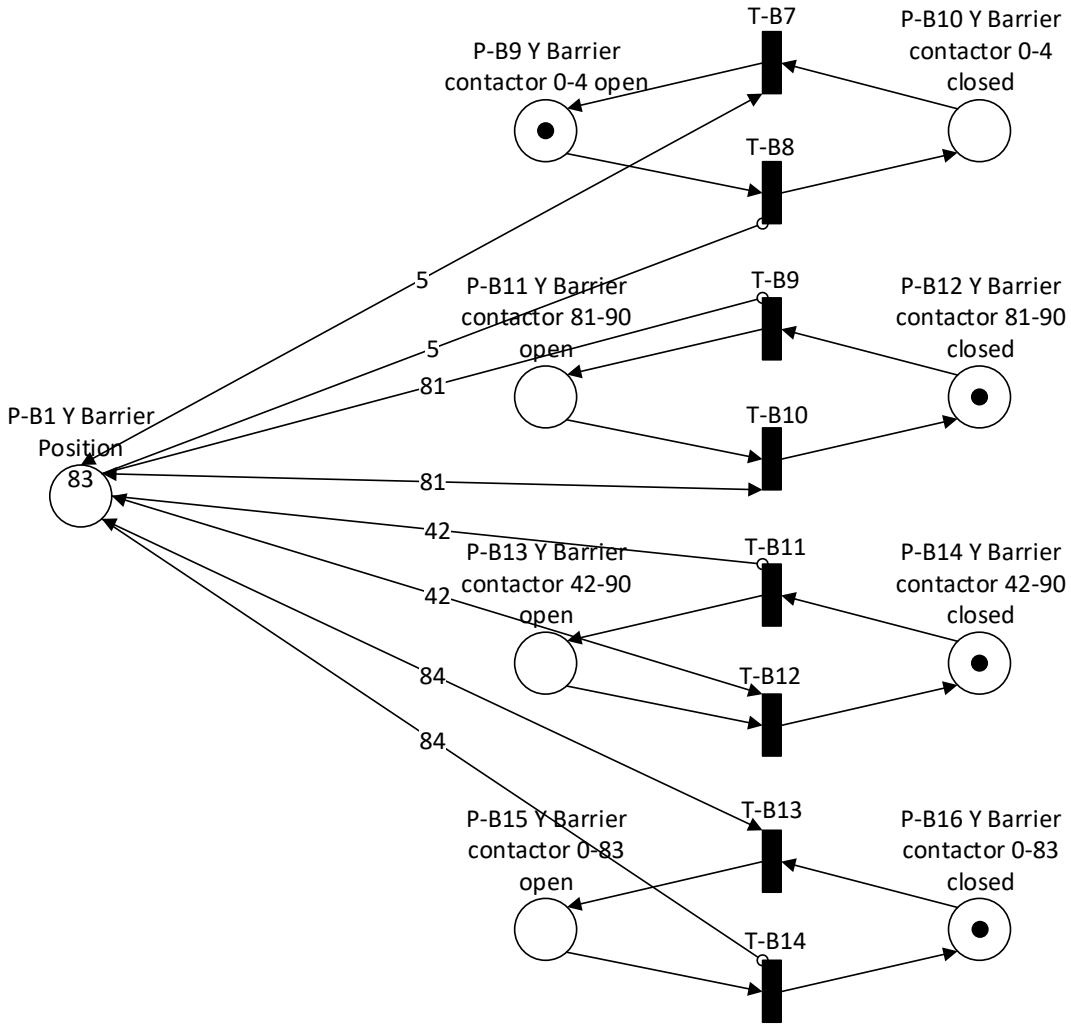

Figure 6-18 Y Barrier Wiper Contacts Petri Net Module, Part 2.

6.2.5 Structure Summary

For brevity, only the key elements of the simulation Petri net model have been presented here. In total, the simulation Petri net contains 717 places and 1064 transitions. The majority of these places and transitions are used in the relay logic module.

6.3 Simulation Model Structure Using Coloured Petri Net

This section presents the structure of a CPN model for the simulation of the AHBC crossing system under failure conditions. This is logically equivalent to the TPN model presented previously. CPNs offer much greater flexibility in model structure compared to regular Petri Nets. Due to this, it is important to outline what the CPN aims to achieve compared to the

regular Petri Net implementation of the model. The intention of this CPN model is to simplify the complex network of arcs which the regular Petri Net requires to represent the Boolean relay logic circuits. This may improve the readability and reproduction of the model.

The syntax to present the CPN uses a mixture of the CPN ML modelling language created by Jensen and C++. The CPN ML language clearly defines a syntax to refer to tokens and colour sets (Jensen & Kristensen, 2009). This has been adopted within the CPN models presented here. However, to speed up development of the model and improve readability, functions have been written using a style which will be familiar to users of C++. An explanation of the syntax adopted is provided as encountered when describing the model. A description of the CPN graphical and syntax conventions adopted is also available in chapter 2.

6.3.1 Train Simulation Module

By enabling the use of mathematical functions within a Petri net, a CPN enables the use of division and multiplication operations which are otherwise not possible using a regular Petri net. This has been exploited to improve the functionality of the train simulation module for the CPN implementation. The time taken to reach a track circuit, and pass that circuit is dependent on the train speed, length and track circuit length. In the regular Petri net model, these times must be calculated and input to the model. For the CPN model, a function calculates these times automatically based on variables contained within tokens describing the aforementioned details. Aside from this, the CPN module is logically identical to the TPN module.

The graphical portion of CPN train simulation module is presented in Figure 6-19. It uses three places and sixteen transitions. Transitions T1 and T16 control the arrival and exit of a train into an AHBC's track circuit system respectively. Transitions T2 to T15 simulate the train's passage through the seven track circuit events. The even transitions within this range control the beginning of each track circuit event, whilst the odd transitions control the completion of each track circuit event.

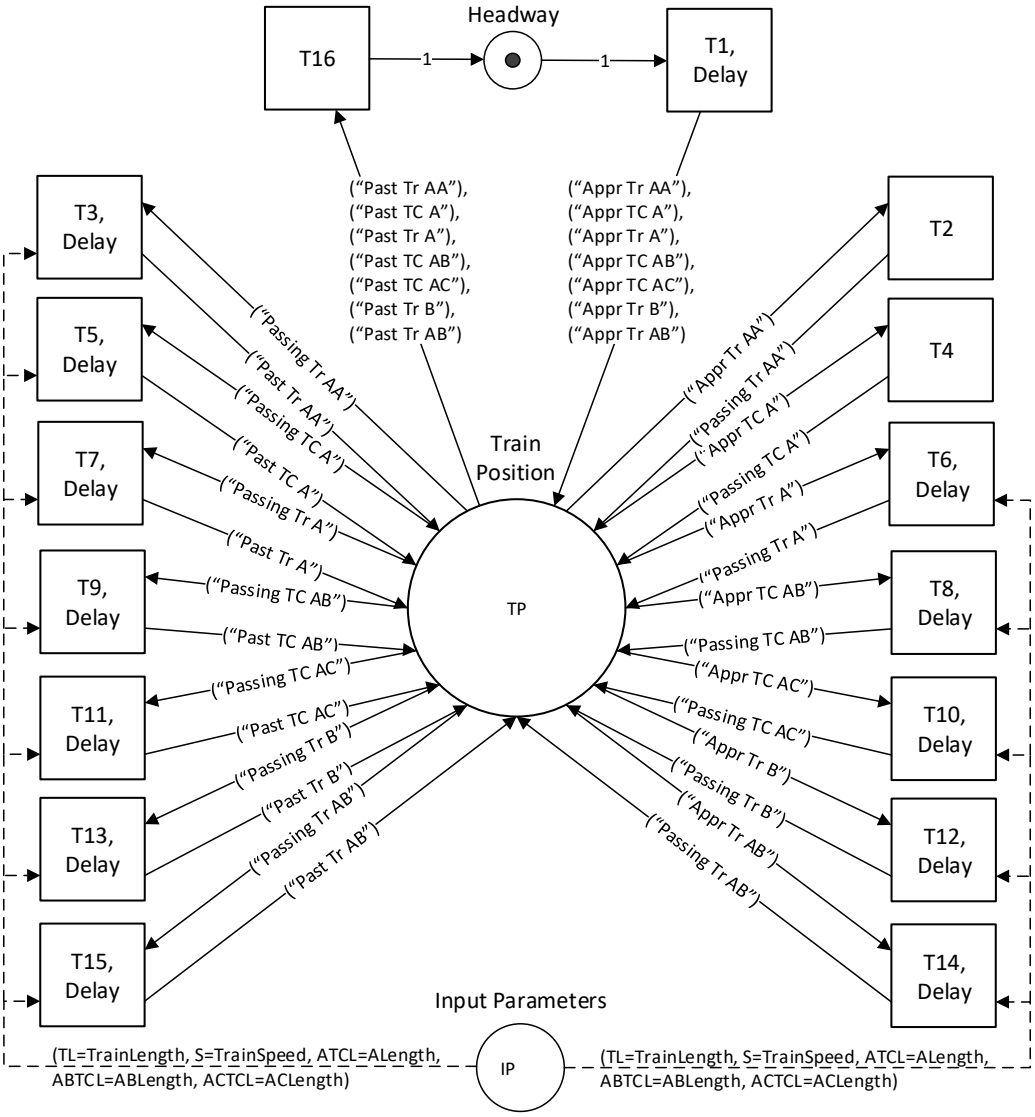

Figure 6-19 CPN Train Simulation Module

The place 'Headway' is used to initialise the module, it uses regular tokens. Place 'Train Position' uses Colour TP, which contains a single string. The string in each token describes the state of each of the seven track circuit events which occur when a train passes an AHBC level crossing. Each string contains a prefix indicating if the train is either: approaching the track circuit event, "Appr"; passing the track circuit event, "Passing"; or has completed the track circuit event, "passed". The suffix describes the track circuit event itself. Track circuit events are abbreviated to "TC" and treadle events "Tr". Therefore, a token containing the string "Passing Tr B" indicates the train is currently passing over treadle B.

The Colour set for the place 'Input Parameters' is used to store information of both the train and the track circuits. This Colour set is defined in Figure 6-20 line 190. It is formed by the union of five other colours shown lines 191-196. Colours 'TrainLength' and 'TrainSpeed' are

used to describe the length and speed of the train respectively. Colours 'ALength', 'ABLength' and 'ACLength' describes the lengths of the three track circuits which comprise a single line of a AHBCs track circuit system.

```
190. Colour InputParameters As TrainLength x TrainSpeed x Alength x ABLength x ACLength
191. Colour TrainLength As Real Integer Number
192. Colour TrainSpeed As Real Integer Number
193. Colour Alength As Real Integer Number
194. Colour ABlength As Real Integer Number
195. Colour ACLength As Real Integer Number
196. Colour TP As String
```

Figure 6-20 CPN Colour Set for Train Simulation Module.

As per the TPN train simulation module, the simulation begins with a 1,000 second delay representing the headway between trains. This is modelled using the transition labelled T1. After this delay the regular token in the place 'Headway' is removed and seven tokens, each containing a string is placed in the 'Train Position' place. These strings indicate the train is approaching all seven track circuit events.

One of the seven track circuit events within the CPN model is described below to illustrate how the model functions. The first track circuit event describes the train passing over the first treadle, AA. Immediately after T1 fires, placing a token within 'Train Position' which contains the string "Appr Tr AA", transition T2 is enabled. This transition fires without delay, removing the token and replacing it with the token "Passing Tr AA". There is no delay as this track circuit event occurs immediately upon arrival of the train. This transition is not strictly necessary, as transition T1 could simply place a token with the string "Passing Tr AA" in place 'Train Position' instead of the "Appr Tr AA" string. It is however included for consistency with the other track circuit events.

The token with string "Passing Tr AA" now enables transition T3. This transition simulates the train passing over the treadle AA. A formula within the transition calculates the time required for the train to pass over this treadle and uses this to delay transition firing. To simplify the graphical portion of the model, this formula is displayed in Figure 6-21 line 198, along with the delay formulas for the other transitions. This formula is declared as 'T3 Delay', the parenthesis immediately following indicates this function requires variables TL and S. These variables are passed to the transition via a conditional arc from the place 'Input Parameters'. Thus referring to Figure 6-19, it is seen that variable S is of Colour 'TrainSpeed' and 'TL' is of Colour 'TrainLength'. The curly brace contains the formula to calculate the delay length itself.

```
197. T1 Delay = 1000seconds
198. T3 Delay(TL, S) { Delay = (TL) / (S / 2.237) seconds}
199. T5 Delay(ATCL, TL, S) {(ATCL + TL) / (S / 2.237) seconds}
200. T6 Delay(ATCL, S) {(ATCL - 50) / (S / 2.237) seconds}
201. T7 Delay(TL, S) {(TL) / (S / 2.237) seconds}
202. T8 Delay(ATCL, S) {(ATCL) / (S / 2.237) seconds}
203. T9 Delay(ABTCL, TL, S) {(ABTCL + TL) / (S / 2.237) seconds}
204. T10 Delay(ATCL, ABTCL, S) {(ATCL + ABTCL) / (S / 2.237) seconds}
205. T11 Delay(ACTCL, TL, S) {(ACTCL + TL) / (S / 2.237 ) seconds}
206. T12 Delay(ATCL, ABTCL, S) {(ATCL + ABTCL + 50) / (S / 2.237) seconds}
207. T13 Delay(TL, S) {(TL) / (S / 2.237) seconds}
208. T14 Delay(ATCL, ABTCL, S) {(ATCL + ABTCL) / (S / 2.237) seconds}
209. T15 Delay(TL, S) {(TL) / (S / 2.237) seconds}
```

Figure 6-21 CPN Delay Formula for Train Simulation Module.

The formula divides the train length, in meters, by the train speed, in miles per hour (mph), a conversion factor of 2.237 is used to convert mph to Meters per second (m/s). These inconsistent units are used as it is convention in the UK to describe train length in terms of meters, but speed using mph. This calculates the time it takes for the train to pass over the AA treadle. The delay formulas for the other transitions are expressed in lines 199 to 209. These follow a similar format to the formula described.

After the delay has elapsed, transition T3 fires. The token which enabled T3, “Passing Tr AA” is removed from place ‘Train Position’ and replaced by a token containing the string “Past Tr AA”. This concludes the track circuit event involving treadle AA. The six other track circuit events are modelled in a similar manner. After all track circuit events have completed, all seven tokens with the ‘Train Position’ place will hold the prefix ‘Past’. This enables transition T16, which removes all the seven tokens and places a single regular token in the place ‘Headway’, allowing a subsequent train to be simulated.

The CPN implementation of the train simulation module is graphically simpler than the TPN implementation. It contains the same number of transitions, though reduces the number of arcs and places required considerably. The CPN implementation requires only three places whereas the TPN required fifteen. It is important for the CPN implementation to hold all the track circuit event states within one place, as this enables simplification of the general form of the control system relay logic circuits described next.

This CPN module could be further graphically simplified. Transitions T2 to T15 could be condensed into two transitions, which would be enabled concurrently for each of the seven tokens. Arc expressions would map each token input to the transition to the appropriate output token and delay time. It is common in CPN modelling to have to decide how much of the model should be presented graphically, and how much should be abstracted away within

functions and complex Colour sets (Jensen, 1997). In this instance, comprehension is best served with more transitions, rather than less.

6.3.2 Relay Based Control System

The CPN implementation of the relay based control system is logically identical to that of the TPN implementation, offering no additional functionality. The aim of the CPN implementation was primarily to reduce graphical complexity compared to the TPN in order to improve model readability and the ease with which the model can be altered. Figure 6-22 shows the graphical portion of this module.

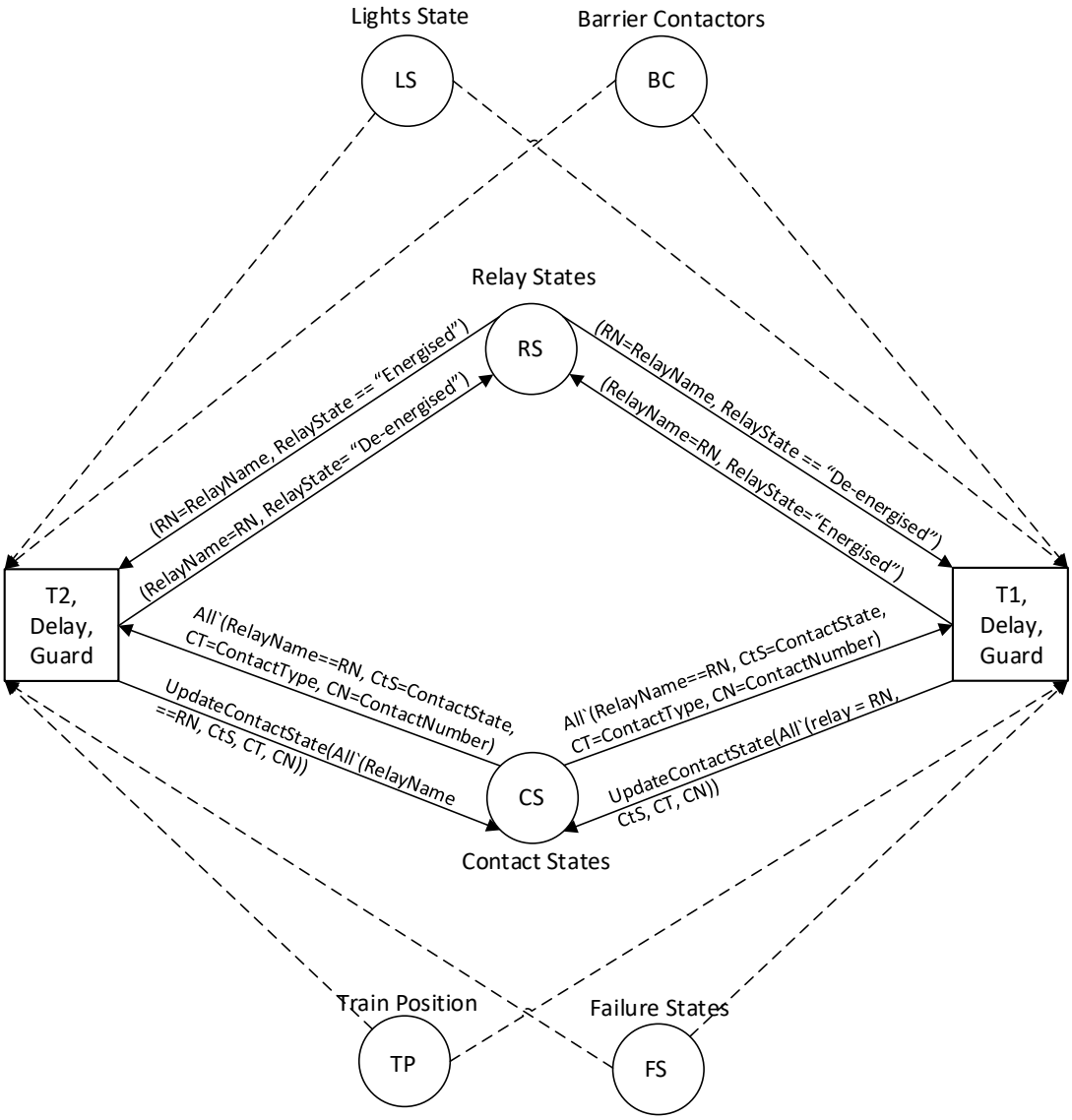

Figure 6-22 General form of CPN Relay Logic Petri Net

This CPN uses five places and two transitions to model the entirety of the relay based control system. Transition T1 energises relays and changes the state of relay contacts accordingly.

Transition T2 performs the reverse of this function, de-energising relays and changing contact states. Both T1 and T2 can be enabled concurrently in different transition modes enabling them to control the state of all relays.

6.3.2.1 Colour Sets and Places

Figure 6-23 details the colour sets for each places in this net. The place 'Relay States' shows what state each relay is in, using the colourset RS, defined in line 210. This colourset is the union of two colours, RelayName and RelayState, both of which are of type string. These are defined on lines 211 and 212. The Colour RelayName holds the name the relay, and the Colour RelayState holds a string which reads wither "Energised" or "De-energised" indicating the state of the relay.

Place 'Contact States' shows the state of specific pairs of relay contacts, using the colour set CS, defined line 213. This is more complex than RS, comprising the union of four single colours. The first Colour 'RelayName' is the same as used in colour set RS. It describes which relay a given contact is of a component of. Colour 'ContactState' describes whether the relay contact is currently open or closed. When a relay changes energisation state and contact state is updated, Colour 'ContactType' is used to determine how the contact's state is affected. It can hold the string "Front Contact" meaning relay energisation closes the contact, or "Back Contact" meaning relay energisation opens the contact. The last Colour in set CS is 'ContactNumber'. This is of type Integer, and is used to identify the location of a pair of contacts within the control systems circuits and thus determine is effect on the wider system.

```
210. Colour RS As RelayName x RelayState
211. Colour RelayName As String
212. Colour RelayState As String
213. Colour CS As RelayName x ContactState x ContactType x ContactNumber
214. Colour ContactState As String
215. Colour ContactType As String
216. Colour ContactNumber As Integer
217. Colour FS as FailedComponentName x FailureMode
218. Colour FailedComponentName As String
219. Colour FailureMode As String
220. Colour LS As Stalk x Position x IlluminationState
221. Colour Stalk As String
222. Colour Position As String
223. Colour IlluminationState As String
224. Colour BC As Side x LowerAngle x UpperAngle x ContactorState
225. Colour Side As String
226. Colour LowerAngle As Integer
227. Colour UpperAngle As Integer
228. Colour ContactorState As String
```

Figure 6-23 CPN Control System Colours.

Place ‘Failure States’ holds tokens detailing which components have failed and in which failure modes. The numbers and values of these tokens are set prior to simulation, depending on which failures are to be simulated. One initialised, the tokens in this place remain static. They may be tested by transitions whose behaviour is affected by component failures, but they are never modified. The Colour set for this place is FS, formed by the union of Colours ‘FailedComponentName’ and ‘FailureMode’. Failure of relay contacts and other components will affect the energisation state of the relays within the control system, therefore conditional arcs have been drawn between this place and both transitions controlling relay state.

Places ‘Lights State’ and ‘Barrier Contactors’ hold tokens detailing the state of each bulb and barrier contactor respectively. These places are explained in greater detail later. Some relay circuits are wired into bulb and barrier contactor circuits for cross-proving. These conditional arcs connect these places to both transitions such that the tokens within these places can be tested. The place ‘Train Position’ was described in the previous section. Similarly, conditional arcs allow both transitions to test tokens within this place. This is specifically to allow relay logic circuits which include track circuits and treadles to be modelled.

6.3.2.2 Initialising Tokens

The ‘Relay States’ place is initialised with a token for every relay modelled. The ‘Contact States’ place is initialised with a token for every relay contact modelled. A sample of these initialising tokens, for the example of relay B TPSR is shown in Figure 6-24. This is the same relay which was presented as an example for TPN.

```
229. Relay States M0 [(“B TPSR”, ”Energised”) + (“OtherRelay”, “State”)]
230. Contact StatesM0 [
231. (“B TPSR”, “Back Contact”, “Open”, 1),(“B TPSR”, “Front Contact”, “Closed”, 2),
232. (“B TPSR”, “Front Contact”, “Closed”, 3),(“B TPSR”, “Back Contact”, “Open”, 4),
233. (“B TPSR”, “Front Contact”, “Closed”, 5),(“B TPSR”, “Front Contact”, “Closed”, 6),
234. (“B TPSR”, “Back Contact”, “Open”, 7),(“B TPSR”, “Back Contact”, “Open”, 8),
235. (“B TPSR”, “Front Contact”, “Closed”, 9),(“B TPSR”, “Front Contact”, “Closed”, 10)]
236. Failure States M0 [(“B TPSR1”, “HR”)]
```

Figure 6-24 Extract of Initial Tokens for Relay Module.

6.3.2.3 Transition T1

The two transitions which simulate relays and their contacts changing state are the most complex transitions presented in this model, and thus are afforded great care in their explanation. This begins with the arcs directed at T1. For clarity, Figure 6-25 shows these arcs alone.

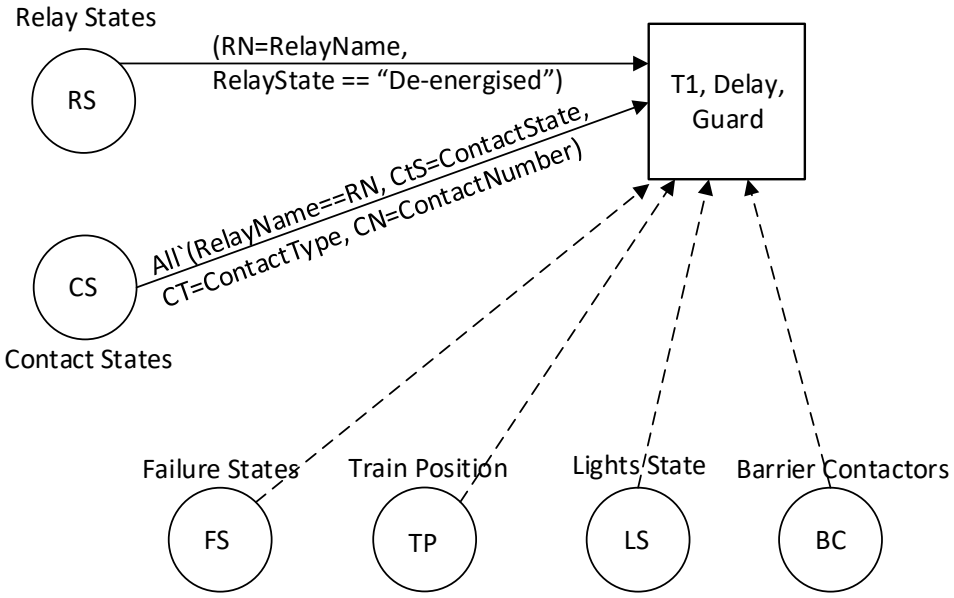

Figure 6-25 Relay Module Extract: T1 Incoming Arcs

An arc connects place 'Relay States' to T1. The variable RN is passed to the transition allowing guard functions to access the Colour 'RelayName'. As T1's purpose is to energise relays, only tokens whose Colour 'RelayState' == "De-energised" may enable T1. A further arc connects place 'Contact States' to T1. This arc specifies that all tokens which share the same relay name as the token from the 'Relay States' place are required to enable the transition in a single mode. The transition may be enabled in as many modes as there are tokens in the 'Relay State' place. T1 updates both the energisation state and the contacts within the relay when it fires. The arc also specifies variable names for each Colour which comprises the CS Colour set allowing them to be accessed in a later function.

Next the functions within T1 must be evaluated for each transition mode presented by the incoming arcs. The firing delay for T1 is always 0.02 seconds, representing the movement time for the relay contacts when the relay is energised. The Guard function for T1 is shown Figure 6-26 lines 238-249. The guard function is given access to all tokens within places: 'Lights State'; 'Barrier Contactors'; 'Train Position'; and 'Contact States' for evaluation of the function. It fulfils a similar purpose to that of an inhibitor arc. If for a given set of enabling tokens, it evaluates to false, the transition will not fire. For the purpose of example, the B TPSR relay is considered. Line 240 evaluates if the relay name passed to the function via variable RN is B TPSR. If true, line 242 calls the function BTPSRBooleanTree, declared over lines 250-265. This function evaluates the Boolean tree presented in Figure 4-17 to determine if the relay circuit is complete, if so the function returns true and the transition

may fire in this mode. The guard function contains If statements and Boolean tree functions for all other relays, however for brevity they are omitted.

```
T1 Delay = 0.02seconds
T1 Guard(LS, BC, CS, TP, RN) {
        //example relay
        If (RN == "B TPSR") {
                //Is relay powered?
                If (BTPSRBooleanTree(CS) == True) {Transition = Enabled}
        }
        //other relays not described for brevity
        ElseIf (RN == "OtherRelay") {
                If (OtherRelayBooleanTree(CS, TP, LS, BC) == True) {Transition = Enabled}
        }
        Else {Transition = Disabled}
}
BTPSRBooleanTree(CS) {
        //is circuit path 1 complete?
        If (AllPresent("AC TPR", "Closed", "1"), ("AB TPR", "Closed", "1"), ("B SR", "Closed", "1"),("A QRR", "Closed", "1"), ("A TPSR", "Closed", "1"))
        { Return True }
        //is circuit path 2 complete?
        ElseIf (AllPresent("AC TPR", "Closed", "1"), ("AB TPR", "Closed", "1"), ("B SR", "Closed", "1"), ("A/B TZJPR", "Closed", "1"))
        { Return True }
        //is circuit path 3 complete?
        ElseIf (AllPresent("AC TPR" , "Closed", "1"), ("A SR ", "Closed", "1"), ("A TPSR ", "Closed", "1"))
        { Return True }
        //is circuit path 4 complete?
        ElseIf (AllPresent("AA TPR" , "Closed", "1"), ("B TPSR" , "Closed", "1"))
        { Return True }
        //Are no circuit paths complete?
        Else { Return False }
}
```

Figure 6-26 Relay Module Extract: T1 Delay and Guard Function.

Finally the outgoing arcs from transition T1 are considered, shown Figure 6-27. The arc from T1 to 'Relay States' adds a token to the place with the Colour 'RelayState' = "Energised" with the same relay name as the token which was removed from this place. Thus, when T1 fires it effectively changes the state of the token which enabled it from de-energised to energised.

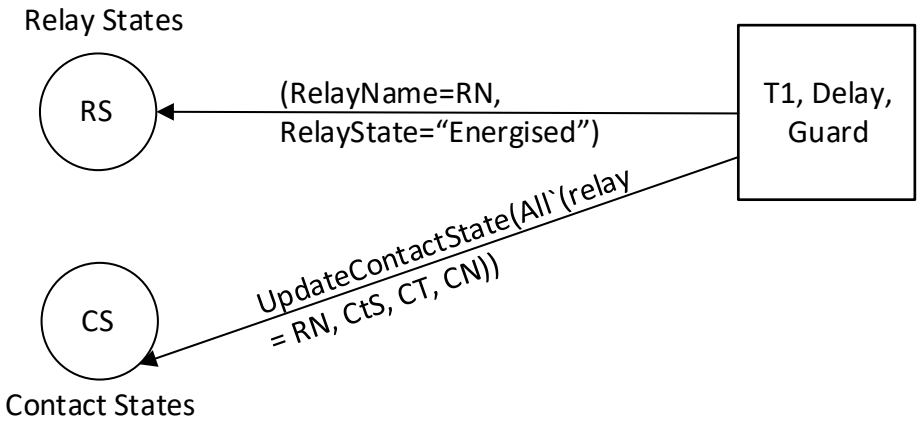

Figure 6-27 Relay Module Extract: T1 Outgoing Arcs.

The arc from T1 to 'Contact States' is more complex as it calls a function. The function it calls is shown in Figure 6-28. Recall that the incoming arc to transition T1 required all the tokens whose Colour 'RelayName' equalled the variable RN to enable. Variables for each of these tokens are now passed to the UpdateContactState function. The purpose of this function is re-evaluate the state of the relay contacts now that the relay has changed energisation state. This means that when T1 fires, the number of tokens in place 'Contact States' remains constant, only the Colour 'ContactStates' may change.

```
266. UpdateContactState(All(RelayState == RN, CtS, CT, CN)) {
267.             //Set opposite state to current state
268.             Variable NewState As String
269.             If (State == “Energised”) {NewState = “De-energised”}
270.             If (State == “De-energised”) {NewState = “Energised”}
271.             //Loop through each contact in relay
272.             For Each RelayState Token(RN, CtS, CT, CN) {
273.                         //create component name variable
274.                         Variable FailedCompName As String = Append(RN + CN)
275.                         //check for high resistance failure mode
276.                         If (Present((FailedCompName,  “High Resistance”))) {
277.                                     Output 1'(relay, CtS = “Open”, CT, CN)
278.                         }
279.                         //check for welded failure mode
280.                         Else If (Present((FailedCompName,  “Welded”))) {
281.                                     Output 1'(relay, CtS = “Closed”, CT, CN)
282.                         }
283.                         //if operating normally
284.                         Else {
285.                                     //contact opening
286.                                     If ((NewState == “Energised” AND CT == “Back Contact”)
287.                                     Or
288.                                     (NewState == “De-energised” AND CT == “Front Contact”))
289.                                     {
290.                                                 Output 1'(RN, CtS = “Open”, CT, CN)
291.                                     }
292.                                     //contact closing
293.                                     Else { Output 1'(RN, CtS = “Closed”, CT, CN) }
294.                         }
295.             }
296. }
```

Figure 6-28 Relay Module Extract: UpdateContactState Function.

The function is designed to be used whether the relay has just energised or de-energised. To accommodate this, the function takes the ‘State’ variable passed to T1 by the incoming arc from place ‘Relay State’ and sets the opposite state to the variable NewState. This occurs over lines 268-270. This is because the ‘State’ variable represents the prior state of the relay, and the function must act on the posterior state of the relay, which is always the opposite of the prior.

Next the function iterates over each of the tokens from place ‘Contact States’ which enabled it. On line 274, it appends variables RN and CN to form the new variable ‘FailedCompName’. Through lines 276-282 it tests the place ‘Failure States’ to see if this specific relay contact has failed. The append function is required as the number of Colours needed to identify a single component element varies across the different types of component. Appending multiple strings to create a unique component name string represents an easy way to resolve this Colour set compatibility when testing the ‘Failure States’ place, Rather than use multiple different colour sets within the ‘Failure States’ place.

On line 276 the place ‘Failure States’ is tested to determine if it contains a token with a Colour matching the appended component name and Colour ‘FailureMode’ == “High Resistance”. If found, this indicates that relay contact cannot close and thus the simulation should not allow this. If this token is present in place ‘Failure States’ the function adds a token to place ‘Contact States’ identical to the token which was removed, except Colour ‘ContactState’ is now set to “Open”. As the function uses the If, ElseIf, Else style of control statements the for loop will now iterate over the next token representing the next relay contact pair within relay B TPSR and determine its new state.

Should the relay contact have not been specified as failed high resistance, the function will next test if the relay contacts have welded. This is performed in a similar manner to the high resistance failure mode test, except now Colour ‘FailureMode’ = “Welded”. If this is the case the function will return the token to place ‘Contact State’ with Colour ‘ContactState’ set to “Closed”.

Should neither failure mode be found for the relay contact its state is evaluated based on what type of relay contact it is and the new state energisation state of the relay. This takes place over lines 286-293. If the new state of the relay is energised, and the contact is a Back Contact type, or if the new state is de-energised and the contact is a front contact, the function outputs a token where Colour ‘ContactState’ = “Open”. Otherwise the function outputs a token where Colour ‘ContactState’ = “Closed” without further if statements as this is the only possible state having excluded the previous two.

6.3.2.4 Transition T2

Transition T2’s purpose is to de-energise any relay which is no longer powered and update its contact states accordingly. It achieves this in a very similar way to T1. An explanation of the function of T2 begins with its input arcs, shown Figure 6-29. This is near identical to the that of T1 which one exception, Colour ‘RelayState’ of the token in place ‘Relay States’ must now equal “Energised”.

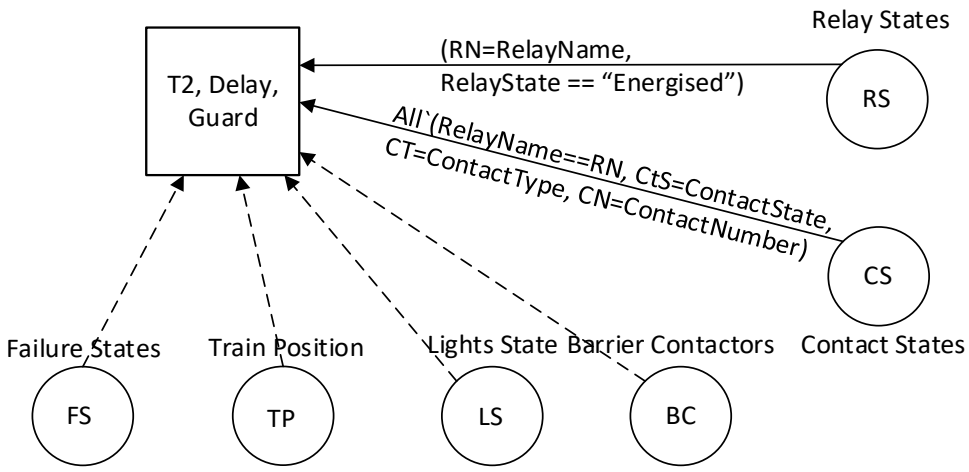

Figure 6-29 Relay Module Extract: T2 Input Arcs.

The guard function for T2 is partially shown in Figure 6-30. This is also near identical to that of T1. However now the transition is enabled if the Relay Boolean Tree function evaluates to false, indicating that the relay is no longer powered and therefore must de-energise.

```
297. T2 Guard(LS, BC, CS, TP, RN) {
298.            //example relay B TPSR
299.            If (RN == "B TPSR") {
300.                        //is relay powered?
301.                        If (BTPSRBooleanTree(CS) == False) {Transition = Enabled}
302.            }
303.            //other relays not shown for brevity
304.            ElseIf (RN == "OtherRelay") {
305.                        If (OtherRelayBooleanTree(LS, BC, CS, TP)  == False) {Transition = Enabled}
306.            }
307.            Else {Transition = Disabled}
308. }
```

Figure 6-30 Relay Module Extract: T2 Guard Function.

The delay function for T2 is substantially different to T1. This is because some relays use capacitors to hold their electromagnets powered for several seconds after power has been cut. Due to this, the delay function of T2 depends on the relay, and the state of decay of the capacitor. Figure 6-31 contains part of this function, for brevity only one relay has been considered, '(CON) JR', though the function acts on all relays. The function compares the relay name to all those which feature a capacitor. If a matching relay name is found the function then tests place 'Failure States' to determine if the capacitor has decayed, shown lines 313-329. If no tokens indicating capacitor decay have been found the delay is set to the design time, line 329. If relay has no capacitor, its delay time is set to 0.02 seconds to model the relay contact movement time.

```
309. T2 Delay(FS, RN) {
310.            //Example relay (CON) JR
311.            If (RN == "(CON) JR") {
312.                        //Is capacitor 75% degraded?
313.                        If (Present(("(CON) JR Capacitor", "75%""))) {
314.                                    Delay = 3.8seconds
315.                        }
316.                        //Is capacitor 50% degraded?
317.                        ElseIf (Present(("(CON) JR Capacitor", "50%""))) {
318.                                    Delay = 2.5seconds
319.                        }
320.                        //Is capacitor 25% degraded?
321.                        ElseIf (Present(("(CON) JR Capacitor", "25%""))) {
322.                                    Delay = 1.2seconds
323.                        }
324.                        //Is capacitor entirely degraded?
325.                        ElseIf (Present(("(CON) JR Capacitor", "0%""))) {
326.                                    Delay = 0.02seconds
327.                        }
328.                        //Otherwise capacitor is functioning normally
329.                        Else {Delay = 5seconds}
330.            }
331.            //other relays not shown for brevity
332.            Else If (RN == other relay) {}
333.            //action time is 0.02seconds if no capacitor present in relay
334.            Else {Delay = 0.02seconds}
335. }
```

Figure 6-31 Relay Module Extract: T2 Delay Function.

The arcs outgoing from T2 to places 'Relay States' and 'Contact States' are very similar to that of T1, highlighted in Figure 6-32. The arc towards 'Relay States' specifies that Colour 'RelayState' = "De-energised", as T2 models the de-energisation of relays. The arc from T2 to 'Contact States' calls the same function as T1 did to re-evaluate the contact states of the relay whose state has changed.

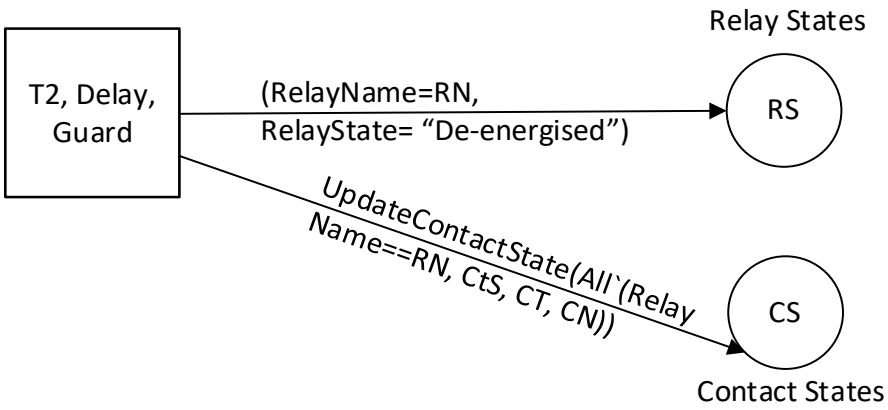

Figure 6-32 Relay Module Extract: T2 Outgoing Arcs.

6.3.2.5 Comparison with TPN Module

The CPN relay module is clearly quite different to its TPN counterpart. It is graphically much simpler, as the logical conditions which determine whether relay circuits are complete or not have been changed from a graphical presentation to high level programming functions. Because these unique conditions for each relay are still present, the CPN model is no simpler to present than the TPN. In many respects, adapting the TPN model into a CPN has likely made it harder to understand, as there are many more features to CPN models which must be understood.

However, the CPN module could be much easier to modify. In general, any change to the TPN module will require redrawing many repeated elements that are featured across all the relay TPNs. This may not be case for the CPN module. Consider if a new relay contact failure mode was to be added. This could be achieved with addition of several lines of code to the UpdateContactState function only.

6.3.3 Road Traffic Lights

The CPN flasher module activates and cycles the road traffic light flasher, which controls the alteration of the red lights. The CPN which models this is shown in Figure 6-33. This module uses a single place and three transitions. The place 'Flasher State' uses Colour Fl as a string. The place holds only a single token, whose string reads either "Off", "Left" or "Right" referring to the state of the flasher. As there is only one flasher unit for all road traffic lights, this net is initialised with a single token in the 'Flasher State' place containing the string "Off".

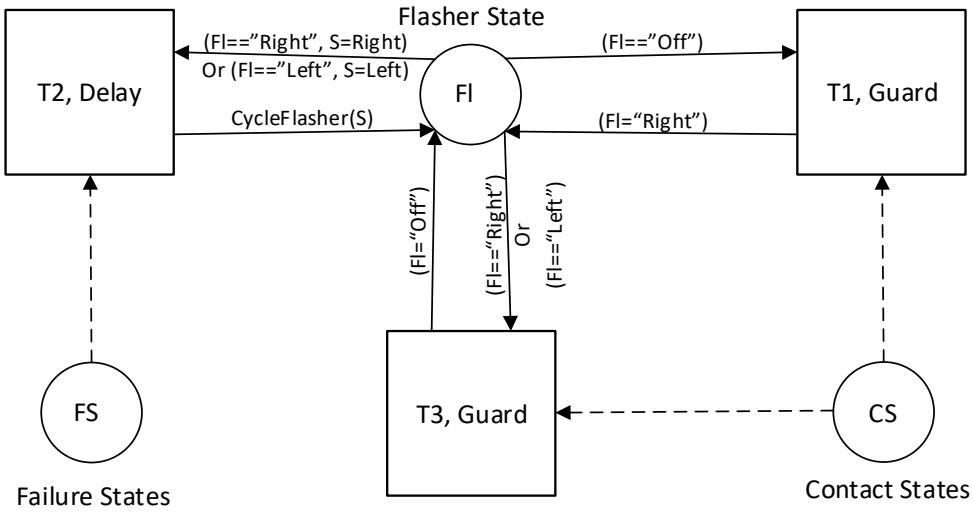

Figure 6-33 CPN Flasher Module

Transition T1 activates the flasher. It is enabled by a token in 'Flasher State' whose Colour Fl == "Off" and outputs a token whose Colour Fl = "Right". It contains a guard function which evaluates the conditions for activating the flasher, shown Figure 6-34. This guard function tests 'Contact States' to determine if either RER relay contact which can power the flasher circuit is closed. If neither contacts are closed the flasher will not activate, and hence T1 will not fire.

```
336. Colour Fl As String
337. Flasher State M0 [("Off")]
338. T1 Guard(CS){
339.            //is circuit complete?
340.            If (Present("RER", "Closed", "Back Contact", "3") Or
341.            Present("RER", "Closed", "Back Contact", "4") )
342.            { Transition = Enabled }
343.            Else {Transition = Disabled}
344. }
```

Figure 6-34 CPN Flasher: T1 Guard.

Transition T2 cycles the flasher from left to right or right to left. The expression within the incoming arc allows it to be enabled by either a token holding the string "Left" or "Right", passing this to the transition via variable 'S'. The output arc calls the function 'CycleFlasher', shown Figure 6-35 lines 353-358. This simple function outputs the opposite string to that input using the 'S' variable. The delay function is shown lines 345-352. It tests the place 'Failure States' for any failures which affect the flasher cycling rate and sets the delay accordingly.

```
345. T2 Delay(FS){
346.            //has flasher failed in slow mode?
347.            If (Present(("Flasher", "Slow"))) { Delay = 1.6seconds }
348.            //has flasher failed in fast mode?
349.            If (Present(("Flasher", "Fast"))) { Delay = 0.4seconds }
350.            //is flasher operating normally?
351.            Else { Delay = 0.8seconds }
352. }
353. CycleFlasher(S){
354.            //Output other side
355.            If (S = "Left") {Output 1`Fl("Right")}
356.            //Output other side
357.            If (S = "Right") {Output 1`Fl("Left")}
358. }
```

Figure 6-35 CPN Flasher Module: T2.

Transition T3 de-activates the flasher when the circuit which powers it is open. It is enabled by either the "Right" or "Left" token, as regardless of illuminated side, cutting power to the flasher de-activates it. It returns an "Off" token. The guard function which enables this transition is shown in Figure 6-36. This is very similar to the T1 guard function, however this transition is disabled by either RER contacts being completed, and enabled when neither is.

```
359. T3 Guard(CS){
360.            //Is flasher powered?
361.            If (Present("RER", "Closed", "Back Contact", "1")) Or
362.            Present("RER", "Closed", "Back Contact", "2") )
363.            { Transition = Disabled }
364.            //not powered
365.            Else {Transition = Enabled}
366. }
```

Figure 6-36 CPN Flasher Module: T3.

The CPN implementation of the flasher does not appear to offer any benefit over the TPN implementation. It may be marginally easier to understand its operation as it is graphically much simpler however.

Operation of the road traffic lights is modelled using the CPN shown in Figure 6-37. This CPN uses four places and six transitions. The transitions fire concurrently allowing this single CPN to model all road traffic lights.

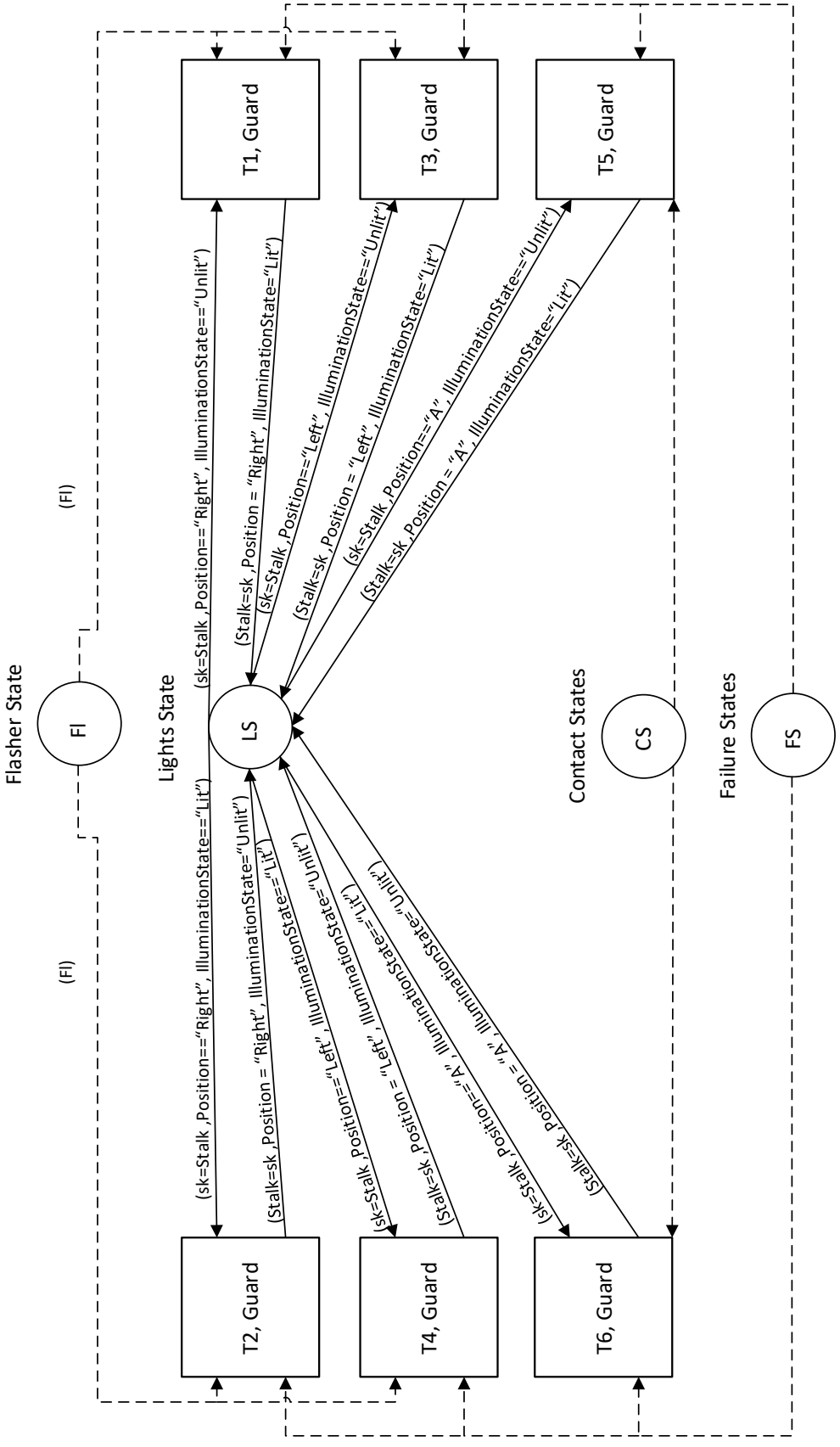

Figure 6-37 CPN Road Traffic Light Module.

The place ‘Lights State’ uses Colour set LS, defined in Figure 6-38. This place holds tokens which describe the illumination state of each bulb within the road traffic light system. Colour set LS, defined line 367, is formed from the union of three Colours. Colour ‘Stalk’ is defined as a string which is used to describe which of the four road traffic lights the bulb is housed in. Colour ‘Position’ describes the bulb position in that traffic light using a string, where “Left” and “Right” indicate the bulb is a red light on either the left or right side of the road traffic light unit. The string “A” indicates the bulb is the amber bulb positioned lower centre on the road traffic light. Finally, the ‘IlluminationState’ Colour specified if the bulb is either “Lit” or “Unlit”.

```
367. Colour LS As Stalk x Position x IlluminationState
368. Colour Stalk As String
369. Colour Position As String
370. Colour IlluminationState As String
371. Lights State M0 [(“YN”, “Left”, “Unlit”), (“YN”, “Right”, “Unlit”), (“YN”, “A”, “Unlit”), (“YO”, “Left”,
     “Unlit”),
372. (“YO”, “Right”, “Unlit”), (“YO”, “A”, “Unlit”), (“ZN”, “Left”, “Unlit”), (“ZN”, “Right”, “Unlit”), (“ZN”, “A”,
     “Unlit”),
373. (“ZO”, “Left”, “Unlit”), (“ZO”, “Right”, “Unlit”), (“ZO”, “A”, “Unlit”)]
```

Figure 6-38 CPN Road Traffic Lights: LS Colour Definition and Initialisation Tokens.

Transition T1 controls the illumination of all right hand red lights. The incoming arc from ‘Lights State’ specifies that only tokens whose ‘Position’ Colour = “Right” and ‘IlluminationState’ Colour = “Unlit” can enable this transition. When the transition fires it replaces a token in place ‘Lights State’ with the ‘IlluminationState’ Colour changed to “Lit”. The transition is controlled by a guard function shown in Figure 6-39, lines 374-376. This guard function calls another function, ‘RedLightLit’, if this function returns True then T1 is enabled, otherwise T1 is disabled.

The function ‘RedLightLit’ is defined over lines 386-390. It appends the variables ‘Stalk’ and ‘Pos’ to create a unique name which can be used to test the tokens in the Failure State place. An If statement on line 389 first checks if Pos == FlState. This is to determine if the state of the flasher matches the side of the bulb. If the flasher is set to “Right”, it follows that the right hand bulb will illuminate. The If statement on line 389 then tests the ‘Failure States’ place to determine if the bulb is functioning. If either the flasher is in the wrong state or the bulb has failed the transition will not fire leaving the token representing the bulb with the ‘IlluminationState’ Colour set as “Unlit”.

```
374. T1 Guard(Fl, FS, Stalk, Pos) {
375.            If (RedLightLit(Fl, FS, Stalk, Pos) == True) {Transition = Enabled} Else {Transition = Disabled}
376. }
377. T2 Guard(Fl, FS, Stalk, Pos) {
378.            If (RedLightLit(Fl, FS, Stalk, Pos) == False) {Transition = Enabled} Else {Transition = Disabled}
379. }
380. T3 Guard(Fl, FS, Stalk, Pos) {
381.            If (RedLightLit(Fl, FS, Stalk, Pos) == True) {Transition = Enabled} Else {Transition = Disabled}
382. }
383. T4 Guard(Fl, FS, Stalk, Pos) {
384.            If (RedLightLit(Fl, FS, Stalk, Pos) == False) {Transition = Enabled} Else {Transition = Disabled}
385. }
386. RedLightLit(Fl, FS, Stalk, Pos) {
387.            Variable CN As String = Append(Stalk + Pos)
388.            //check flasher output corrent and bulb not failed
389.            If ((Pos == FlState) AND NotPresent((CN, "Failed"))) {Return True} Else {Return False}
390. }
```

Figure 6-39 CPN Road Traffic Lights: T1 to T4 Guard Functions.

Transition T2 controls the extinguishing of right-hand red-light bulbs. It performs this function in a similar manner to T1. However, it differs in that when T2's guard function calls the 'RedLightLit' function it requires the function to return False in order to enable the transition. Thus allowing it to extinguish the red light as appropriate. Transitions T3 and T4 model the left-hand red-light bulbs in an identical manner to T1 and T2.

Transitions T5 and T6 control the amber lights. The arcs into and from these transitions follow the same format as transitions T1 to T4, however the 'Position' Colour is now specified to "A", denoting the central amber lights. Whilst the red lights are controlled by the flasher, the amber lights are controlled directly by relays within the control system. Due to this the guard functions for transitions T5 and T6 are different in form to the others.

Figure 6-40 shows the T5 guard function. The variable Stalk and Pos are appended to create a unique name for the bulb. A series of If statements, shown lines 393-408, then compare this unique name to the four possible names representing each of the four amber bulbs. This is necessary as different relay contacts control the illumination of each of the four amber bulbs. Thus the function must know which amber bulb is being evaluated in order for it to check the state of the correct relay contacts. The 'AmberLightLit' function is called, having been passed the appropriate relay contact numbers to check. If it returns True the transition is enabled, otherwise disabled.

The 'AmberLightLit' function is similar to the 'RedLightLit' function. It tests to see if all three relay contacts required to power the amber bulb are closed, and it further tests place 'Failure States' to ensure the amber bulb in question has not failed before returning True.

```
391. T5 Guard(CS, FS, Stalk, Pos) {
392.         Variable CN As String = Append(Stalk + Pos)
393.         If (CN == "YNA") {
394.                 If (AmberLightLit(CS, FS, CN) == True) {Transition = Enabled}
395.                 Else {Transition = Disabled}
396.         }
397.         If (CN == "YOA") {
398.                 If (AmberLightLit(CS, FS, CN") == True) {Transition = Enabled}
399.                 Else {Transition = Disabled}
400.         }
401.         If (CN == "ZNA") {
402.                 If (AmberLightLit(CS, FS, CN) == True) {Transition = Enabled}
403.                 Else {Transition = Disabled}
404.         }
405.         If (CN == "ZOA") {
406.                 If (AmberLightLit(CS, FS, CN) == True) {Transition = Enabled}
407.                 Else {Transition = Disabled}
408.         }
409. }
410. AmberLightLit(CS, FS, CN) {
411.         //is amber light circuit complete?
412.         If (AllPresent(("HER", "Closed", "Back Contact", HERC),("RER", "Closed", "Front
             Contact", RERC),("(CON) JPR", "Closed", "Front Contact", CONJPRC)) And NotPresent((CN,
             "Failed")))
413.         {Return True}
414.         Else {Return False}
415. }
```

Figure 6-40 CPN Road Traffic Lights: T5 Guard Function.

Figure 6-41 shows the guard function for T6, this transition models the amber lights extinguishing. This is very similar to that of T5, but crucially, the transition is enabled when the 'AmberLightLit' function returns false.

```
416. T6 Guard(CS, FS, Stalk, Pos) {
417.         Variable CN As String = Append(Stalk + Pos)
418.         If (CN == "YNA") {
419.                 If (AmberLightLit(CS, FS, CN) == False) {Transition = Enabled}
420.                 Else {Transition = Disabled}
421.         }
422.         If (CN == "YOA") {
423.                 If (AmberLightLit(CS, FS, CN) == False) {Transition = Enabled}
424.                 Else {Transition = Disabled}
425.         }
426.         If (CN == "ZNA") {
427.                 If (AmberLightLit(CS, FS, CN) == False) {Transition = Enabled}
428.                 Else {Transition = Disabled}
429.         }
430.         If (CN == "ZOA") {
431.                 If (AmberLightLit(CS, FS, CN) == False) {Transition = Enabled}
432.                 Else {Transition = Disabled}
433.         }
434. }
```

Figure 6-41 CPN Road Traffic Lights: T6 Guard Function.

The CPN module for the Wig-Wag traffic lights can be used to model many units concurrently. Though its expression requires a similar level of complexity compared to the TPN, addition of new Wig-Wag lights is simple. It also offers the same advantages that other CPN

implementations benefit from. Modification of the CPN is a much faster process than that of a TPN as only one Petri net must be changed, though it requires understanding the more complex CPN graphical presentation and syntax.

6.3.4 Road Traffic Barriers

6.3.4.1 Solenoid Valve

The CPN which models the barrier solenoid valves is shown in Figure 6-42. This CPN Has one place and two transitions which may test two further places. These transitions control the opening and closing of the solenoid valve for each barrier.

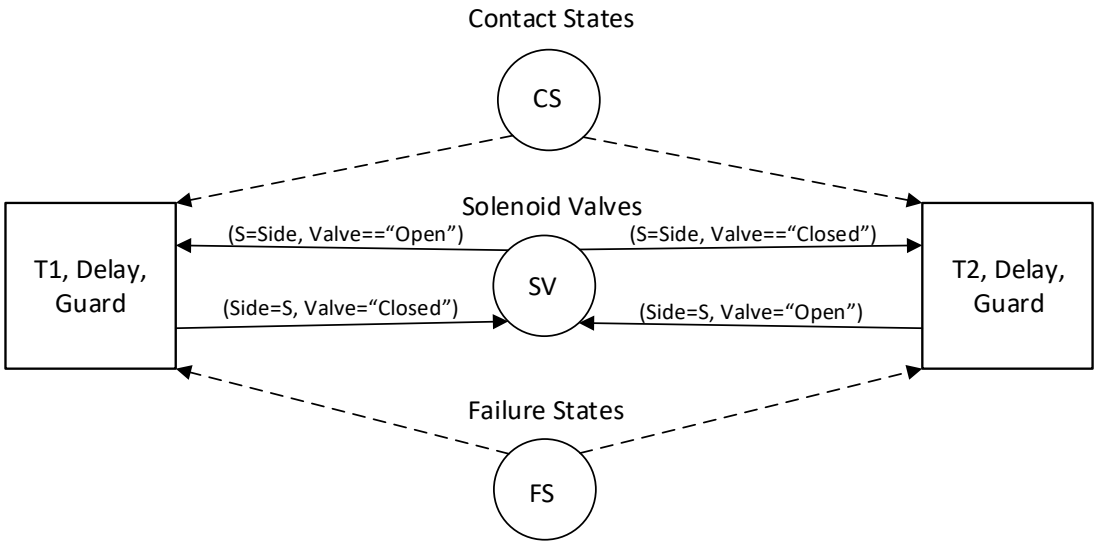

Figure 6-42 CPN Barrier Solenoid Valve.

The place ‘Solenoid Valves’ uses Colour set ‘SV’. This is formed from the union of Colours ‘Side’ and ‘Valve’, each of which uses a string to describe the side of the road the crossing the barrier is positioned on and the state of the valve respectively. This place always holds two tokens, one describing the state of the solenoid valve within each barrier unit.

```
435. Colour SV As Side x Valve
436. Colour Valve As String
437. Colour Side As String
438. Solenoid Valves M0 [(“Y”, “Closed”), (“Z”, “Closed”)]
```

Figure 6-43 CPN Barrier Module: Solenoid Valve Colours.

Transition T1 performs the function of closing the solenoid valves when appropriate. It is enabled by tokens whose ‘Valve’ Colour = “Open”. When this transition fires it outputs a token whose ‘Valve’ Colour = “closed”. This transition has a delay of 0.4seconds to model the movement time of the valve. The Guard function for T1 is shown in Figure 6-44. This function determines which side of the crossing the barrier is located and passes the appropriate parameters to the ‘ValveClosed’ function. This function evaluates if the valve should be

closed by testing the state of the relay contacts which control it and testing for any failures which may affect its function. If the function returns True the transition is enabled and the valve closes.

```
439. T1 Delay = 0.4seconds
440. T1 Guard(CS, FS, Side) {
441.            //create and set variable for component name
442.            Variable CN As String = Append(Side + " Solenoid Valve")
443.            If (Side == "Y") {
444.                        //is Y side valve closed?
445.                        If (ValveClosed(CS, FS, CN, 1, 2, 3) == True)
446.                        {Transition = Enabled} Else { Transition = Disabled}
447.            }
448.            If (Side == "Z") {
449.                        //is Z side valve closed?
450.                        If (ValveClosed(CS, FS, CN, 2, 4, 5) == True)
451.                        {Transition = Enabled} Else { Transition = Disabled}
452.            }
453. }
454. ValveClosed(CS, FS, CN, C1, C2, C3) {
455.            //has valve failed in closed position?
456.            If (Present(("CN", "Stuck Closed"))) {Return True}
457.            //is circuit which closes valve complete?
458.            ElseIf (AllPresent(("(CON) JPR", "Closed", "Back Contact", C1), ("(CON) JPR", "Closed", "Back
                Contact", C2), ("(CON) JPR", "Closed", "Back Contact", C3))  {Return True}
459.            Else {Return False}
460. }
```

Figure 6-44 CPN Solenoid Valve: T1 Guard Function.

Transitions T2 performs the opposite function of opening the valve. Like T1, it has a 0.4 second delay to model the movement time of the valve. Its guard function is shown in Figure 6-45. This is near identical to that of T1, however now the transition is enabled when the 'ValveClosed' function evaluates to False.

```
461. T2 Delay = 0.4seconds
462. T2 Guard(CS, FS, Side) {
463.            //create and set variable for component name
464.            Variable CN As String = Append(Side + " Solenoid Valve")
465.            If (Side == "Y") {
466.                        //is Y side valve open?
467.                        If (ValveClosed(CS, FS, CN, 1, 2, 3) == False)
468.                        {Transition = Enabled} Else { Transition = Disabled}
469.            }
470.            If (Side == "Z") {
471.                        //is Y side valve open?
472.                        If (ValveClosed(CS, FS, CN, 2, 4, 5) == False)
473.                        {Transition = Enabled} Else { Transition = Disabled}
474.            }
475. }
```

Figure 6-45 CPN Solenoid Valve: T2 Guard Function.

6.3.4.2 Barrier Position

The barrier position CPN models the movement of the barriers. The graphical portion of this CPN is shown in Figure 6-46. This CPN introduces the 'Barrier Position' place. This place uses

the Colour set BP, formed by the union of Colours ‘Side’ and ‘Angle’. Tokens in this place specify which side of the crossing the barrier is located, and the angle of the barrier. It uses two transitions, one to lower the barrier and another raise it.

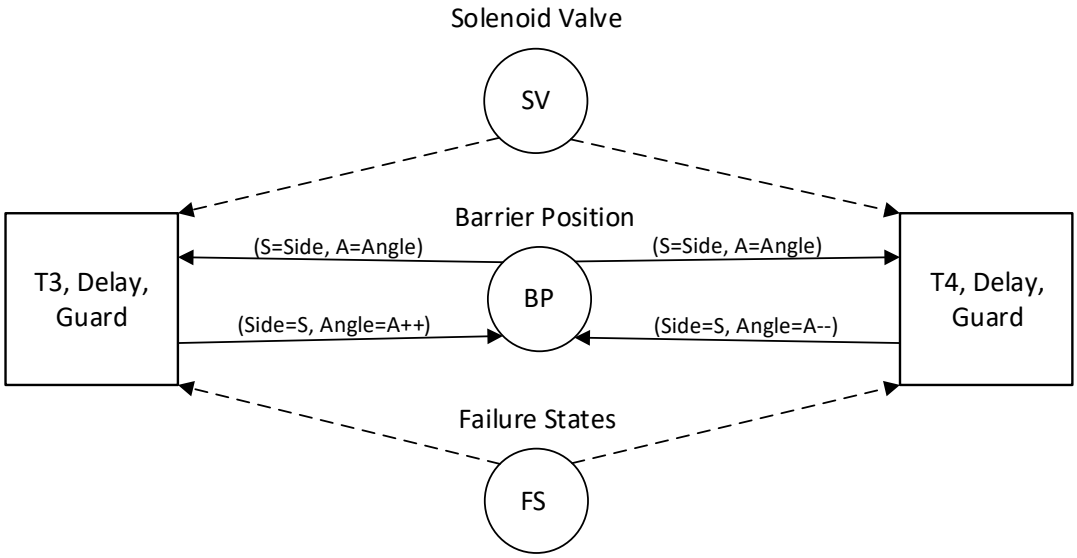

Figure 6-46 CPN Barrier Position.

Transition T3 increments the angle of a barrier by 1degree from horizontal every time it fires. Its delay function is shown in Figure 6-47, lines 479-485. The delay function checks ‘Failure States’ for a token indicating if the barrier has failed in a manner which slows its rising. If found, its delay is set to 0.1 seconds. Meaning every 0.1 seconds the barrier will raise 1 degree. If a token indicating failure is not found the delay is set to 0.05 seconds.

Transition T3 features a guard function, shown lines 493-502. This function evaluates three conditions before enabling the transition. The angle must be less 84 degrees, the limit to which the barrier will rise. The Solenoid valve for the barrier unit must be closed, and there must be no failures present which will prevent the barrier from rising.

```
476. Colour BP As Side x Angle
477. Colour Angle As Integer
478. Barrier Position M0 [(“Y”, 84), (“Z”, 84)]
479. T3 Delay(FS, Side) {
480.            //create and set variable for Component Name
481.            Variable CN As String = Append(Side + “ Barrier”)
482.            //has barrier failed in a manner which slows its rise?
483.            If (Present(CN, “Slow Rise”)) {Delay = 0.1seconds}
484.            Else {Delay = 0.05seconds}
485. }
486. T4 Delay(FS, Side) {
487.            //create and set variable for Component Name
488.            Variable CN As String = Append(Side + “ Barrier”)
489.            //has barrier failed in a manner which slows its fall?
490.            If (Present(CN, “Slow Fall”)) {Delay = 0.1seconds}
491.            Else {Delay = 0.05seconds}
492. }
493. T3 Guard(SV, FS, Side, Angle) {
494.            //create and set variable for Component Name
495.            Variable CN As String = Append(Side + “ Barrier”)
496.            //has barrier over rotated?
497.            If (Angle >= 84) { Transition = Disabled}
498.            //if nothing to prevent it, raise barrier
499.            ElseIf (Present((Side, “Closed”)) And NotPresent((CN, ”Unable To Rise”))) {
500.                        Transition = Enabled} Else { Transition = Disabled }
501.            }
502. }
503. T4 Guard(SV, FS, Side, Angle) {
504.            //create and set variable for Component Name
505.            Variable CN As String = Append(Side + “ Barrier”)
506.            //has barrier over rotated?
507.            If (Angle <= 0) { Transition = Disabled}
508.            //if nothing to prevent it, lower barrier
509.            ElseIf (Present((Side, “Open”)) And NotPresent((CN, ”Unable To Fall”)))
510.            {Transition = Enabled} Else { Transition = Disabled }
511. }
```

Figure 6-47 CPN Barrier Position Code.

Transition T4 models the lowering of the barrier in a similar manner to that of T3. A guard function evaluates if the barrier should be descending, and a delay function determines the rate. In this instance, the guard function evaluates the barrier angle to prevent the barrier from descending beyond 0 degrees, and also checks that the solenoid valve is open and no failures which would prevent descent have been set.

6.3.4.3 Barrier Contactors

The CPN for simulating the barrier contactors is shown in Figure 6-48. This CPN models the opening and closing of the barrier contactors which report the position of the barrier to the control system. This CPN Uses one place and two transitions which test a further two places.

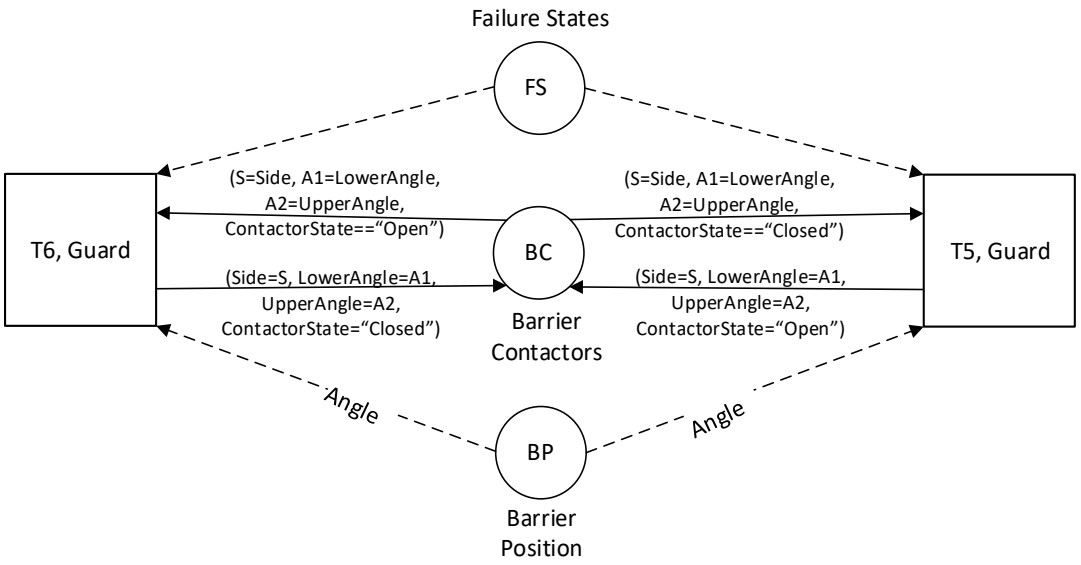

Figure 6-48 CPN model for Barrier Contactors.

The ‘Barrier Contactors’ place hold tokens which contain information on the position, configuration and state of each the barrier contactors. The place uses the BC Colour set, formed from the union of four Colours. Colour ‘Side’ details which side of the crossing the barrier contactor is situated on. Colour ‘LowerAngle’ is an integer corresponding to the smallest angle which will complete the barrier contactor circuit. Similarly, Colour ‘UpperAngle’ is an integer corresponding to the highest angle which will complete the contactor circuit. Together they form the interval through which the contactor circuit is complete. Lastly, Colour ‘ContactorState’ holds a string indicating if the specific contactor circuit is open or closed.

```
512. Colour BC As Side x LowerAngle x UpperAngle x ContactorState
513. Colour Side As String
514. Colour LowerAngle As Integer
515. Colour UpperAngle As Integer
516. Colour ContactorState As String
517. Barrier Contactor M0 [("Y", 0, 4, "Open"), ("Y", 81, 90, "Closed"), ("Y", 42, 90, "Closed"), ("Y", 0, 83,
     "Open"),
518. ("Z", 0, 4, "Open"), ("Z", 81, 90, "Closed"), ("Z", 42, 90, "Closed"), ("Z", 0, 83, "Open")]
519. T5 Guard(FS, Angle, Side, A1, A2) {
520.          //create and set variable for Component Name
521.          Variable CN As String = Append(Side +  A1 + "-" + A2)
522.          //is barrier within contactor range, or failed open?
523.          If((Angle < A1 Or Angle > A2) OR Present(CN, "Open"))
524.          //open contactor
525.          {Transition = Enabled}
526.          //leave closed
527.          Else {Transition = Disabled}
528. }
529. T6 Guard(FS, Angle, Side, A1, A2) {
530.          //create and set variable for Component Name
531.          Variable CN As String = Append(Side +  A1 + "-" + A2)
532.          //is barrier outside contactor range and not failed open?
533.          If((Angle >= A1 And Angle <= A2) And NotPresent(CN, "Open"))
534.          //close contactor
535.          {Transition = Enabled}
536.          //leave open
537.          Else {Transition = Disabled}
538. }
```

Figure 6-49 CPN Barrier Contactor Code.

Transition T5 controls the opening of barrier contactors. It can be enabled concurrently by any token whose 'ContactorState' Colour reads "Closed". The transition is further controlled by a guard function, shown Figure 6-49 lines 519-528. This function checks to see if the current angle of the barrier is outside the interval specified by the 'LowerAngle' and 'UpperAngle' Colours. It also tests 'Failure States' to determine if the specific contactor has failed open circuit. If either is true the transition may fire, opening the contactor.

Transition T6 controls the closing of barrier contactors. It operates in a similar manner to T6. The main difference is the guard function. T6's guard function evaluates if the barrier angle is within the interval where the contactor circuit will close, and if the contactor has not failed open circuit. If these conditions are met, the transition will fire closing the contactor circuit. Shown lines 529-538.

The CPN is much more concise than that of the TPN version. This was made possible due to the simple nature of barrier contactors. They each feature only one angle interval through which they can be completed, the bounds of which are easily specified as Colours in a token, and tested with a universal function.

6.4 Results

As both the CPN and the TPN are logically identical, only the TPN was simulated. Later chapters will test for computational difference between CPN and PN models. This has not been performed here as the models are deterministic and will not benefit from faster simulation as stochastic models solved via Monte Carlo methods might.

The TPN model was used to determine the effects of component's failures on the functioning of the protection systems and self-reporting system. The model was configured for the specific failure scenarios using the 'Failure Selector' places and the passage of several trains simulated. The model was simulated quickly, requiring only ~20seconds to simulate 400 scenarios.

The software which simulates the model records the state of both barriers and lights on either side of the crossing, and fault reporting systems at fixed intervals. This is then used to determine if the lights and barriers followed the correct timing. This information can also be used to determine if any fault will be self-reported, discovered during an inspection or neither. An example of these results can be seen in Table 6-1 for a relay contact within the B TPSR relay. The results show that if this relay contact welds, the first train after the welding event passes completely unprotected, though this failure is reported. The second train which passes after this train is protected, this is because the crossing detects the train exiting the crossing as if it was entering from the opposite direction. As the train was not passing from the opposite direction it does not strike out and thus the protection systems remain active until the crossing is reset electrically.

Table 6-1 Model Results for B TPSR Contact 1 Welded.

B TPSR Contact 1 Welded	
1st Train	2nd Train
Y Lights – Off	Y Lights – Active since 1st train passed
Z Lights – Off	Z Lights – Active since 1st train passed
Y Barrier – Raised	Y Barrier – Active since 1st train passed
Z Barrier – Raised	Z Barrier – Active since 1st train passed
Self-Reporting – No operation reported (hazardous failure self-reported)	Self-Reporting – Constantly reporting protection system active (safe failure self-reported)

Due to the scale of the output of this model, its results cannot be reproduced in this thesis, but instead are summarised.

6.4.1 Fail Safe Events

The model was used to discover component failures which have non-hazardous effects. This was achieved by simulating each possible component failure and failure mode and recording the effects for later analysis. A summary of the results of this are shown in Table 6-2. These results are used to configure the effects of component failure in the asset management models presented in the following chapter.

Table 6-2 Summary of Fail Safe Component Failures Modes, Number in Brackets Indicates the Number of Failures Which Were Self-Reported.

	Track Circuit	Control System	Barrier	Lights
Lights Stuck On	7 (7)	47 (45)	12 (12)	0 (0)
Barrier Stuck Down	7 (7)	39 (39)	12 (12)	0 (0)
Minor Timing Issue	0 (0)	38 (38)	2 (0)	0 (0)
No Observable Effect	11 (1)	261 (5)	0 (0)	0 (0)
Complete Fail Safe	7 (7)	39 (39)	12 (12)	0 (0)
Total Component Failure Modes	18	348	14	15
Total Components	9	168	14	13

The results of the analysis were as expected. Failures of track circuit components mostly produced no effect, on account of the high level of redundancy – this is because the track circuits are operated by both track circuits and treadles which fulfil the same purpose. Where track circuits did effect the operation of the AHBC they generally caused a complete fail safe event where both barriers and lights remain activated until repaired. In two instances treadle failure caused a minor timing issue as train detection using track circuits alone can be slower.

The control system is the most complex of the systems, containing 168 components with 348 failure modes considered. The vast majority of these failures do not produce any effect. This is mostly due to the bi-directional design of AHBCs. All AHBCs are designed to operate bi-directionally without any additional re-configuration, however in practice are only operated in this manner in exceptional circumstances, allowing component failures which affect the unused direction to remain hidden. This presents a network resilience issue, as a serious fault which leads to a line being operated bi-directionally as a shuttle service could reveal a previously hidden level crossing failure causing further disruption.

Most barrier system failures trigger a complete fail self-event. This is unsurprising as the control system features cross-proving circuits which detect barrier position, allowing the crossing to detect a failure of the barriers to operate. The same is true of the road traffic

lights. Barrier position is self-reported to the monitoring signal box, allowing all failures which result in the barriers not functioning to be discovered. Failures which cause the barriers to descend slower than normal can only be detected via inspection.

No single failure of the road traffic light system can cause a fail-safe event, however all are detectable via inspection. This is because more than one bulb per light stalk must fail before the system self-reports. A failure of a single bulb does not constitute a hazardous failure as there are two red bulbs per light stalk, and two light stalks per side of the road, allowing drivers ample opportunity to observe the lights even if one bulb has failed.

Unlike hazardous failures, no fault tree was created for fail safe events. As a result, when applying these results to the development of an asset management model later in this thesis the number of fail safe events is likely to be under estimated. However, any events which require multiple elements to be failed concurrently will be much rarer than those which require only a single failed element. Therefore it is not expected this will have a significant impact on any results produced by the asset management model.

6.4.2 Hazardous Failures

This model can be used to find hazardous failures, and combinations of component failures which are hazardous. However, to check all possible component failure combinations would require simulating approximately 400^{400} combinations and thus is computationally intractable. Instead, fault trees were developed. The simulation model is used to confirm these fault trees are correct, and to determine which failures are self-reported via simulation. The fault trees and Petri net simulation model produced the same results.

Figure 6-50 shows the fault tree for a train passing whilst the Y-side barrier is in the raised position. It has been annotated with red dots indicating which events cause the top level event, and green squares indicating which events are self-reported. It shows that in three out of four areas where there is redundancy, a single component failure will not be self-reported. If relay contact (CON)JPR4 fails through welding, its condition will remain until the component is replaced via a renewal or (CON)JPR3 fails triggering a hazardous failure and enabling discovery of both failed components.

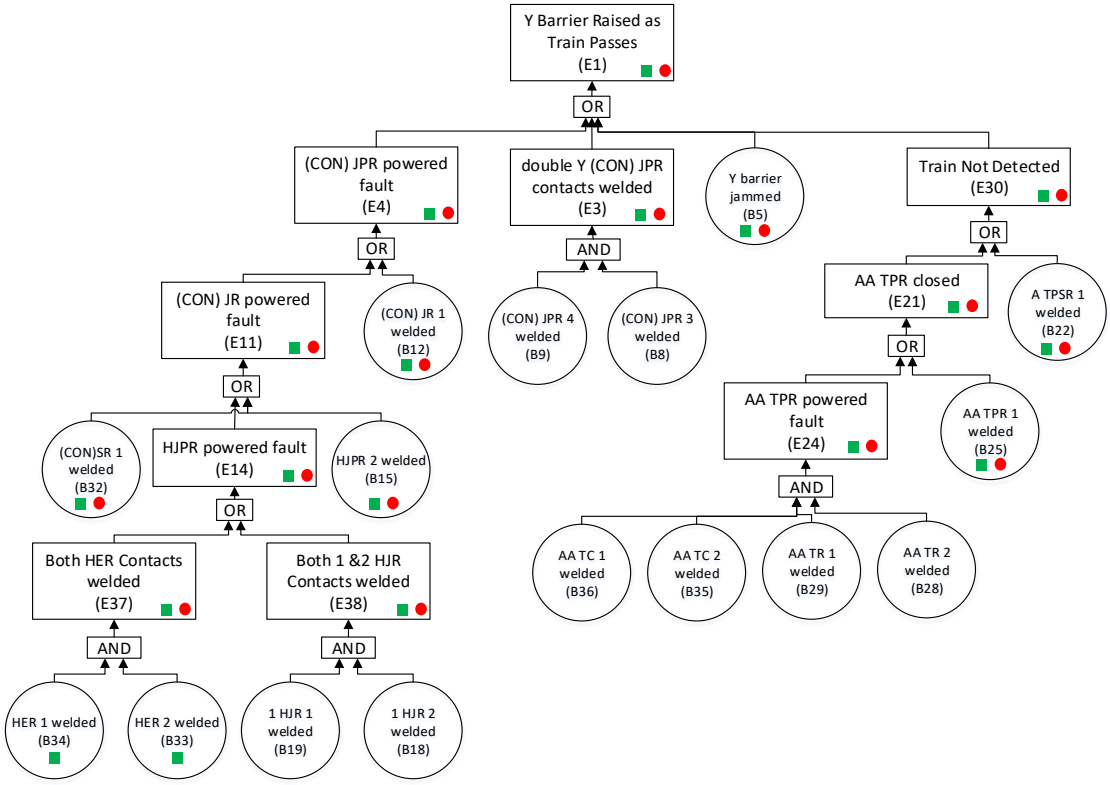

Figure 6-50 Fault Tree for Y Barrier Raised as Train Passes, Green Squares Indicate Fault is Self-Reported, Red Dots Indicate Event is Sufficient to Cause Top Level Event.

Figure 6-51 shows the fault tree for Y-lights not activated whilst a train passes. Like the previous fault tree, a green square indicates the event is self-reported and a red dot indicates the top level event occurs. A hollow green square indicates the event is discoverable only when the crossing is inspected. The fault tree shows that event 35 'RER Powered Fault' is unlikely to occur, as all of the component failures which comprise the minimal cutsets which can trigger this event feature at least two independent failures which will be self-reported. In practical terms, this means that a large number of component failures must occur at the same time to trigger event 35, which in the absence of a common cause failure is of negligible probability. Considering also that relay contact welding is a rare event, the most probable causes of the lights failing to illuminate (E1) are: multiple bulb failures (E2); power failure (E3); and both RER contacts failing high resistance (E30).

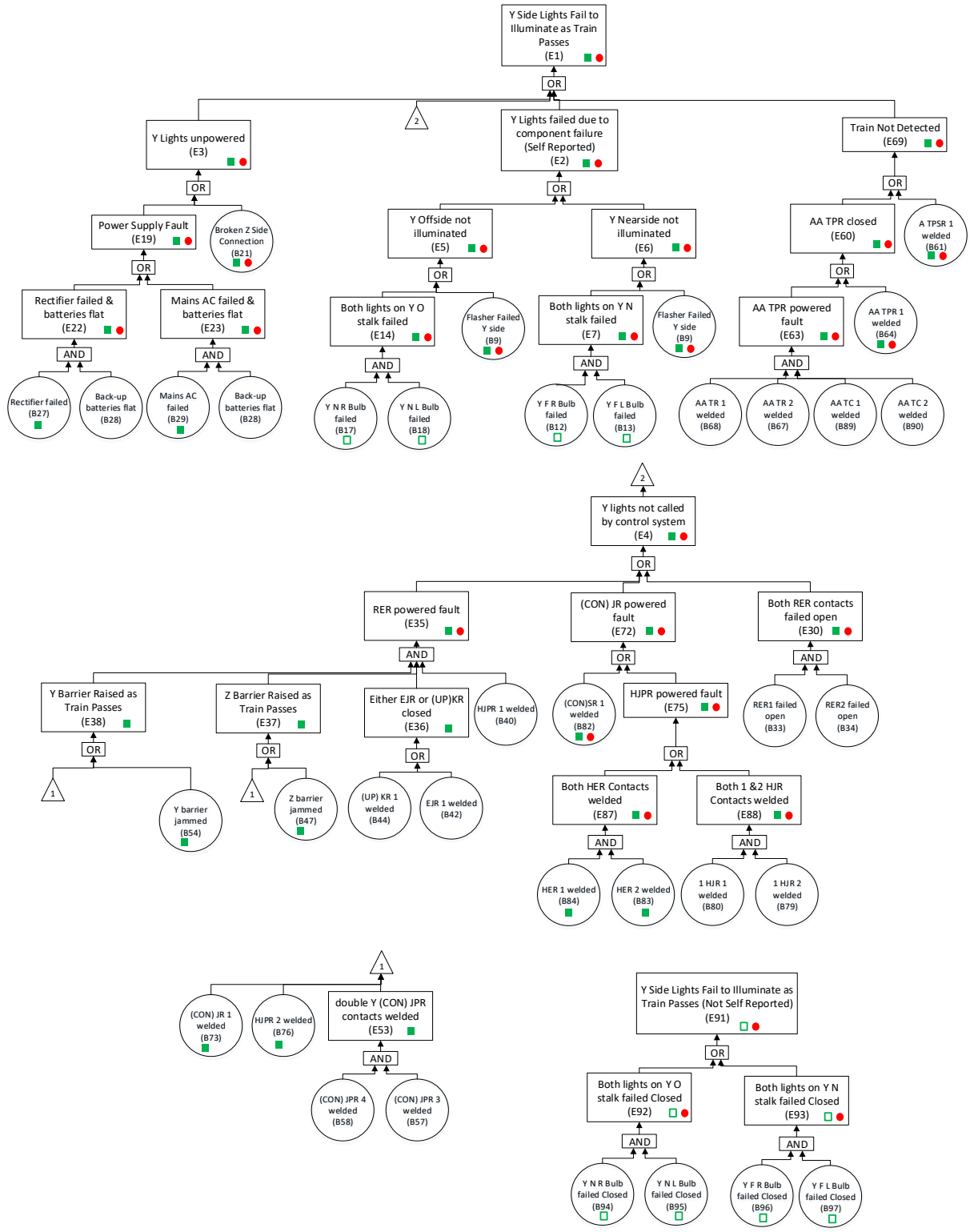

Figure 6-51 Fault Tree for Y Lights Off as Train Passes, Green Squares Indicate Fault is Self-Reported, Red Dots Indicate Event is Sufficient to Cause Top Level Event, Hollow Green Squares Indicated Failure is Discoverable by Inspection Only.

6.5 Summary

This chapter has presented a timed Petri net model for simulating the effects of specific component failures on the operation of an Automatic Half Barrier Crossing. The same model has been recreated as a CPN for comparison. No clear advantage between either methodologies was found in this application. The TPN model tended to be graphically complex, whereas the CPN was often graphically simpler, and often still more concise even when the programmatic function are included. However, the CPN requires a much greater level of understanding to manipulate compared to the simpler TPN. The TPN requires understanding a graphical modelling language with around ten elements only. The CPN requires knowledge of a more complex graphical modelling language in addition to a novel programming language. If the CPN were to be simulated, it would have required more complex software for the same reasons.

The TPN model was used to determine the effects of each possible component failure in isolation. The effects determined were the operational state of the crossing's protection systems, whether the fault was self-reported, and whether it would be detected during an inspection. Hazardous failure modes were found which cause the protection systems to malfunction. Combined with the driver modelling results in the previous chapter, this demonstrates a correlation between crossing condition and risk of road vehicles colliding with trains. These results are used in the next chapter to model the effects of component failures in an asset management context.

Hazardous failures were modelled using fault trees, to allow consideration of the effects of multiple concurrent component failures on crossing function. The TPN simulation model was used to verify the accuracy of these fault trees. The simulation model was further used to analyse which of the fault tree's event gates would trigger the self-reporting system to report a failure, and which could be discovered during inspection. These effects are incorporated into the asset management model in the next chapter.

The model is created by considering the effects of component failure in mostly Boolean terms. This fits well with the relay logic based control system, however it ignores complex electrical effects. A laboratory study on an AHBC's control system could determine the accuracy of the modelling approach adopted, however was not performed due to resource constraints, therefore is left to further work. This could entail creating high resistance or welding type failures through deliberate modification of the relay contacts, and observing the effects comparing them with the simulated system behaviour predictions.

A more thorough methodology would be to collect failed components, and replaced but otherwise functional components from level crossing assets. The model presented in this chapter considers relay contacts to be either working or failed. However, it does not consider if relay contacts may have intermittent failure states. For example, several near high-resistance failure components within the same circuit may create an intermittent high resistance failure depending on the operational state of the crossing. This could be explored further but would require extensive laboratory work to qualify and quantify these states.

Verification of the accuracy of the model will suffer from the same constraints found in using the model, namely that it would be impossible to test all combinations of relay and other component failures. As such, any physical testing to verify the model's results should be limited to likely failure modes such as a single component failures, and extremely hazardous failure modes where improbable combinations of failures are predicted to produce severe effects.

A simpler method of verifying the accuracy of this model would be to review failure records of this asset class, noting the components which were replaced to resolve the failure state and confirming if the model can reproduce these effects. This however can only consider failure models which have occurred in the past, and so is unlikely to be able to verify less probable failure scenarios where multiple components fail simultaneously. It will also be limited by the thoroughness and accuracy of the fault diagnosis and record keeping processes.

7. Application of Asset Management Modelling Methodologies

This chapter presents an asset management model for Automatic Half Barrier Crossings (AHBC). The model estimates maintenance costs, crossing operational state and frequency of both safe and hazardous failures. The model has been formulated using several different Petri net methodologies for comparison of their use in a railway asset management modelling framework. This evaluation focuses on the computational differences for each methodology. Following the In2Rail framework, this is an example of an integrated model as it comprises the results of the functional model presented in chapter 6, a degradation model, and a maintenance model.

7.1 Model Overview

The model comprises six core modules listed below, with the general structure shown in Figure 7-1.

- Component Failure
- Equipment Renewal
- Discovery
- Protection System States
- Reactive Maintenance
- Data Logging

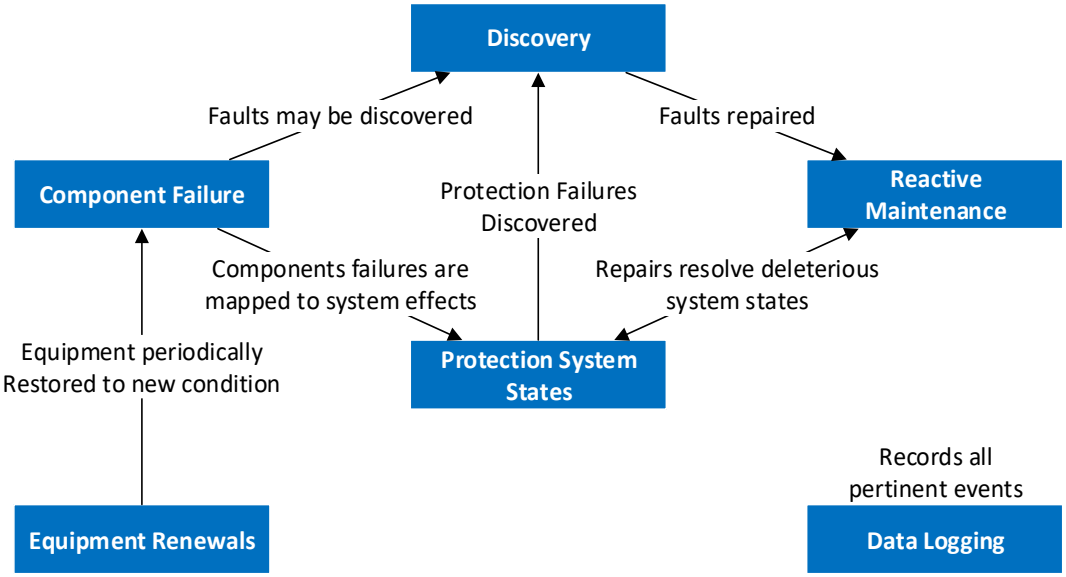

Figure 7-1 Asset Management Model spilt into Modules and Their Interactions.

7.1.1 Component Failure Module

The Component Failure module comprises a Petri net for each modelled component which may be separated into elements. The distinction between a component and a component element is that only the component may be replaced, thus the functionality of any failed element can only be restored via the replacement of the entire component. Random variables are used to determine the time until each component or element fails. When a component or element fails it may affect the crossing protection systems. When this occurs the Protection System States module will be updated reflecting the effects of the component failure.

7.1.2 Equipment Renewal Module

Periodically, components may be replaced by the Equipment Renewal module. This occurs with the regular frequency defined in the model and may differ for each system which is renewed. Once the renewal period elapses, a work order is generated for the renewal of that system and a random variable is sampled to determine the time at which that work begins. In a later chapter focusing on framework development, the time till the work begins will be determined by a scheduling module and is function of available work hours and work order priority. Equipment may also be renewed prior to the renewal period elapsing, known as an opportunistic renewal. This event is also randomly generated. However, in the later framework chapter this will again be determined by the availability of budget. Equipment renewal occurs within the model by restoring all components within a system to a new state.

7.1.3 Protection System States Module

This module controls the current operational state of the crossing's protection systems. Barriers and road traffic lights may fail to operate as a train passes, or operate continually in the absence of an approaching train. These protection systems may also operate later than intended when an approaching train is registered by the crossing's track circuits.

The state of the protection system is determined as a function of the currently failed components or elements. Safe protection system states, such as those which cause continual operation of protection systems are determined by assessing the failure of single components only. Hazardous protection system states are determined more accurately by use of a fault tree (see chapter 6) which enables the effects of combinations of component failures to be considered. Any component, or combination of components which causes an abnormal protection system state is made detectable during inspection or repair work.

7.1.4 Discovery Module

The Discovery module controls when and how failed components are discovered. Inspections periodically check for discoverable failed components. A failed component is only discoverable during an inspection if it is easily visually identified as failed or if it effects the operation of the protection system. Inspections occur periodically, when an inspection is due a work order is created for that inspection. There are two levels of urgency for an inspection, with the higher level of urgency reserved for overdue inspections. The time until an inspection actually occurs is determined randomly, however in the framework examples in later chapters it is determined based on real time availability of staff. Inspections also occur after every repair or renewal action.

Failed components may also be discovered in real-time by the crossing's self-reporting system. The discovery of any fault which affects the protection systems results in the generation of an urgent work order. Faults with no additional accident risk associated result in the generation of routine work orders.

7.1.5 Reactive Maintenance Module

The reactive maintenance module controls the repair of any failed components of the crossing, in fulfilment of either an urgent or routine work order. Following the creation of either work order type the time till a repair begins is stochastic, but on average much shorter for urgent work. The time for the repair to complete is determined from the sum of the repair time of all the components which are being replaced.

7.1.6 Data Logging Module

The data logging module counts the occurrences of any important events. These are listed below:

- Cost of replacement parts per system
- Number of renewals carried out per system
- Number of inspections
- Number of trains delayed
- Work orders generated
- Power failures and their cause
- Number of trains passing the crossing per protection system malfunction type

The last point concerns trains passing whilst the crossing protection systems are not operating correctly. This is important as it is used to determine the expected frequency of

collisions at a crossing. Each train passage is recorded between the start of the protection system failure, and an urgent work order being issued. It is assumed that after the fault has been discovered all trains using the line will either be re-routed, or instructed to pass at caution, meaning they will pass slow enough to negate any accident risk. After an urgent work order has been submitted, but before a work crew arrives onsite each passing train will be recorded as delayed.

7.2 Petri Net Structure

The Petri net structure of the model is described below. The distributions and parameters used for times to failure are presented in appendix B, along with component costs and repair times.

7.2.1 Component Failure Module

Each component within the model that can fail is modelled individually. This is to take into account the wide range and highly specific effects which occur when specific components fail. The Petri nets used to model individual component failure follow a standard form. The general component Petri net model is shown Figure 7-2. This Petri net models a component's failure and eventual replacement.

At the beginning of simulating the model each component is in its working state. This is signified by a single token in place P-GC1. This token enables transition T-GC1 which samples a random number from an appropriate continuous distribution to determine the time to failure. As the crossing systems are entirely inactive whilst no trains are approaching, failures are often only likely to occur as a train is passing. A second special transition is used to account for this.

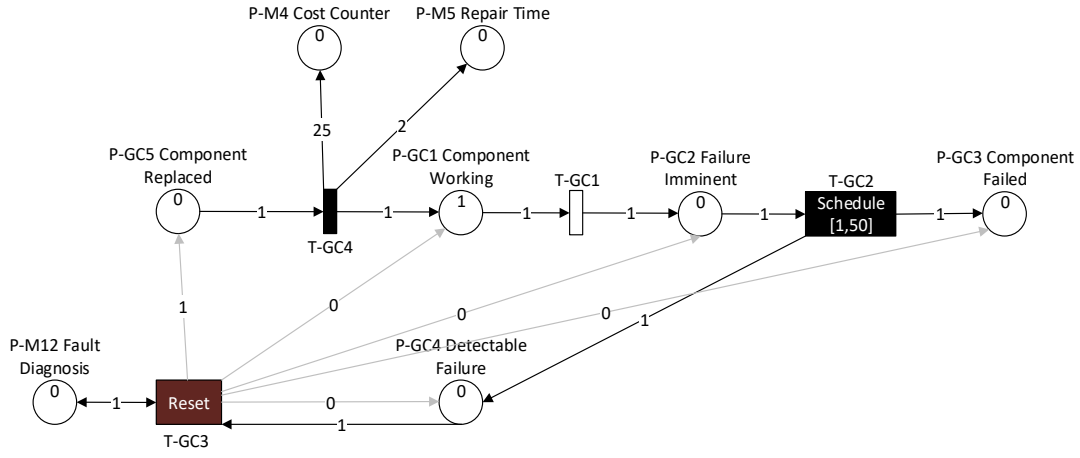

Figure 7-2 General Component Failure Petri Net.

When transition T-GC1 fires a token is removed from P-GC1 and a token placed in P-GC2 indicating that component failure will occur imminently. This enables the aforementioned special transition, T-GC2. This transition shall be referred to as a Schedule Transition. When enabled, this transition uses a uniform distribution to select a number, N, between 1 and 50. It then looks up the time till the Nth train is scheduled to pass and uses this value as its delay.

This is important for three reasons, chiefly level crossings do not perform any functions in the absence of an approaching train. As such many components are likely to fail only when they are operating – when a train is approaching or passing. Secondly, if the failure causes the protection systems to malfunction, having that malfunction begin at the correct time allows more accurate recording of the number of trains which pass before the fault is reported and then rectified. Thirdly, as a consequence of the first point, level crossings are much more likely to fail during peak times. By uniformly selecting which train will trigger the failure from a wide range, peak times will be represented higher in the overall failure times, better reflecting reality.

Should a component's failure be discoverable in isolation an arc is placed between T-GC2 and place P-GC4. The presence of a token in this place allows the component to be replaced when reactive maintenance occurs via transition T-GC3. This reset transition is enabled by the aforementioned place, and place P-M12, which holds a token only when reactive maintenance is ongoing. When the reset transition fires it resets the places in the net to the values indicated by the outgoing grey arcs. Rather than restore a token to place P-GC1 indicating the component is now in a working state, it instead sets the number of tokens of P-GC5 to one. This enables transition T-GC4, which fires without delay adding tokens to place P-M4 to account for the cost of the repair. It also adds tokens to place P-M5 corresponding to the time required to complete the repair, this place is shared between all components and indicates the length of time for all discoverable failed components to be repaired. It is important that this occurs when the repair is underway and not before. Some component failures may be undiscoverable in the absence of other concurrent failures, therefore the repair time and replacement costs should only be determined at the time of repair.

7.2.2 Relay Contact Failure

The relays which perform the logical operations of the control system contain multiple pairs of relay contacts, each of which may fail individually. Failed relay contacts are not repaired, instead the entire relay is replaced when a failure is discovered. The effects of relay contact failure depend on which specific set of contacts has failed. The effects of this component

type failing were determined using the fault tree and FMEA model described in chapter 6. Relay contacts may fail either open circuit due to high contact resistance (Kagra, 2013), or closed circuit via welding (Gao, et al., 2014). This requires modification of the general component failure net presented above.

An example relay contact failure Petri net is shown in Figure 7-3. This example relay contains two pairs of contacts. The structures labelled 'GC' are unchanged from the general component Petri net, whereas those labelled 'RF' are used to model relay contact failure. Transition T-RF1 models open circuit failure of the first pair of contacts. When enabled by a token in P-GC1, this transition fires following a Weibull distributed delay time, adding a token to place P-RF1 indicating that open contact failure is imminent. As the transition is enabled via test arc, an inhibitor arc from P-RF1 to T-RF1 prevents the transition from firing a second time.

Now place P-RF1 enables transition T-RF2. This is a schedule transition behaving exactly as described in the general component failure Petri net earlier. Open circuit failures in relay contacts are caused by electrical arcing as the relay opens or closes, thus the schedule transition is appropriate here as the relay contacts will only change state in response to a train approaching and passing the crossing. When this transition fires the token is removed from place P-RF1 and a token placed in place P-RF2, signifying that the relay contacts have now failed open. Inhibitor arcs from place P-RF2 inhibit transitions T-RF3 and T-RF4 preventing mutually exclusive failure modes occurring to a single relay contact.

Transitions T-RF3 and 4 model closed circuit failure in a similar manner. Again, the schedule transition is appropriate here as if the relay contacts weld during a spontaneous power surge this will only be discovered, or have any effect on the crossing at all, until a train passes. The second pair of relay contacts is modelled by transitions T-RF5-8 and places P-RF5-8 in an identical manner. In this example, only the open circuit failures are detectable, modelled by the presence of arcs from transition T-RF1/5 which will add tokens to places P-GC3 when they fire. When the relay is replaced, places P-RF1-8 are all reset to zero indicating that the relay is fully working.

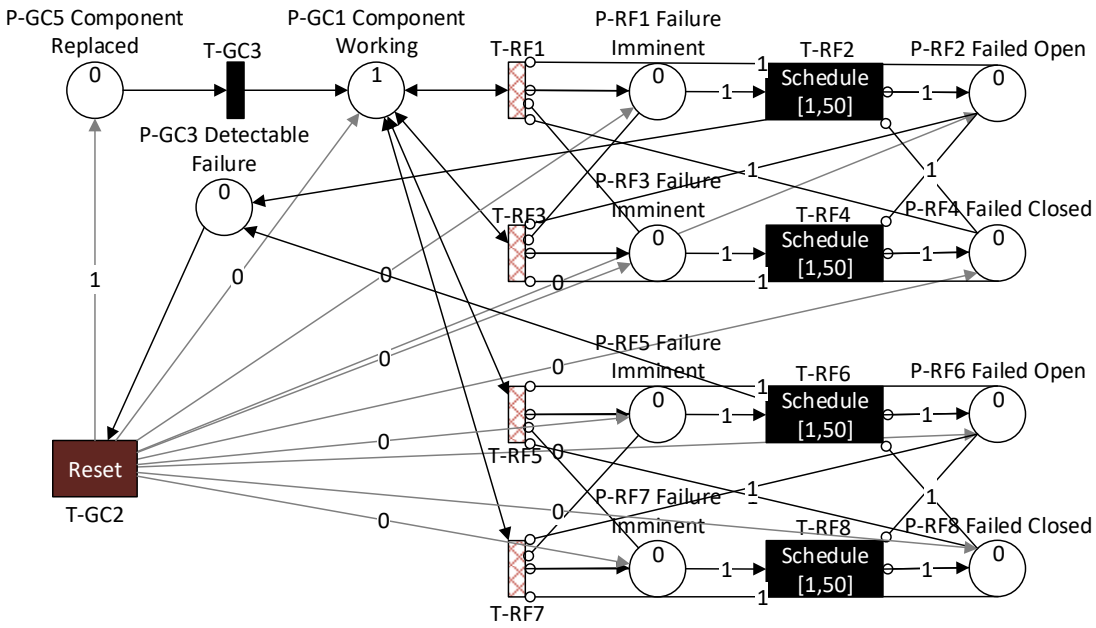

Figure 7-3 Example Relay Contact Deterioration Petri Net.

7.2.3 Capacitor Deterioration

Electrolytic capacitors lose their capacitance progressively over time due to evaporation of the electrolyte within (Hewitt & Green, 2016). This process is governed by the temperature of the capacitor, with higher temperatures forcing faster evaporation. As a capacitor loses electrolyte its resistance also increases, increasing its operating temperature, resulting in an exponentially increasing loss of capacitance. Modelling this stochastically via Petri net is challenging as Petri nets model discrete events. A continuous hybrid Petri net could be used, however this would add additional complexity. Instead Capacitor Capacitance shall be divided into five discrete states from 100% to 0%.

Figure 7-4 shows the example relay Petri net expanded to include the degradation of a capacitor. Places P-CS1/2/3/4/5 indicate the level of degradation of the capacitor. The degradation is controlled by transitions T-CS1/2/3/4. In this example, 75% capacitance is sufficient for the degradation to have an adverse effect on the crossing protection system and thus be discovered during an inspection. When T-CS1 fires a token is placed in P-CS2 indicating the capacitor has decayed to 75% of its original capacity. This enables schedule transition, T-CS5, which fires when the next train is due to pass the crossing, placing a token in P-GC6 allowing the effects of the capacitor failure to propagate to the protection system module. Also a token is placed in P-GC3, indicating there is a detectable failure. Capacitors are not a mechanical component, thus there is no reason to suspect they are more likely to fail during use. However, they can only have an effect on the crossing when it is operating, as capacitors are used in the control system to control protection system timings. The use of

the schedule transitions ensures accurate counting of the number of trains which pass whilst the protection systems are affected.

The control system comprises 35 relays, each of which is laid out in the same manner as the example shown above. For brevity, their structures have not been reproduced in this thesis.

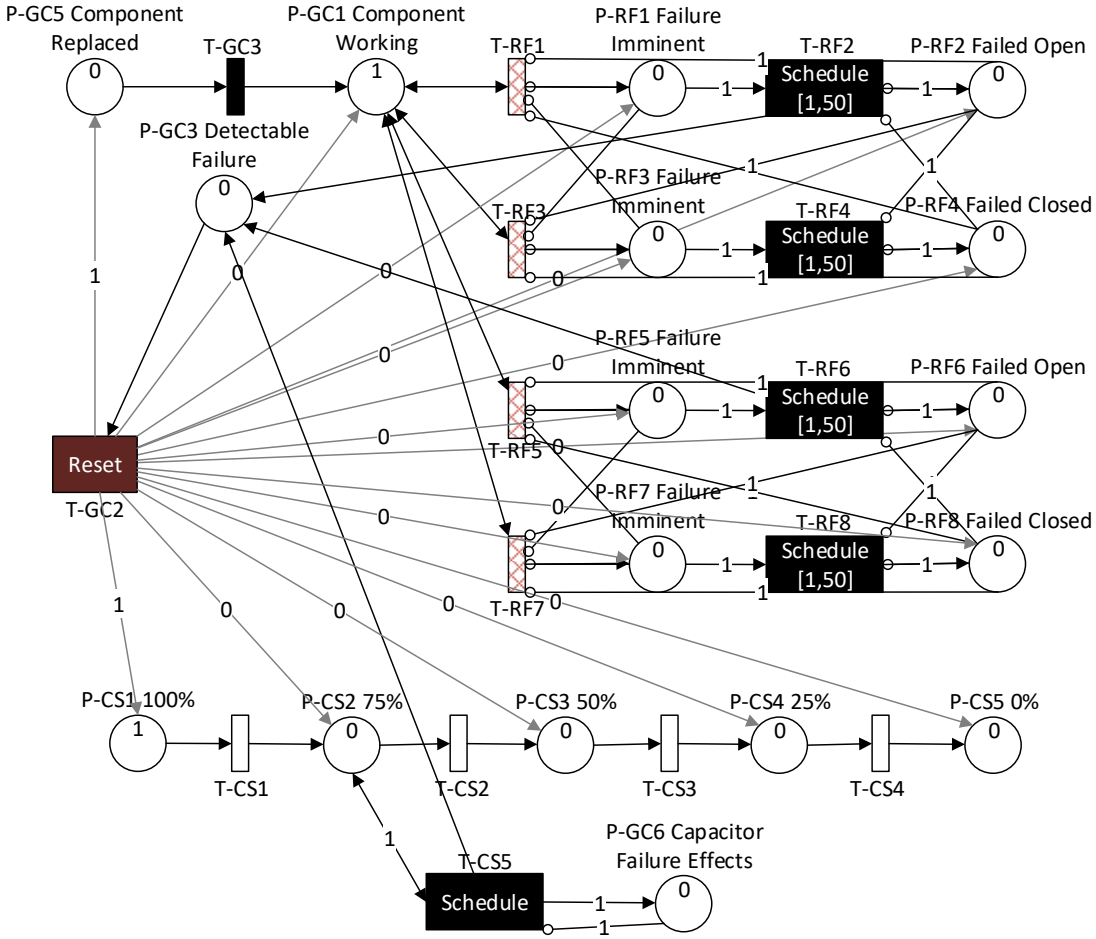

Figure 7-4 Example Relay Contact Deterioration Petri Net with Capacitor.

7.2.4 Track Circuits

Track circuit are operated by passing trains, triggering a response from a AHBCs control system activating the protection systems. As track circuits are controlled via relays, the Petri net modelling their failure and replacement is identical to that of the control system relays, using one relay contact pair per track circuit. Each rail line modelled requires 3 track circuit relay Petri nets, for the A, B and C track circuits respectively. Treadles assist track circuits by promptly cutting power to track circuits and adding additional redundancy. There are 6 treadles per rail line, each of which uses a Petri net identical to that used for relays, as the treadles are modelled as failing either open circuit or closed, exactly like a relay.

7.2.5 Road Traffic Lights

Lights are used at AHBCs to warn road drivers of the approaching train, and to warn of the imminent descent of the barriers. An AHBC crossing contains 3 lights per traffic light stalk, 1 bulb per light, and 4 stalks for a total of 12 light bulbs. The failure of each light bulb is modelled using a modified general component failure Petri net, shown Figure 7-5. This Petri net allows two failure modes, where the bulb fails either open circuit and therefore may be self-reported, and closed circuit where the failure is not self-reported. It is assumed that the closed circuit failure mode is caused via dimming of the LED. As this is a gradual process, it is not associated with the passage of a specific train and therefore does not use a schedule transition.

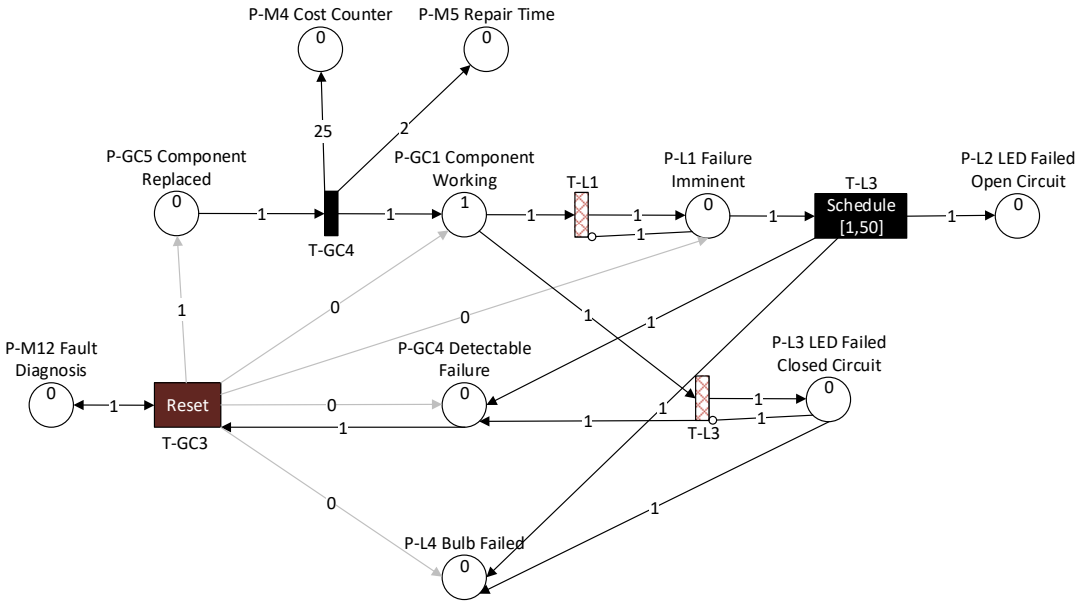

Figure 7-5 LED Bulb Failure Petri Net

Damage to fixtures and fittings are also modelled, using an unmodified general component failure Petri net.

7.2.6 Barriers

7.2.6.1 Simple Components

The following barrier components use the generalised component failure Petri Net presented earlier: Support/Structure, Bearing, Hydraulic Ram, Motor-Pump unit. Failure of these components results in the barrier being unable to rise.

7.2.6.2 Solenoid Valve

The solenoid valve which controls the return of hydraulic fluid from the ram to the reservoir may fail in either the open or closed position. The Petri net which models this is shown in

Figure 7-6. It is a modified form of the general component failure Petri net, featuring an additional mutually exclusive failure mode.

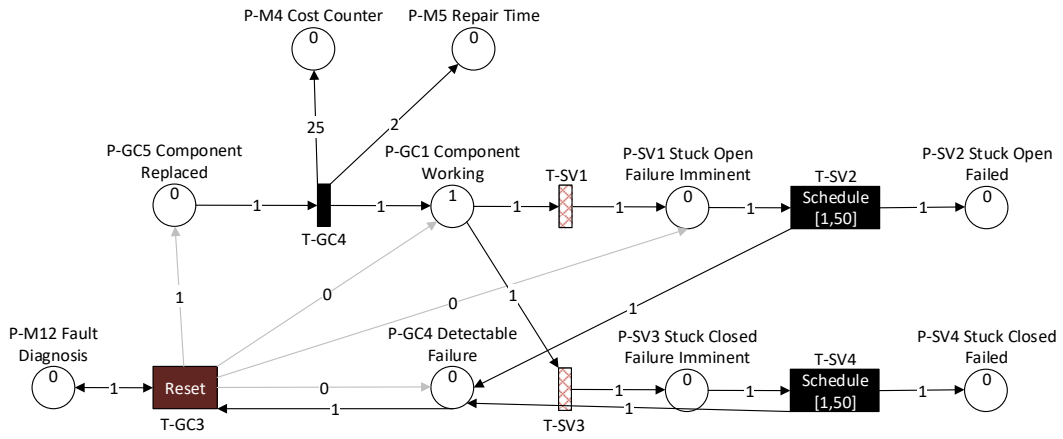

Figure 7-6 Solenoid Value Failure Petri Net.

7.2.6.3 Proving Contacts

The proving contacts report each barriers position to the control system, allowing it to self-report a failure of the barriers to operate when called for. These contacts may fail in high resistance modes only, as the design of the unit makes welding or bridging impossible. Proving contacts are a single unit (British Rail, 1991), thus failure of any single proving contact requires replacement of the entire unit. The Petri net structure for this is shown in Figure 7-7. This net is very similar to the general component Petri net presented earlier, however features four branching pathways, one for each of the proving contact. The general component failure Petri net portion has been omitted from the figure, with only the element failure structures shown.

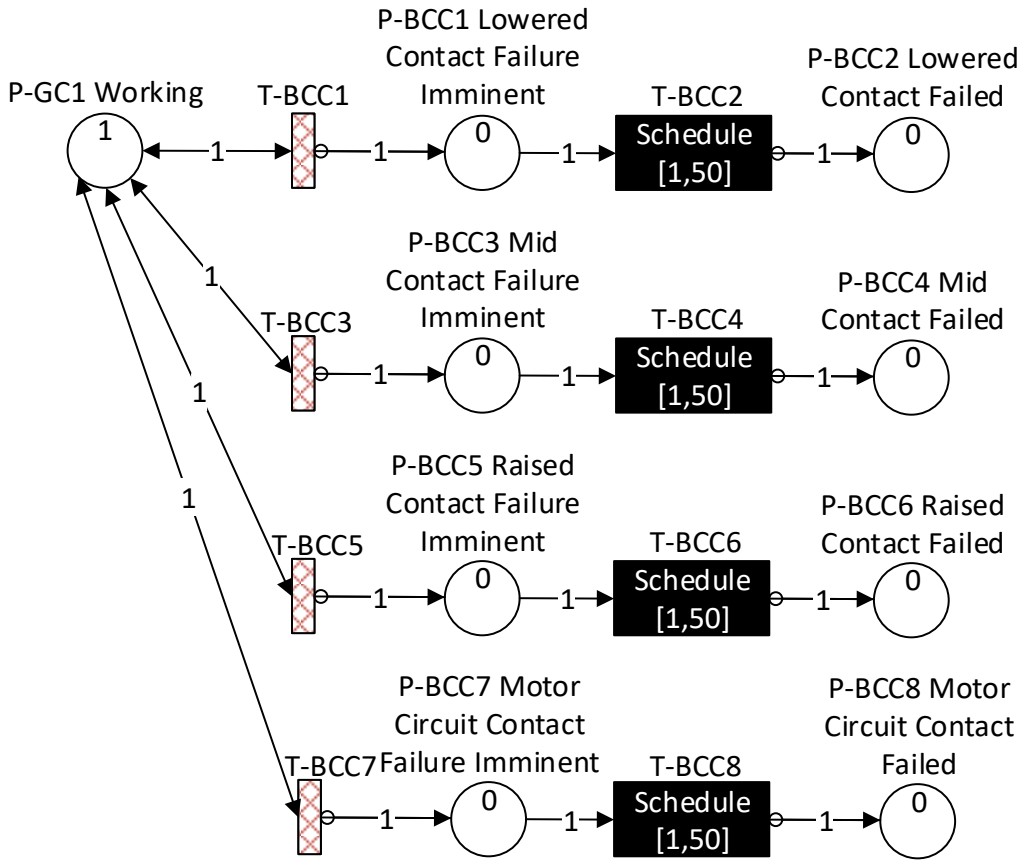

Figure 7-7 Barrier Contact Failure Petri Net

7.2.7 Cabin Petri Net

The equipment cabin keeps the vulnerable control system in a controlled environment and prevents tampering/theft. The design of the cabin failure Petri Net differs from the generalised component Petri Net structure somewhat, the cabin may be repaired no more than twice before the repairs become ineffective and the entire cabin must be renewed, requiring modifications to the generalised Petri Net. This is in keeping with the semi temporary nature of these structures.

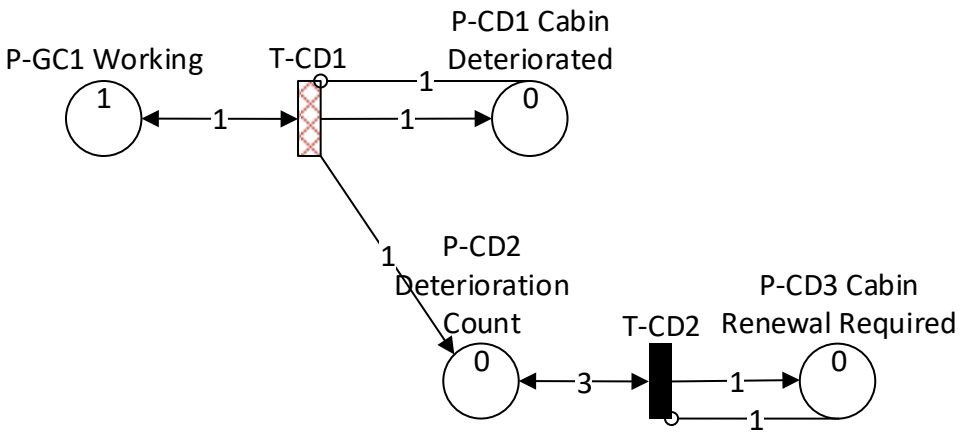

Figure 7-8 Cabin Deterioration Petri Net.

Examining Figure 7-8, Deterioration of the cabin is controlled by stochastic transition T-CD1, when this transition fires a token is placed in place P-CD1 indicating the cabin has deteriorated and requires refurbishment. Each time the cabin fails a token is also added to place P-CD2. When the number of tokens in P-CD2 is equal to 3, T-CD1 is enabled, fires immediately, and a single token is placed in place P-CD3. This token schedules an immediate renewal of the cabin which occurs in a Petri net shown later. This prevents more than 2 repairs of the cabin before the cabin is replaced via a renewal.

7.2.8 Power Supply

The power supply provides DC power to all the level crossing's electrical systems. It comprises an AC to DC power supply, circuit breaker and back-up batteries. The back-up batteries are specified to be sufficient for 8hrs of crossing operation using deep discharge batteries. The number of batteries will vary with capacity and the frequency of passing trains, for this model it has been assumed 24 individual batteries are required. Each battery, and the other power supply equipment, is modelled using the general component failure Petri net outlined earlier. Failure of the power supply or a power cut will result in the back-up batteries engaging. Once the batteries have fully discharged the road barriers will descend under gravity, but road traffic lights will remain dark until power is restored. A circuit breaker may trip spuriously and require an engineer to reset, again, modelled using the general component failure Petri net.

7.2.8.1 Back Up Battery Failure

The model features 25 back up batteries, however the number at an actual crossing will vary depending on the frequency of passing trains and the size of the batteries used. Batteries are modelled with two states, working and deteriorated. When deteriorated, batteries do not contribute any back-up power. Batteries fail individually and but are replaced collectively. This requires a slightly different Petri net structure to those presented so far, shown in Figure 7-9.

This Petri net shares many features with the general component failure Petri net. It begins with a token in place P-PW2 indicating that the battery is working. This enables transition T-PW2 which stochastically models the time till the battery deteriorates. When this transition fires a token is added to P-PW3 indicating the battery has deteriorated, and a further token added to place P-PW4. This place counts the total number of battery presently deteriorated.

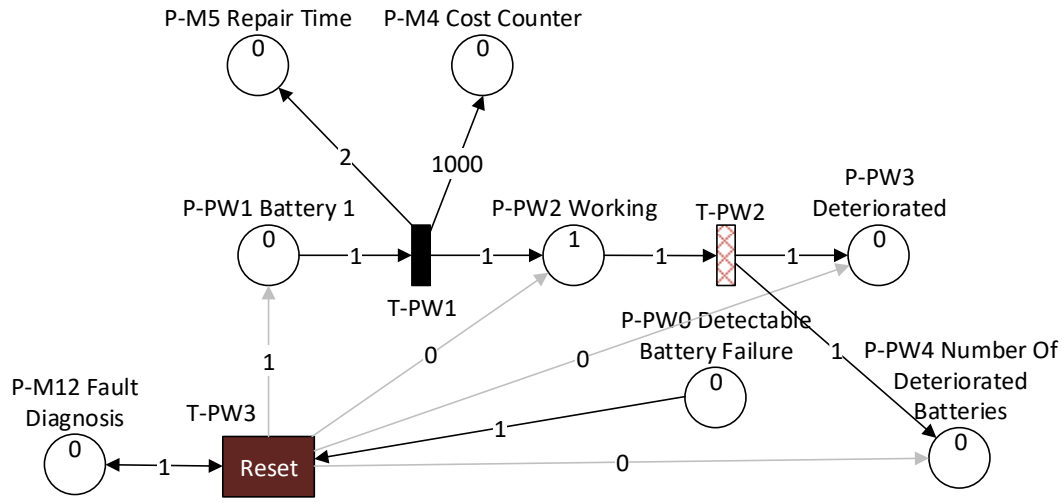

Figure 7-9 Back Up Battery Petri Net Structure.

The guidance for inspecting automatic level crossing makes no provision for checking the state of the back-up batteries (Network Rail, 2004), therefore battery failures can only be discovered when they fail to operate during a power cut, the Petri nets for this are described next. When a back-up battery failure becomes discoverable a token is placed in place P-PW0, enabling transition T-PW3 to reset all the back-up battery Petri nets to model replacement of the entire battery back-up.

7.2.8.2 Power Cuts

The power cut Petri Net simulates the occurrence of mains electricity outages, triggering the battery back-up to engage. This keeps the crossing systems powered till mains electricity is restored or the back-up battery runs flat. Figure 7-10 shows the Petri Net which initiates power cuts. A token in place P-PW5 indicates that the mains power supply is normal, whilst a token in place P-PW6 indicates there is a power cut.

Transition T-PW4 models the occurrence of power cuts, enabled by place P-PW5. This transition features an exponentially distributed delay modelling the constant probably of a mains power cut occurring. When T-PW4 fires, it removes a token from P-PW5 and adds two tokens to P-PW6. The second token enables a Petri net which models battery discharge. Transition T-PW5 models the resolution of power cuts, using another exponentially distributed random delay time. It is enabled by a token in place P-PW6. When T-PW5 fires it removes a single token from P-PW6 and adds a single token to P-PW7, representing that the power cut has been resolved. The last transition in the net, T-PW6 resets the entire net to its original state after the power cut has resolved. This must be a reset transition due to the possibility of a second token in P-PW6.

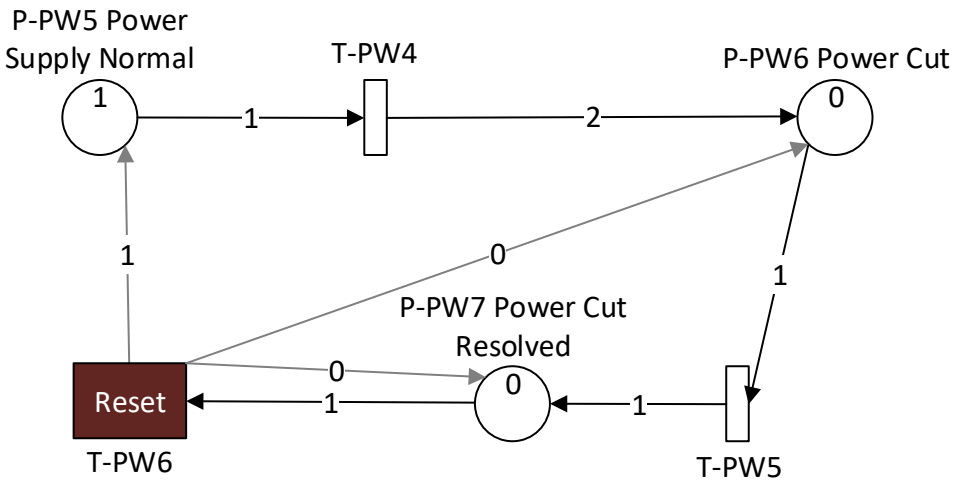

Figure 7-10 Power Cut Petri Net.

Mains power cuts are not the only circumstances in which the battery back-up will be engaged. Failure of the power supply equipment within the crossing which converts mains AC electricity to lower voltage DC will force power to be drawn from the battery back-up. The Petri net modelling this is shown in Figure 7-11. This is very similar to the general component failure Petri net. T-PW11 uses a stochastic delay time to model failure of the power supply equipment, when it fires it removes a token from P-PW9 and places a token in P-PW10 indicating that the equipment has failed. It also places a token in place P-PW6 indicating that a power cut has taken place, an inhibitor arc from place P-PW10 to transition T-PW5 prevents this from resolving forcing the power supply failure to be resolved first. A similar Petri net models spurious circuit breaker tripping.

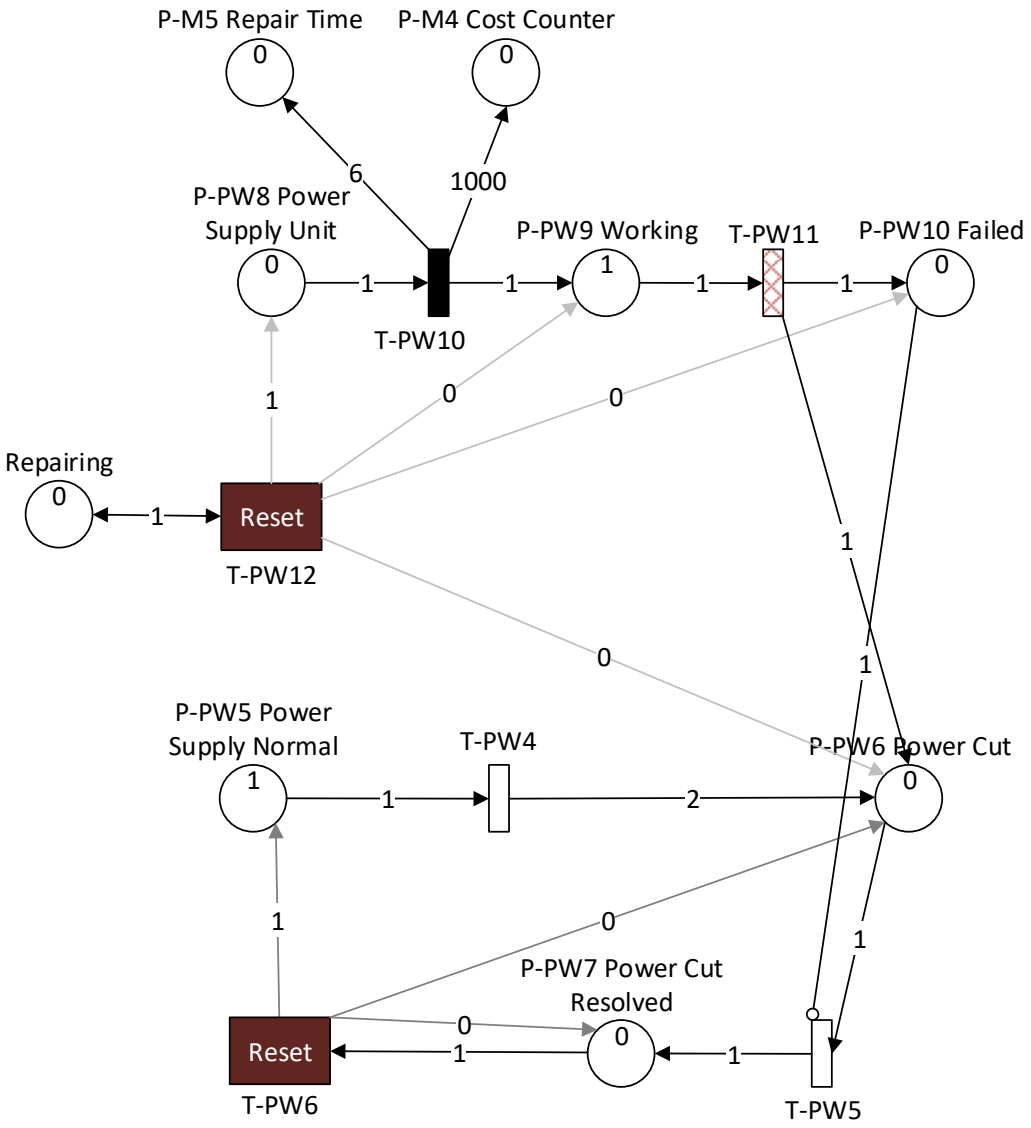

Figure 7-11 Rectifier Failure Petri Net

Once a power cut has occurred the Petri Net in Figure 7-12 is enabled, modelling the discharge of the back-up batteries. If all batteries are functioning the back-up battery time is 8hrs, losing 20minutes of back-up time for each battery that has failed. This is modelled using transition T-PW13, a conditional delay transition. This transition's delay is set at 0.333days (8hrs), losing 0.0133days (20minutes) for every battery that has failed, facilitated by a conditional arc from place P-PW4. When transition T-PW13 fires, a token is placed in P-PW11 indicating that the crossing is no longer powered. An inhibitor arc from this place prevents the transition from firing a second time. Spontaneous resolution of the power cut by transition T-PW5 & T-PW6 firing will remove all tokens from P-PW6, disabling transition T-PW13.

A second conditional delay transition, T-PW14, models the scenario where due to excessive battery failure, the crossing back up lasts less than 5hours. In this example, this requires more than 10 of the 25 batteries to have failed. Once 10batteries have failed, an inhibitor arc from P-PW4 disables transition T-PW13, and a test arc enables T-PW14. Should the power cut not be resolved prior to T-PW14 firing, when this transition fires tokens are placed in both place P-PW11 indicating complete power failure, and in place P-PW3 allowing the failed batteries to be discovered and repairs scheduled. Finally, a complete power failure due to discharged batteries will be resolved by transition T-PW15 once the mains power supply is restored.

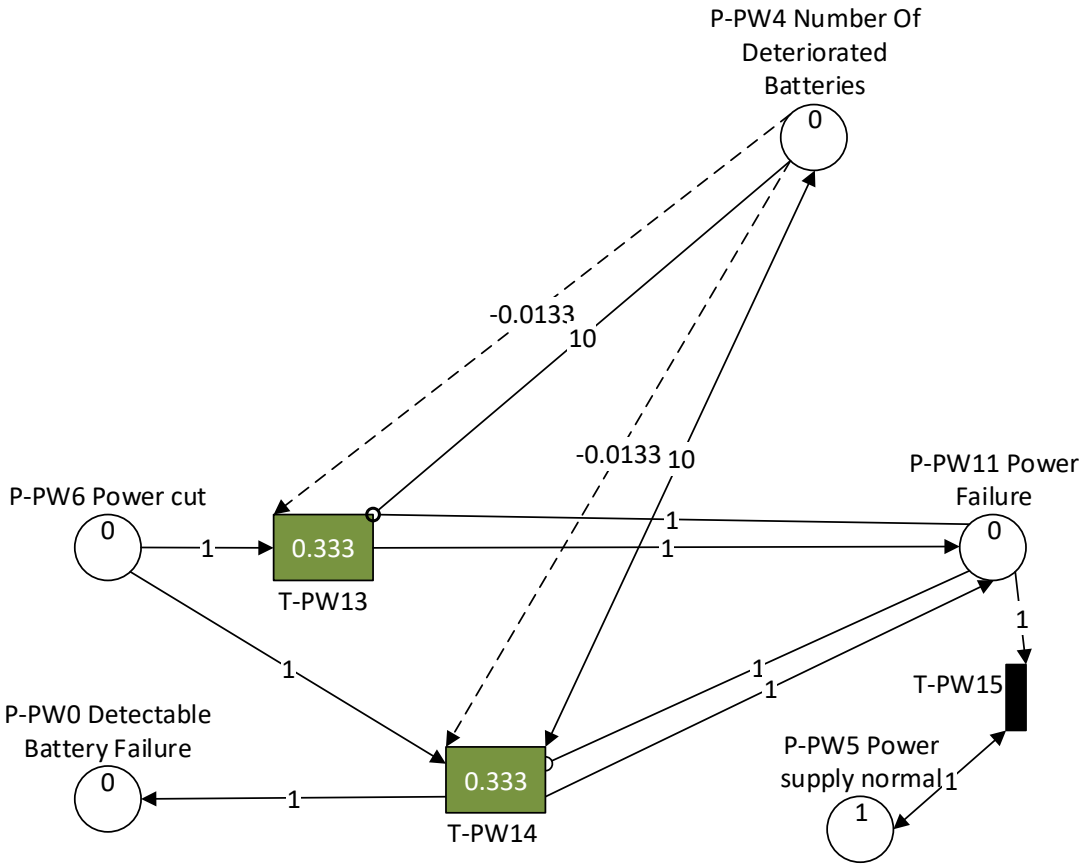

Figure 7-12 Back-Up Battery Discharge Petri Net.

7.2.9 Road Surface Blocks

Road surfaces come in many different forms at level crossings, in this work, the modern modular type of road surface has been explored. 66 individually replaceable road surface blocks have been modelled. It has been assumed that the degradation of these blocks is not hazardous, therefore their repair is modelled as routine, and not carried out unless five or more blocks degrade. A Petri net for a single road surface block can be seen in Figure 7-13.

Like others shown in this chapter, this is based on the general component failure Petri net. There are some key additions for the purpose of modelling road surface deterioration

however. Whenever transition T-RB1 fires, modelling the deterioration of a road surface block, a token is added to place P-RB3 which holds tokens equal to the current number of deteriorated blocks. When this place holds more than five tokens, a routine work order is generated to repair the road surface, shown later. The reset transition which replaces road surface blocks must be enabled by a token in the Routine Work Order place. This is because road surface repair requires planning and equipment which would not generally be on hand during an urgent repair.

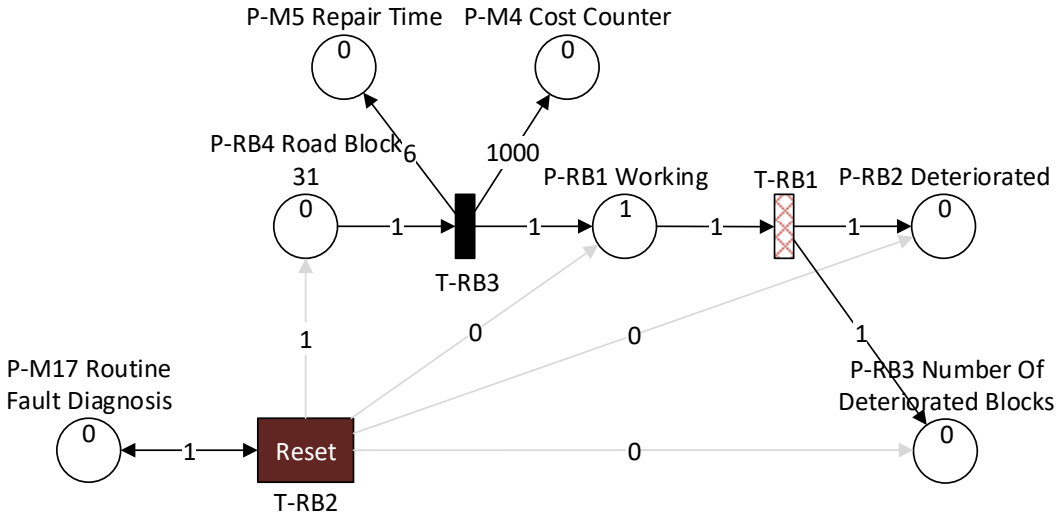

Figure 7-13 Road Block Petri Net

7.2.10 Equipment Renewal Module

The asset management model has the facility to model the periodic replacement of entire systems, either as scheduled maintenance actions, or as opportunity presents. Renewals are controlled using work orders, Figure 7-14 shows the generation and fulfilment net for control system renewals. A token in place P-R3 enables scheduled renewals, and a token in place P-R1 enables opportunistic renewals. Transition T-R1 fires with a delay modelling the time till an opportunistic renewal would be considered. When T-R1 fires a token is removed from P-R1 and a token added to P-R2, representing the creation of an opportunistic renewal work order for the control system. This enables transition T-R3, which fires with an exponentially distributed delay, modelling the time till the work order is fulfilled. An exponential distribution was used here as opportunistic work orders are low priority, occurring only when maintenance staff are available. The availability of maintenance staff is assumed to have a constant probably of occurrence over time.

Transition T-R2 models the time till a renewal is scheduled in a similar manner to opportunistic renewals described above. The time till a scheduled renewal occurs is modelled using a transition with a log-normal delay, transition T-R4. When either T-R3 or T-R4 fires,

indicating a renewal is beginning, a token is added to place P-R5. This place is then able to enable transition T-5, which fires immediately setting all previously mentioned places to zero. This is done to prevent an opportunistic renewal beginning whilst a scheduled renewal is occurring or vice versa. Finally, transition T-R5 places a token in place P-R6 allowing the renewal to progress.

In the later framework chapter, the time till the occurrence of either an opportunistic or scheduled renewal will not be determined randomly. It will occur on the basis of availability of staff and the priority of other work orders present at other crossings.

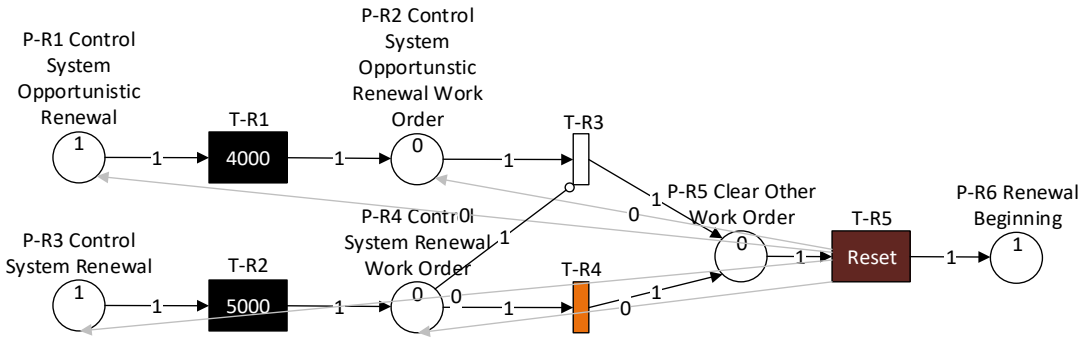

Figure 7-14 Start of Control System Renewal Petri Net.

The next section of the renewal Petri Net ensures that the road is closed to road traffic, if it is not already closed when renewal begins, shown Figure 7-15. This is achieved with two instant transitions which both remove a token from P-R6 and place a token in P-R7 allowing the renewal to progress. Transition T-R6 is inhibited by a token in P-M3 (indicating the road is open), which enables T-R7 which fires removing the token from P-M3. If the road is already closed, transition T-R6 will fire.

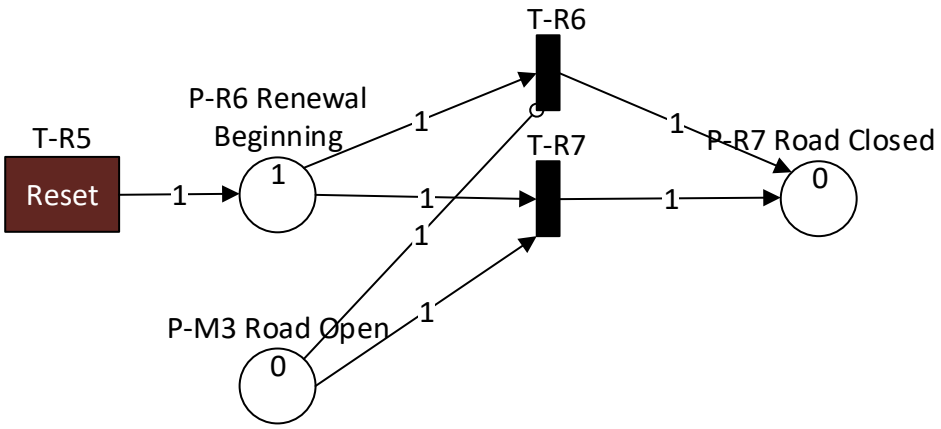

Figure 7-15 Petri Net Ensuring the Road is Closed during Barrier Renewal.

The final section of the renewal Petri Net is shown in Figure 7-16, this section of the Petri Net completes the actual renewal by first removing all the tokens from the component failure

Petri Nets for the system using transition T-R8. This transition also removes a token from P-R7 and places a token in P-R8, enabling transition T-R9. This transition has a delay which models the time required to complete the renewal, 6hrs in this example. When T-R9 fires it the removes the token from P-R8 and places it in P-R9 signifying that the renewal has finished. This enables the last transition, T-R10, this reset transition restores all the tokens in the replaced system's component failure Petri nets to a working state. This bypasses the reset transitions for each individual component as these are for urgent or routine repairs only. Finally, Transition T-R10 also adds a token to place P-M4, and resets the other inspection related places, forcing an inspection to occur prior to the repair crew leaving.

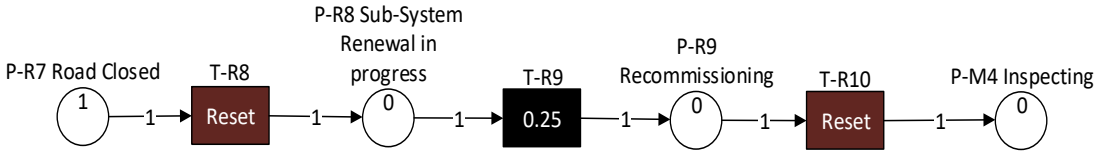

Figure 7-16 Final Section of Barrier Renewal Petri Net.

The example shown above was for renewal of the control system. The model features renewals for all level crossing systems, the timings of which are shown in Table 7-1.

Table 7-1 Renewal Parameters

System	Opportunistic	Scheduled	Duration
Control System	4000days	5000days	6hrs
Track Circuits	4000days	5000days	6hrs
Barriers	7000days	8000days	6hrs
Lights	7000days	8000days	6hrs
Road Surface	7000days	8000days	6hrs
Cabin	7000days	8000days	6hrs
Power Supply	7000days	8000days	6hrs

7.2.11 Protection System Failure States

This portion of the model tracks which protection system failure modes are currently occurring and counts how many trains pass the crossing whilst in the failed state(s). The frequency of these protection system failure events is important to asset management decision making as some significantly increase the likelihood of a rail – road vehicle collision, explored in chapter 5. The failures states which the model tracks are listed in the table below. Each of the states has both its own place indicating if that state is currently active, and a

second place which counts how many trains have passed the crossing whilst that failed state was active.

Table 7-2 Protection System States Tracked by Model

Failed State	Failed State
Y side lights off when train passing	Z side lights off when train passing
Y side barrier late	Z side barrier late
Y side lights stuck on	Z side lights stuck on
Y side barrier up when train passing	Z side barrier up when train passing
Y lights off and barriers up when train passes	Z lights off and barriers up when train passes
Y side barrier stuck down	Z side barrier stuck down

Figure 7-17 shows the Petri net structure which links each component failure net to the protection system failure states, which the component failure causes. P-GC3 links in from the generalised component failure Petri net. There is a single token in place P-GC3 indicating the component has failed. This enables transitions T-PS1, T-PS2 and T-PS3 via test arc, which fire instantly, placing tokens in places P-PS1, P-PS2 and P-PS3 respectively. In this example, this indicates that the component failure causes both sets of lights to remain stuck illuminated, and that the fault is self-reported. Inhibitor arcs prevent these transitions firing again, or if there are already tokens in these places. Use of test arcs allows the model to retain the information on which component has failed, whilst also enabling the appropriate failure states. The failure state places shown and the others not shown but listed in the above table appear only once in the model, and are shared between all components.

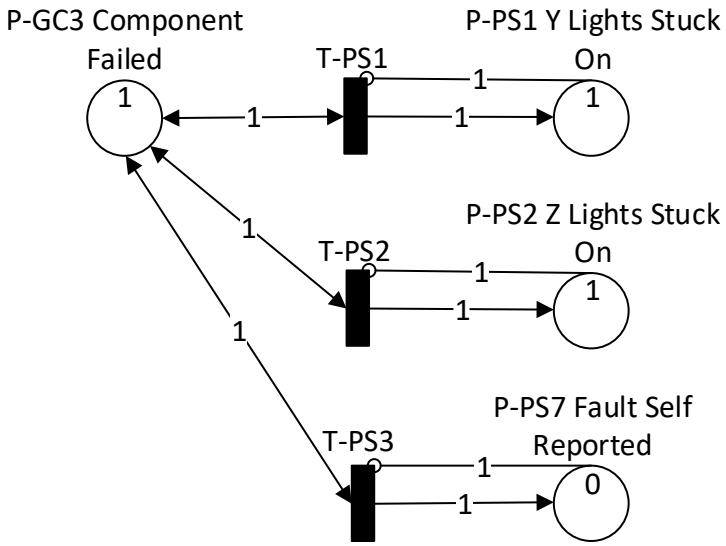

Figure 7-17 Petri Net Mapping Single Component Failure to Effects.

This design enables Protection system states to be stored independently of the components which have failed, greatly reducing the overall number of states the model must store. After the model completes any action which might resolve a protection system fault, either renewing a system or repairing a failed component, all the protection system failure places are reset to zero tokens. The model does not know if repairing any given component or renewing any given system will restore crossing function.

Should a failed component persist after either action, tokens will immediately propagate back into the appropriate protection system failure places. Whilst it is possible for the model to store information on which component was responsible for a given protection system failure for a single level crossing model, this requires many additional places and transitions per component, and the associated computational burden.

The fault trees presented in chapter 6 were implemented using the same methodology as that used to simulate the control system logic in the same chapter. Each AND and OR gate in the fault tree uses the Petri net module described in that chapter. A small example branch of the fault tree for failures of the road traffic lights is shown in Figure 7-18. This models the scenario where both relay contacts RER1 and RER2 fail high resistance preventing the road traffic lights from illuminating. Here Places P-FT1-2 and transitions T-FT1-3 form an AND gate.

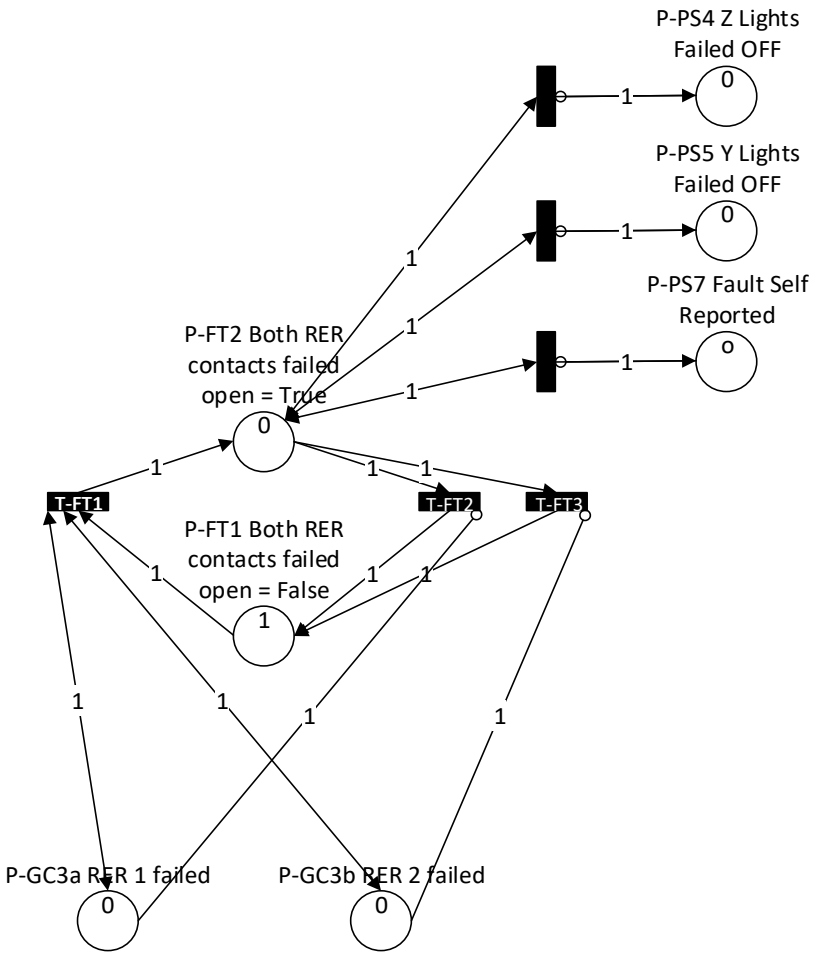

Figure 7-18 Example of Fault Tree Petri Net

The number of trains which pass whilst an undiscovered protection system fault is occurring are counted. It is assumed that once the protection system fault is discovered, any subsequent trains passing the crossing are instructed to 'pass at danger', meaning they reduce their speed sufficiently that collision or injury is unlikely to occur as the train passes the crossing. This is recorded as a delayed train.

An example Petri net modelling this is shown in Figure 7-19. This net counts the number of trains which pass whilst the Z lights have become illuminated in the absence of any approaching train. A token in place P-PS3 enables the schedule transition, T-C1. This transition also requires a token in place P-M3, indicating that the crossing is open to road traffic, to be enabled. The transition fires with a delay equal to the time till the next train passes. The presence of an urgent work order inhibits this transition, as this means the fault has been discovered. This enables transition T-C2, which counts the number of trains which are delayed in a similar fashion to transition T-C1.

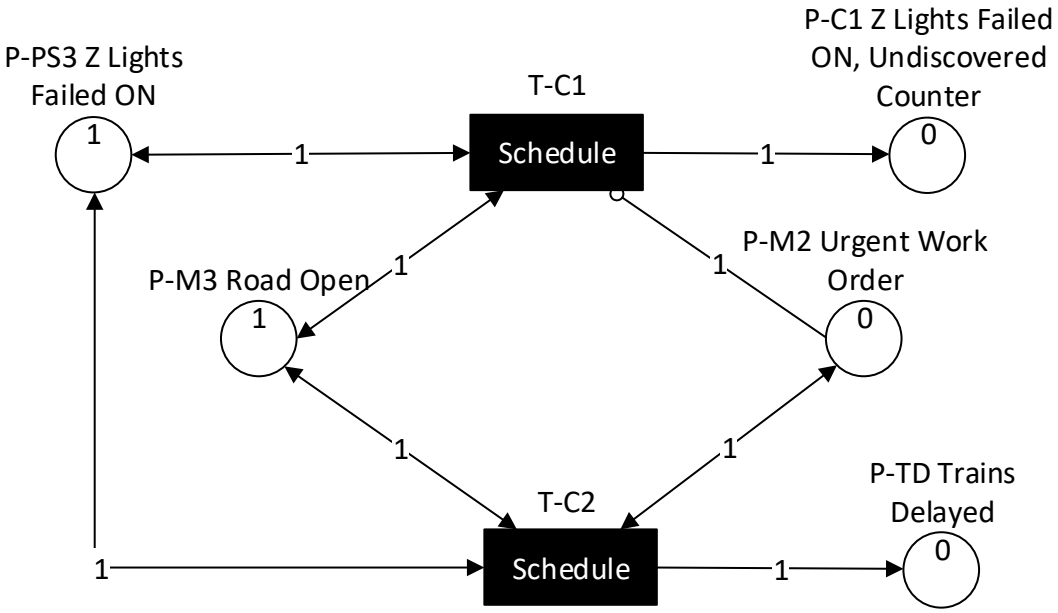

Figure 7-19 Dangerous Train Passage and Delay Counter

7.2.12 Protection System Failure - Self Reporting

A failed component must be discovered before it can be repaired outside of a renewal. In this model, only component failures which have an effect on the protection system, or are self-reported can be discovered and repaired as part of an unplanned maintenance action. Inspecting engineers may note protection system irregularities as part of their inspection routine.

Mostly commonly though, faults are self-reported by the level crossing's control system, Figure 7-20 shows the Petri net which performs this action. Place P-PS7, which signifies that the crossing is reporting a fault, enables transition T-M1 via test arc. After the transition fires a token is added to place P-M1, indicating that signalling engineers are now aware of the fault, and a repair crew will be dispatched when available. A second token is added to place P-M2 which prevents a second concurrent urgent work order being created.

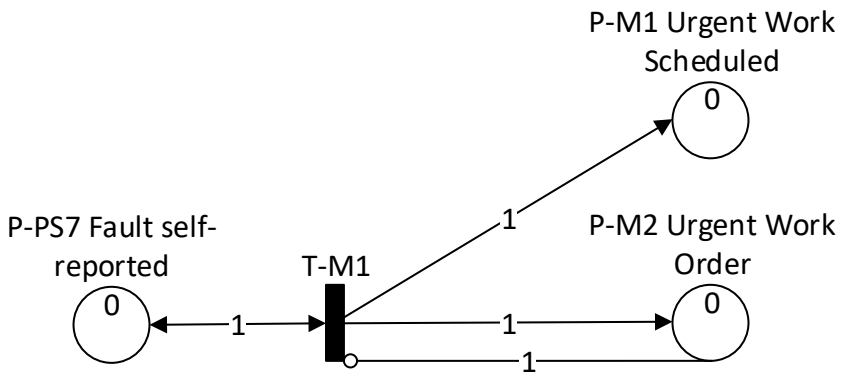

Figure 7-20 Petri Net for Self-Reporting Faults

7.2.13 Inspections

Automatic Half Barrier Crossings must be regularly inspected by Network Rail, with a maximum duration of 6 weeks between inspections. These inspections are brief, comprising a simple visual inspection and verification of the function and timing of protection systems (Network Rail, 2004). These inspections are important, as there are some failure states that can only be detected during inspection including: Cabin deterioration, road surface deterioration, incorrect protection system timings, vegetation overgrowth and partial light failures.

Figure 7-21 shows the Petri net which models periodic inspection of the crossing. The inspection net begins with place P-M5, a token in this place enables transition T-M3 whose delay models the time till an opportunistic inspection work order is created. When T-M3 fires it the removes the token from P-M5 and places a token in P-M6. Now an opportunistic inspection is considered, transition T-M5 is enabled, this exponentially distributed delay transition models the constant probability of a repair crew being available to opportunistically inspect the crossing. If an opportunistic inspection does not occur within 14 days, transition T-M4 fires. This removes the token from place P-M6, disabling transition T-M5 and places a token in place P-M7, enabling transition T-M6. This transition uses a log-normal delay to model the time till a scheduled inspection occurs. When either T-M5 or T-M6 fire a token is removed from either P-M6 or P-M7 and added to place P-M4. This indicates that an inspection is now occurring. The inspection takes 30 minutes, modelled by transition T-M2. When this transition fires the inspection is over and the cycle repeats. In a later framework chapter both stochastic transitions are replaced with a deterministic scheduling system.

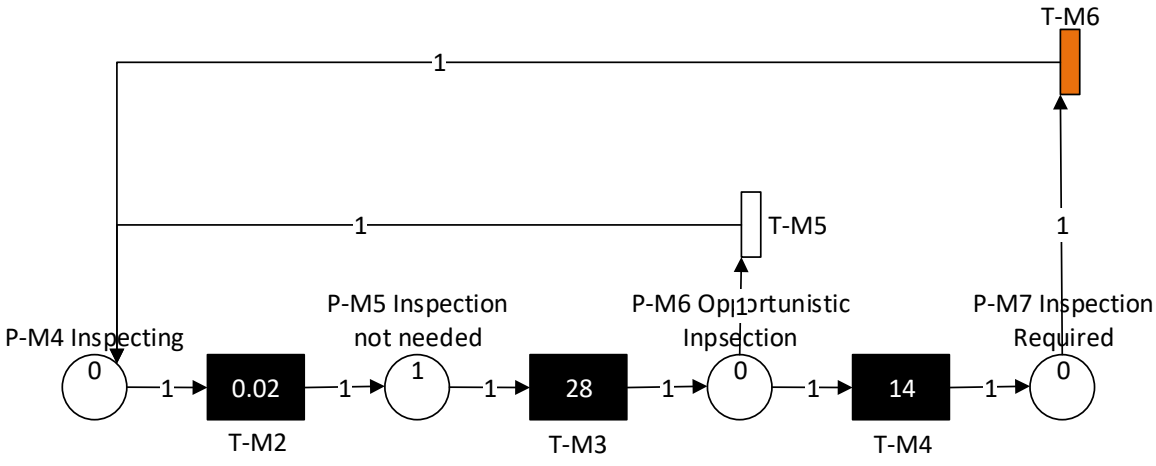

Figure 7-21 Inspection Petri Net.

The inspection process may discover hazardous protection system failures that were not self-reported by the crossing. An example for detecting failure of the Z side lights is shown in Figure 7-22. If the Z lights have failed on a token will be present in P-PS3, this will enable T-M7 via test arc. This transition fires immediately, placing a token in place P-M8, this place indicates there is a protection system failure waiting to be discovered. Should the Z lights have failed off, P-PS4 and T-M8 also place a token in P-M8 in a similar manner. All places representing protection system malfunctions will add a token to place P-M8, though not shown for brevity.

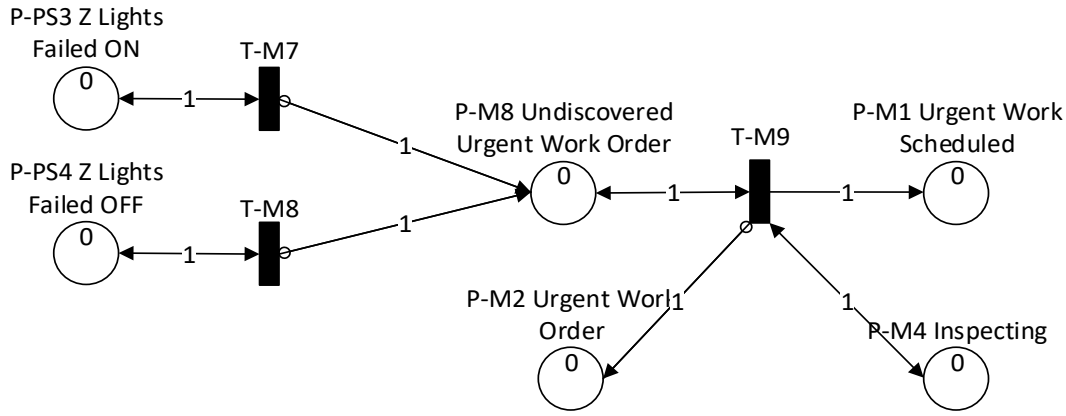

Figure 7-22 Discovery of Hazardous Protection system States via Inspection

Transition T-M9 controls the scheduling of an urgent work order during an inspection. It is enabled by tokens in places P-M8 and P-M4 via test arc. It may be inhibited by a token in place P-M2, preventing concurrent urgent work orders at the same crossing. When T-M9 fires it places a token in P-M1 representing an urgent work order, and another token in P-M2.

A Petri net with an identical structure handles the discovery of routine failures, shown Figure 7-23. Deterioration of the cabin triggers a routine work order, as does five or more deteriorated road blocks.

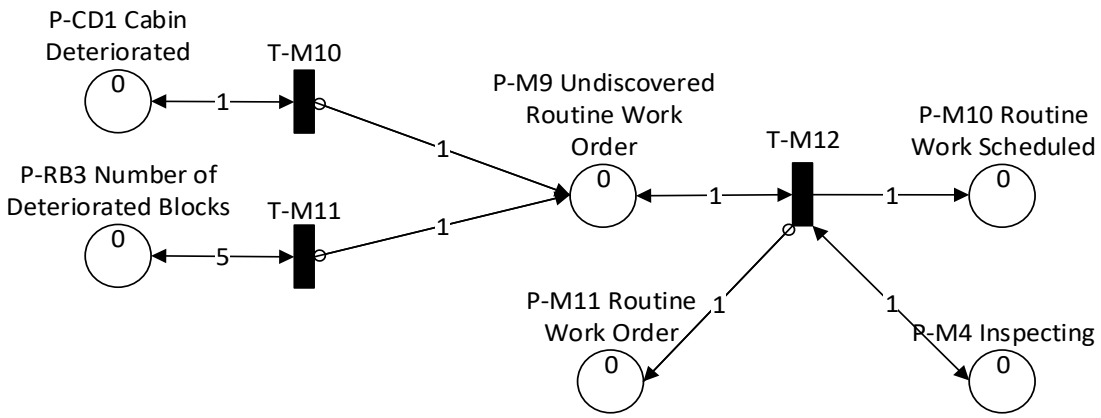

Figure 7-23 Discovery of Routine Failure States via Inspection

7.2.14 Unplanned Maintenance

Level crossings require unplanned maintenance to correct randomly occurring component failures. Figure 7-24 shows the first part of the Unplanned Maintenance Petri Net, controlling mobilisation of the repair crew. The presence of a token within place P-M1 indicates an urgent repair work order has been created for the crossing. The token within this place enables transition T-M14, which uses a log-normally distributed random variable to model the time till the work order is acted upon. After T-M14 fires it removes the token from P-M1 and places a token in P-M11 indicating that the work crew has arrived.

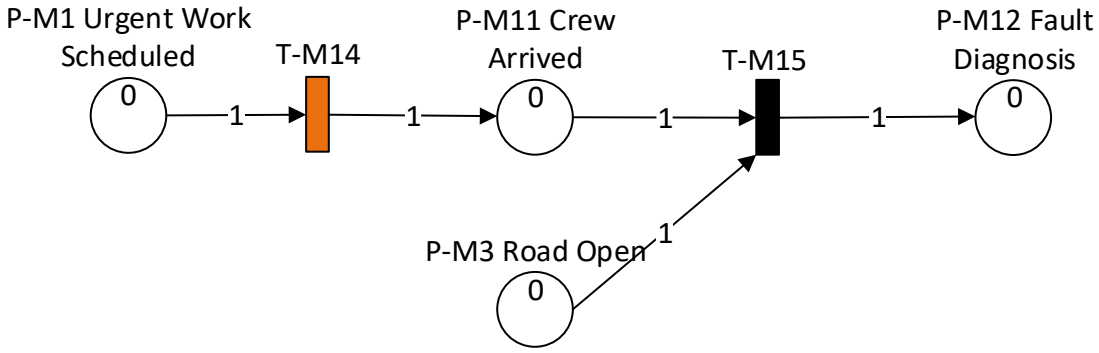

Figure 7-24 Unplanned Maintenance Petri Net, Part 1.

Upon arrival the first task of the repair crew is to close the crossing to road traffic. This is achieved with transition T-M15, which is enabled by tokens in places P-M11 and P-M3, removing tokens from both when it fires and adding a token to place P-M12. Indicating that the repair crew have begun to diagnose the fault. Recall the component failure Petri nets presented earlier, the token in place P-M12 is what finally enables the reset functions which repair each discoverable failed component, adding the repair time to place P-M14 to determine the total repair time required.

The second part of the Unplanned Maintenance Petri Net controls the repair process, shown Figure 7-25. Transition T-M16's delay models the time for the faults to be diagnosed, set at 30mins. When this transition fires the token is removed from P-M12 and placed in P-M13. The next few actions of the net control the length of time the repair rakes. The tokens in place P-M14 stop the reset transition T-M18 firing and completing the repair process. Transition T-M17 is a reset transition with a conditional delay which removes the tokens from place P-M14 and allows the inspection to complete. The delay is conditional on the number of tokens within P-M14, at 10 minutes per token. This models the specific time required to repair the failed components. When T-M17 fires, transition T-M18 is no longer inhibited and it too fires, resetting its work orders, the inspection Petri net, and the protection system states. Finally a token is added to place P-M4 triggering an inspection.

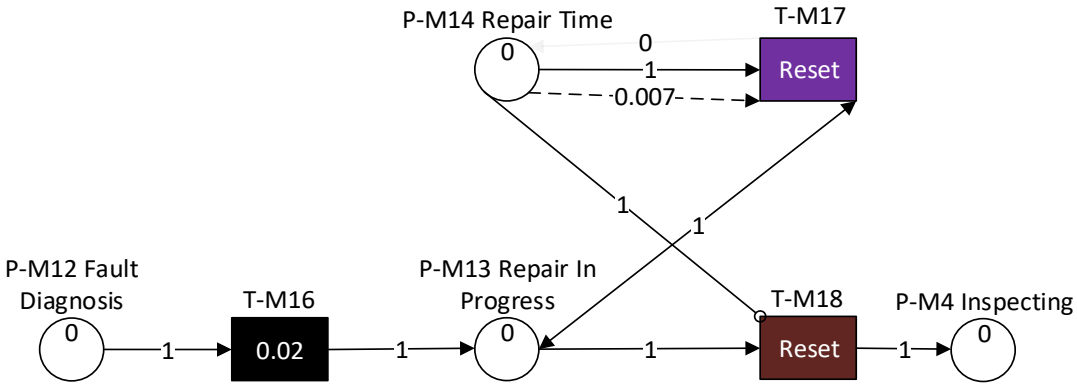

Figure 7-25 Unplanned Maintenance Petri Net, Part 2.

These Petri nets are duplicated to model the routine/scheduled maintenance processes.

7.2.15 Asset Management Model Output

The model outputs a variety of data to enable asset management decision support. The purpose of these outputs, and example results of the model are presented below.

7.2.15.1 Equipment Failure Rates

For each year of operation, the model outputs the number of failures of each system which comprises the level crossing, but also whether the effect of the failure was a fail-safe scenario or whether the failure created conditions hazardous to rail and road vehicles. This information estimates how frequently emergency repairs will be required, which systems are most prone to failure, and how those systems tend to fail. This is useful when determining the availability of emergency maintenance staff/equipment, but also when considering the frequency at which equipment should be renewed to minimise failures. It may also be useful when considering if a system design is sufficient for the crossing site and the level of performance required.

The failure rate of the control system is shown in Figure 7-26. Owing to the design of the control system, and the weld resistant relay contacts, the most common failures do not have hazardous consequences, with these occurring around 100 times less often than safe failures. There is a cyclic change in failure frequency, corresponding to the age of the system. The Weibull distribution and parameters used create a failure rate for the system that increases with age. Failure frequency increases for the first 10 years of operation. Opportunistic renewals begin at 11 years and scheduled renewals at 13 years, this dramatically reduces the failure rate of the system.

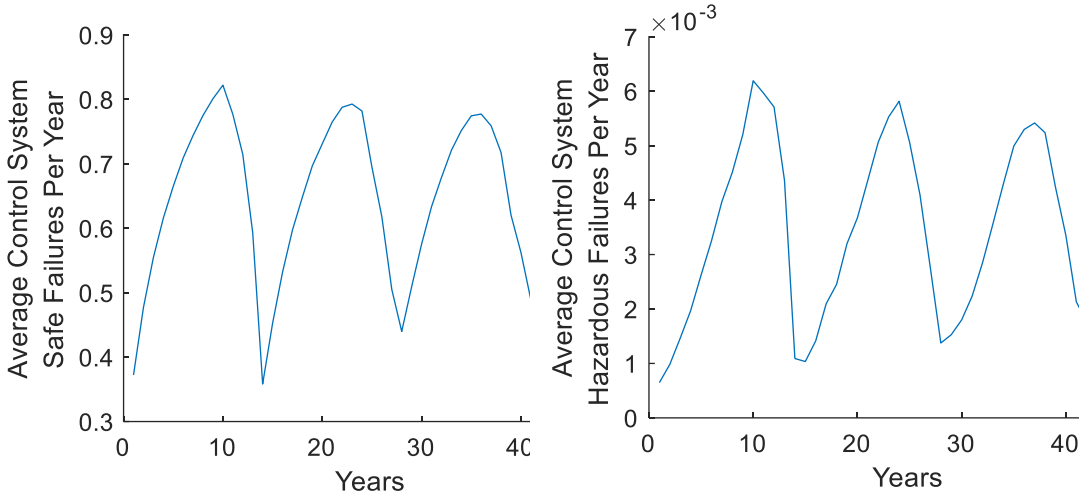

Figure 7-26 Control System Failures with Periodic Renewals. Note: scale is different for each graph.

The peaks and troughs of the failure rates reduce with each renewal cycle. This is because renewals occur stochastically. This effect is more pronounced when viewing the troughs, as the low failure rate of recently renewed systems is raised by unrenewed old systems with high failure rates. Figure 7-27 shows when control system renewals occur, it can been seen each group of renewals occurs over a progressively larger time period. This results in failure rates converging to a fixed value over time. This is consistent with the results of other authors who have used Petri nets to model rule based maintenance (Le, 2014).

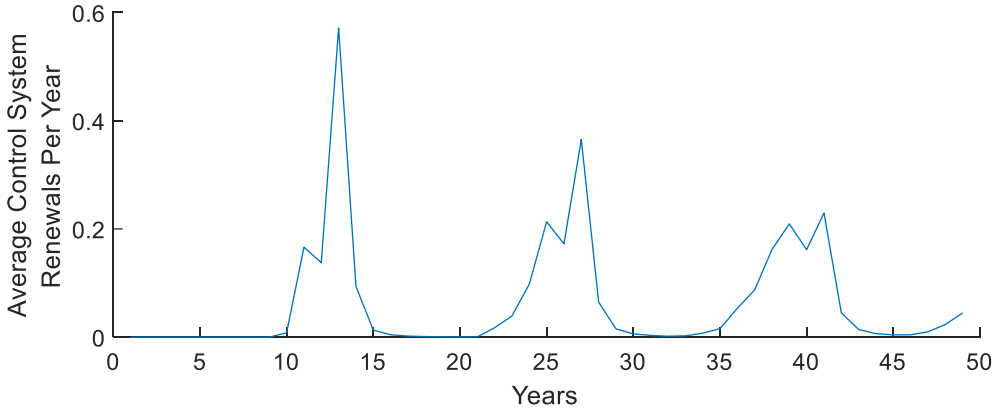

Figure 7-27 Control System Renewals

Deactivating periodic renewals of the control system results in the failure rate reaching an equilibrium, shown in Figure 7-28. As components fail they are replaced with new, overtime the average age of a control system component stabilises corresponding to a stable failure rate.

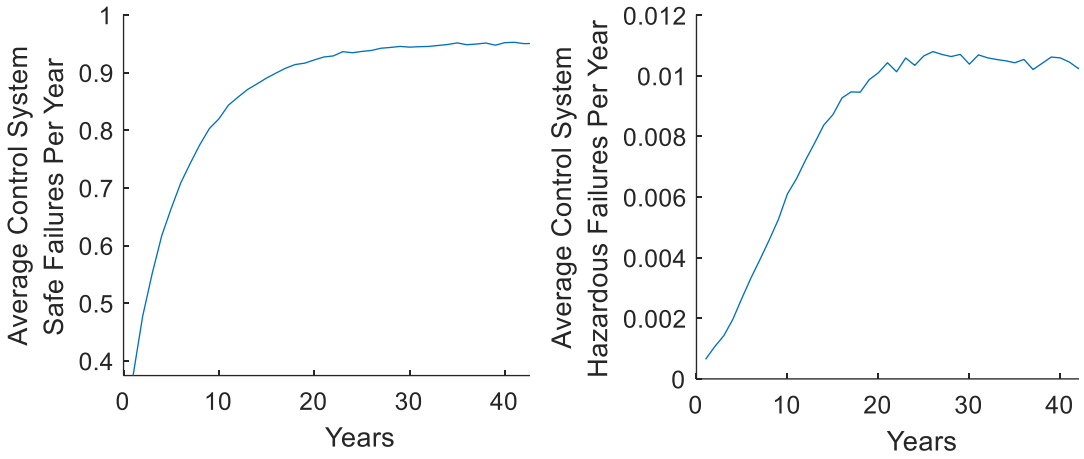

Figure 7-28 Control System Failures without Periodic Renewals

7.2.15.2 Specific Hazardous Failures

For each year of operation, the model outputs the expected number of trains which pass the crossing when it is in each specific failure state. This can be used with the results obtained in Chapter 5 from the traffic model to determine the expected number of collisions between road and rail vehicles directly attributable to crossing failures. This can be used to determine if changes to maintenance regime or usage of a crossing will have an adverse effect on safety.

Failures of the protection systems are mostly triggered by failures of the control system. This is unsurprising given the number of the moving parts within the control system and the wide range of functions it performs. Figure 7-29 shows the number of trains passing whilst the crossing has failed safely, where both lights and barriers remain activated regardless of whether a train is approaching or passing. This shows a cyclic pattern, broadly following the renewal pattern of the control system, though other systems can trigger a complete fail safe event.

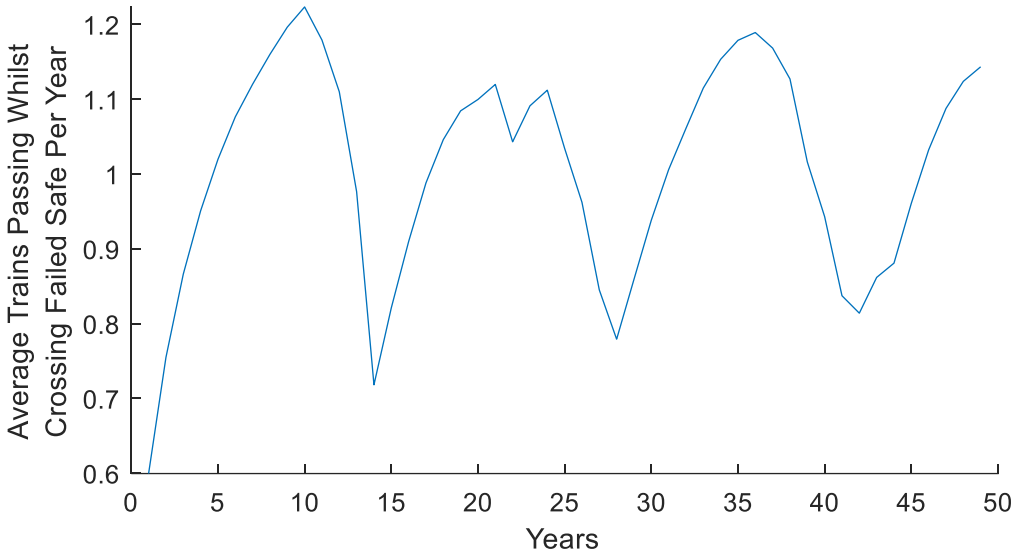

Figure 7-29 Trains Passing Crossings Whilst Both Lights and Barriers are Active in Perpetuity.

Other Protection system failures exhibit more complex changes over time. The number of trains passing per year when the lights are not operational is one such example, shown in Figure 7-30. The total values follow an incoherent pattern, which can only be understood when it is separated into the system failures that caused the lights to fail. Both the control system and the lights themselves may be responsible for the lights failing to operate as a train passes. For this type of protection system failure, the light system is responsible for the most trains passing at danger. This is because whilst light failures may not occur more frequently, there are failure modes which are not self-reported, and therefore may not be discovered until an inspection occurs. In contrast, all control system failures which affect the functioning of the lights are self-reported.

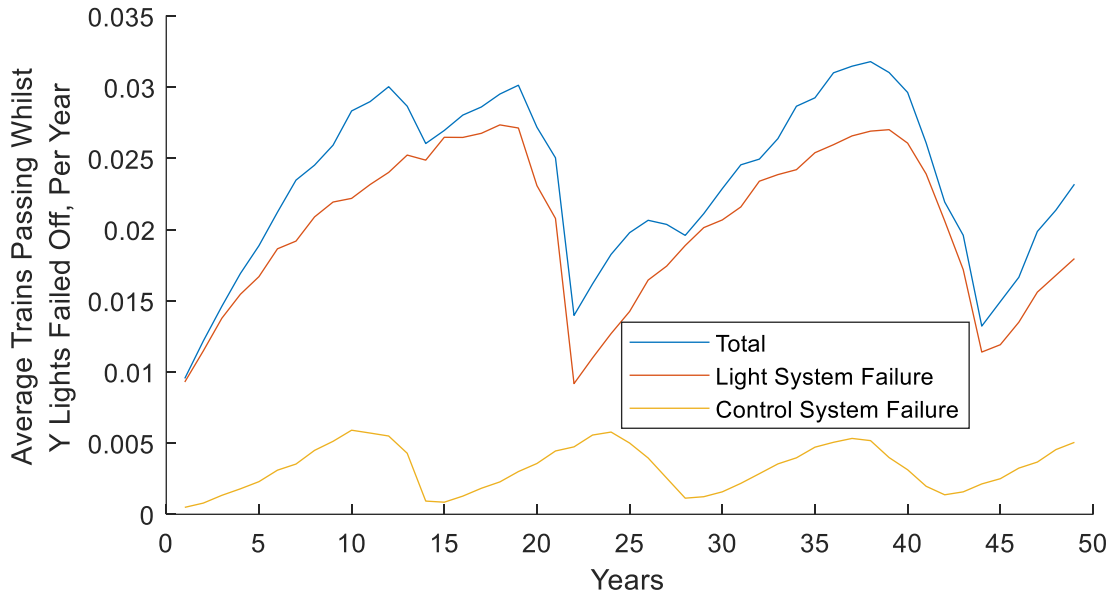

Figure 7-30 Trains Passing Crossings Whilst Y Side Lights are Off per Year.

The most serious failures occur when neither protection systems respond to an approaching train. The occurrence per year of these failures for the crossing is shown in Figure 7-31. The convergence of this result is poor due to the rarity of this event. These failures occur at a constant rate as they are caused by welding of specific control system relay contacts which are modelled with a constant probability of occurrence.

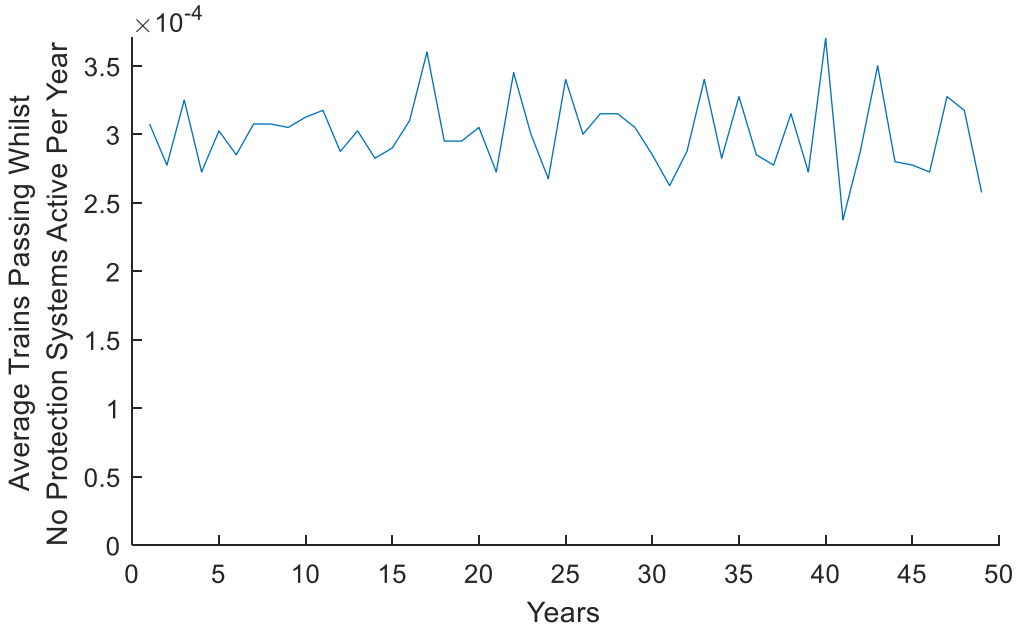

Figure 7-31 Graph of Complete Protection System Failures against Time.

7.2.15.3 Power failure

Complete power failures are rare, and display a cyclic pattern following the renewal cycle of the power supply equipment and battery backup. As back-up battery capacity decreases due to wear and age, the number of power failures per year increases. This is shown in Figure 7-32.

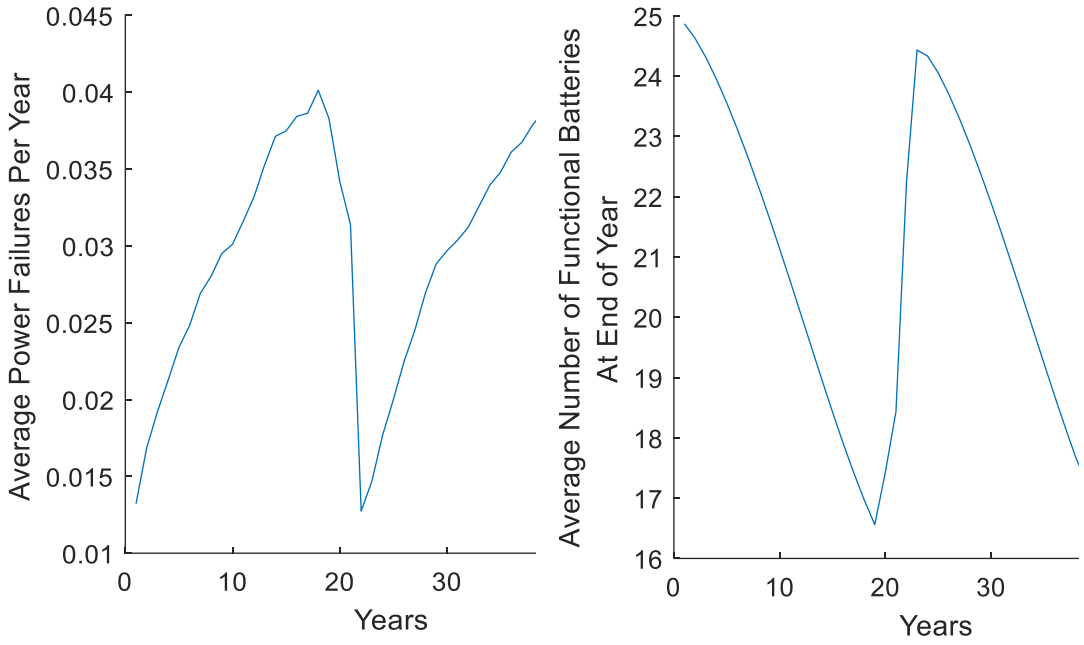

Figure 7-32 Graphs showing Correlation between Power Failure Frequency and Back-Up Battery State

De-activating periodic renewals for the power supply and back-up battery reduces this cyclic behaviour, however it still remains, shown in Figure 7-33. This is because the back-up batteries are only replaced when they fail to provide sufficient back-up battery power. As back-up battery capacity decreases, power failures become more likely, and thus so are unplanned battery system renewals. Rates of complete power failure peaks at around 35years. This conflicts with the lowest number of functional batteries occurring at 25years. This is because the model captures data about the system state at the end of each year. As a result, poor battery condition at the start of the year resulting in a power failure, and then initiating a renewal will record as-new battery condition at the end of the year. Considering that the gradient of the number of functional batteries graph is the sum of the rate of battery failure and rate of battery replacement, the highest period of battery replacement, and hence power failures, occurs at the point of inflection at year 35.

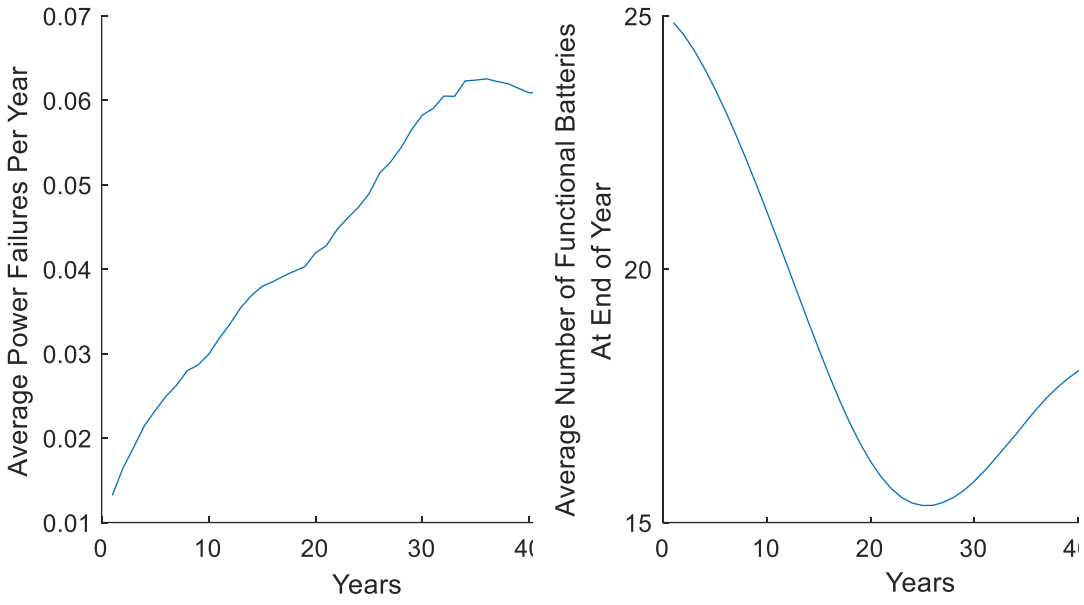

Figure 7-33 Graph Showing Effects of Not Renewing Power Supply System on Power Failure Frequency.

7.2.15.4 Train Delays

When a level crossing fails an emergency repair crew is dispatched, until these staff arrive at the crossing to close the road to road vehicles trains will either need to be diverted or delayed. The model outputs the expected number of trains delayed per year, an example shown Figure 7-34. The number of delays follows the cyclic pattern of renewals, with renewals reducing delays in the years after.

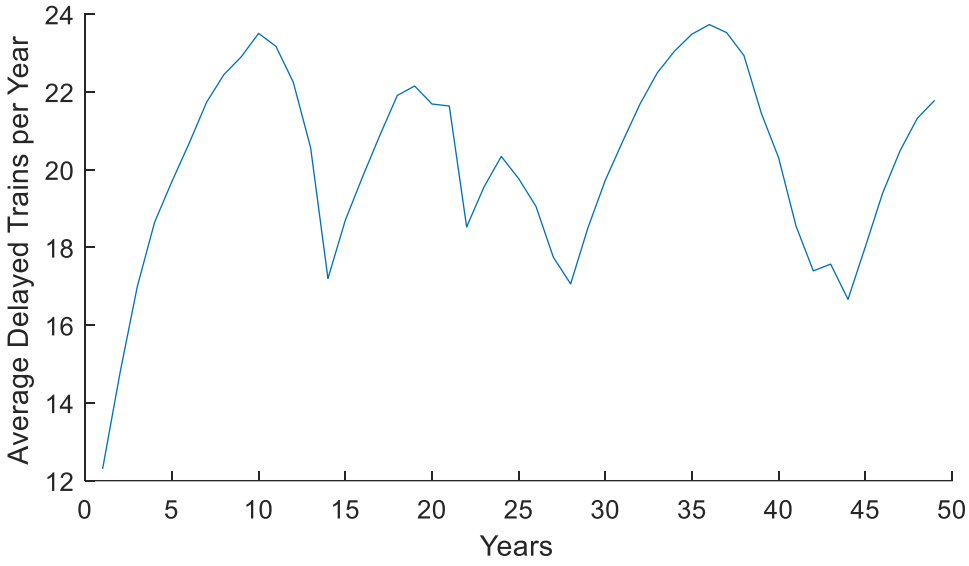

Figure 7-34 Frequency of Delayed Trains due to Crossings Malfunction.

7.2.15.5 Maintenance and Renewals

The model outputs the expected number of maintenance actions, inspections and renewals. Graphs showing the average number of repair work orders carried out against time are displayed in Figure 7-35. The left hand side of the figure shows the number of routine work orders carried out, this shows a cyclic relationship in sync with the renewal of the road surface and cabin renewals which occur around every 22years. The urgent work orders show a now familiar pattern, matching the occurrence of protection system failures.

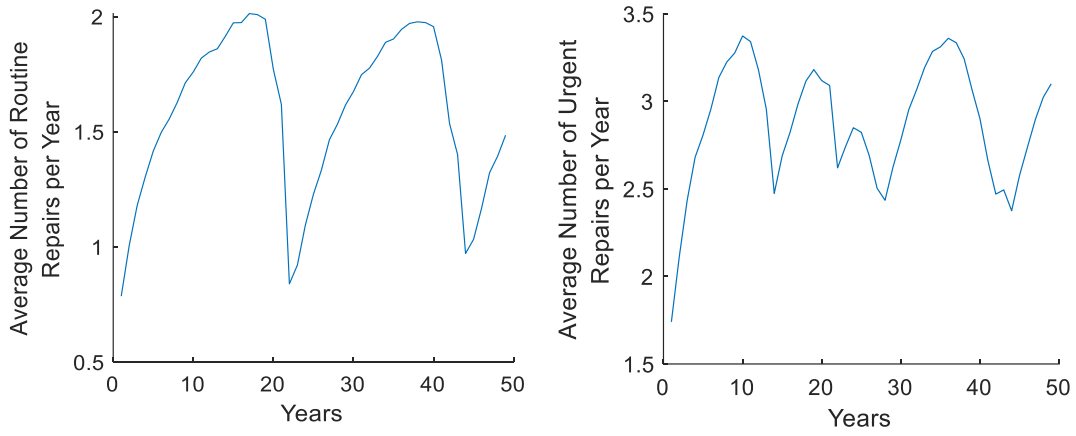

Figure 7-35 Work Order Generation Frequencies

The rate with which inspections are carried out on the crossing is shown in Figure 7-36. The figure also displays the actions which triggered the inspection. Inspections carried out after a repair has occurred follow the same pattern as the generation of work orders. Renewals irregularly trigger inspections. Scheduled inspections occur more frequently than opportunistic with the parameters used, both follow an irregular pattern as the occurrence of repair or renewal actions reduces the need for normally scheduled inspections.

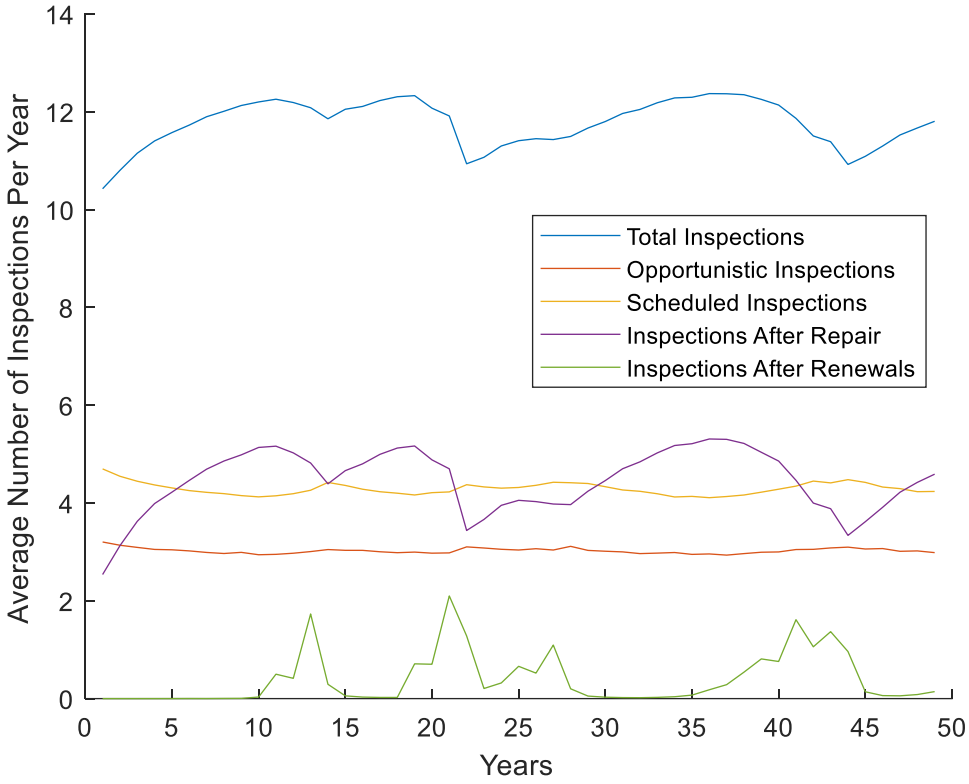

Figure 7-36 Inspection Frequency

Finally, the model keeps track of the expenditure of replacement parts for each level crossing system. An example of this is shown in Figure 7-37. This shows the bulk of the expenditure to be from replacing deteriorated road surface blocks, for the model parameters used.

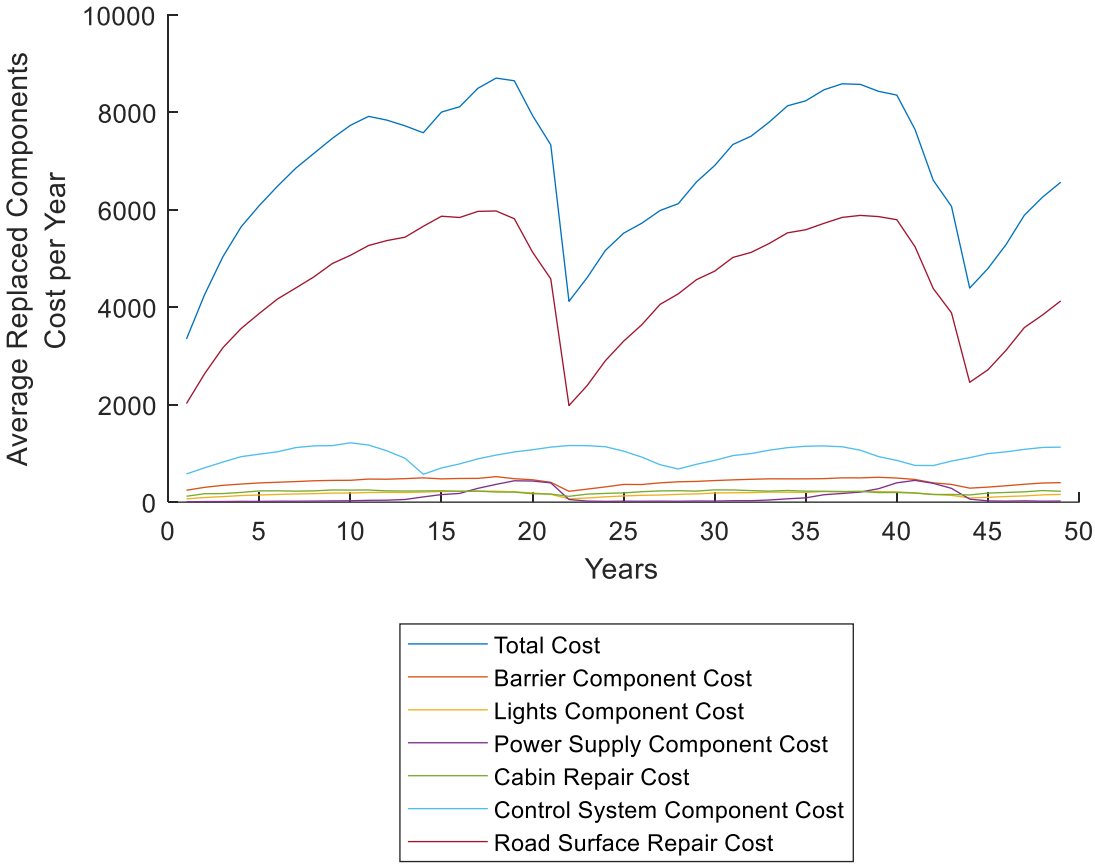

Figure 7-37 Replacement Parts Cost Prediction, Excluding Renewal Costs.

7.2.16 Risk Consequences of Asset Management Planning

Effective allocation of resources is key to asset management decision making. The union of the collision risk model and the asset management model presented within this chapter allows exploration of the effects of asset management decisions on risk. Several basic asset management policies were explored and compared to a baseline policy. The baseline policy renewed all crossing systems every 15years. Other parameters are the same as used the analysis presented previously in this chapter, listed in Appendix B. The effects of frequent renewals, renewing every system every 5 years was explored, as well as never performing renewals.

The effects of changing urgent maintenance response times were explored, as would occur if more staff were hired, or based closer to assets. In addition to the baseline response time, a rapid response time of mean 0.075days (SD = 0.04days) and a slow response time of mean 0.3days (SD = 0.16days) was explored. All used the log-normal distribution and the model structure described earlier. The effect of changing inspection frequencies was explored.

Using a baseline inspection schedule of once per 42 days, a comparison was made with a more frequent schedule of once every 21 days and a more relaxed schedule of once per 63 days.

The results of this analysis were obtained by simulating the asset management model for 50years for each scenario and averaging the results per year, 80 trains pass the simulated crossing per day. To assess the change in collision risk, it has been assumed that the crossing is high risk, with an average vehicle arrival rate of 1.8 per minute. From the results of the collision risk analysis model, this corresponds to the following probabilities of a collision per train passing whilst hazardous failures are present: Lights failed off but barriers still operating, 2.8 * 10^-6; barriers stuck in the raised position but lights operating, 6.9 * 10^-5; both barriers raised and lights off, 0.09.

The results of this analysis are shown in Table 7-3. This table shows the prediction of the number of trains which will pass the crossing each year whilst the fault is undiscovered, and thus not mitigated in any way. It shows that the selection of an asset management plan has little effect on the occurrence of hazardous failures. Based on the design of the model, this is to be expected. Almost all hazardous failures are self-reported by the crossing, thus changing the inspection frequency has no effect. Maintenance teams only respond after a fault has been discovered and therefore changing the response time does not affect risk of hazardous failure events.

Changing the renewal frequencies does have a large effect on the risk of the lights failing, this occurs due to high resistance relay contact failures in the control system and bulb failure. It has minimal effect on hazardous barrier failures and no effect on simultaneous light and barrier failures as these are extremely rare and modelled as occurring with constant probability.

Table 7-3 Effects of Asset Management Planning on Hazardous Failure Events.

Failure Event	**Lights Failed Off Only**		**Barriers Failed Up Only**		**Lights Failed Off and Barriers Failed Up**	
Asset Management Plan	Trains Passing Per Year	Estimated Collisions Per Year	Trains Passing Per Year	Estimated Collisions Per Year	Trains Passing Per Year	Estimated Collisions Per Year
Baseline	0.042	1.2E-07	0.0010	7.1E-08	0.00058	0.000053
Frequent Renewals	0.024	6.7E-08	0.0010	7.1E-08	0.00058	0.000053

No Renewals	0.070	2.0E-07	0.0011	7.7E-08	0.00058	0.000053
Frequent Inspections	0.042	1.2E-07	0.0010	7.1E-08	0.00058	0.000053
Few Inspections	0.042	1.2E-07	0.0010	7.1E-08	0.00058	0.000053
Rapid Maintenance Response	0.042	1.2E-07	0.0010	6.9E-08	0.00058	0.000053
Slow Maintenance Response	0.042	1.2E-07	0.0010	6.7E-08	0.00058	0.000053

The effects of these asset management plans were also used to explore the occurrence of fail-safe events, shown in Table 7-4. These fail-safe events were divided into trains passing whilst the fault was undiscovered, and further trains passing whilst the faut was discovered. Here, changing the asset management plan has a significant effect, however the effects of fail-safe events on collision risk have not been explored so the effect on safety here is unknown.

Table 7-4 Effects of Asset Management Planning on Fail-Safe Events.

Asset Management Plan	Trains Passing Whilst Crossing Failed Safe Per Year	
	Fault Undiscovered	Fault Discovered
Baseline	2.00	17.77
Frequent Renewals	1.42	12.59
No Renewals	2.57	22.79
Frequent Inspections	2.00	17.78
Few Inspections	2.00	17.74
Rapid Maintenance Response	2.00	8.56
Slow Maintenance Response	2.00	30.86

Despite the effect of fail-safe events on collision risk being unknown, it is still possible to optimise the model parameters to minimise them. Section 2.5 of this thesis outlines how Genetic Algorithms can be used to optimise for both cost and risk. Single candidate optimisation methods require cost and risk to be related, typically by applying a cost to the occurrence of any hazardous events. However, as Genetic Algorithms are a multi candidate algorithm, a ranking system can be used instead allowing the optimisation of unrelatable objectives, in this case, maintenance cost and occurrence of fail-safe events.

7.2.17 Model Validation and Calibration

The model presented here has been created using assumed distributions for degradation, it has not been calibrated or validated. This was a deliberate decision to allow the work within this thesis to explore the computational costs of solving various types of various Petri net model via Monte Carlo methods. In leu of calibration and validation, a brief outline of how this could be achieved is outlined here.

7.2.17.1 Validation

The most rigorous methods of validating models use external means and data. This may entail using independent parties to use the model with data obtained from a different time and place to validate its efficacy. It would be difficult to have the model result independently verified as part of a doctoral thesis, however it is possible to obtain data from different times and places.

Whilst AHBC crossings are only used within the UK, Network Rail divides the UK into regions, each of which has a separate asset management budget and plans. Every five years, the asset management budgets and plans are reviewed against past performance and updated to improve performance and resolve current issues, referred to as Control Periods. To date there have been five control periods. This provides many options for calibrating the model's stochastic transitions and validating the model against data from different times and places. Analysis of the different regions' asset management plans during each control period may discover some variety in the asset management plans for AHBCs. This would provide an opportunity to validate the model against a variety of different maintenance policies.

Where there are two or more similar asset management plans and asset ages, one Control Period or region's data could be used to calibrate the model. Its predictions could be then verified against the other similar dataset from the different region/Control Period. Should there exist a unique asset management plan, it is possible to both calibrate and validate the model with the same dataset by splitting it, using half of the data set to calibrate and half to validate, however doing so is less robust than using separate datasets obtained from different regions and times.

Network Rail collect a variety of data on the maintenance of level crossings which could be used to validate this model. Records of components failures are kept which include the location, date of failure, component failed and failure mode. Maintenance records are kept which detail when a failure occurred, the nature and location of the failure, the time the repair began and was completed. Records of renewal works are kept in a similar manner.

Detailed inspection records of crossings are kept enabling independent verification that inspections are being carried out according to the specified frequencies and details. An annual detailed inspection is carried out which may enable assessment and analysis of the accuracy of the routine assessments.

The data kept is sufficient to calibrate the model as designed, however the processing required to do so is substantial, requiring disparate database records to be joined and cleaned in a manner which does not affect their validity. Many records have been entered in an inconsistent manner which whilst suitable for human reading, is difficult to programmatically analyse. As such, this has not been completed as part of the work in this thesis.

7.2.17.2 Calibration

AHBC crossings are not an entirely homogeneous asset. With the design of the model, and the data available, there exists several opportunities through which the model may be calibrated to represent a specific asset. The potential differences between AHBC assets and modelling adaptions required to represent them are discussed below.

AHBC crossings follow a standardised design, however where required this design can be modified. Non-standard track circuit configurations can be accommodated by modifying the control system circuits which detect the approach of trains. This may be required if the crossing is located near a junction, or where several crossings are very close together. Another common modification to the standard design occurs when additional road traffic lights are required to ensure the lights are visible from all possible approaches to the crossing. Minor changes such as these can easily be incorporated into the model through the addition or modification of the Petri net structures within.

The statistical distributions which model the time till components fail may be calibrated to represent the crossing's own failure rate. It is expected that failure rate data from an individual crossing will be insufficient in quantity to accurately derive parameters for modelling failure for all components owing to infrequency of some failure modes, but there are common features among these assets that will facilitate calibration. The frequency of passing trains is likely to be a significant factor in the lifespan of many components, as most crossing systems only operate whilst a train is passing. Using this, crossing component failure data may be grouped by rail traffic volume and matched to specific assets. Similarly, failure data could also be grouped by equipment supplier, date of manufacture, design revision or geographic location of the asset.

A limitation of the model presented is that it does not include any measure of asset condition. Commonly, asset management models categorise condition through several states which equate to a subjective measure of condition such as: good; serviceable; poor; or a condition state which corresponds to an interval of an important condition measurement such as geometry deviation or the size of a defect. The absence of this creates uncertainty when modelling a specific asset which may have defects affecting its rate of degradation or failure.

Several forms of degradation were identified affecting AHBC crossings which could be used as a measure of asset condition. Listed below:

- With use and environmental exposure, the relay contacts within an AHBC's control system wear, resulting in increasing values of electrical resistance, which ultimately causes the control system to malfunction necessitating the replacement of the relay (Kagra, 2013).
- The condition of the hydraulic barriers may be inferred from the electrical current drawn by the motor and the time taken for the barriers to rise (Roberts, et al., 2010).
- The light output of the LEDs within the road traffic lights could be used to infer their condition, as LEDs dim as they degrade (Pecht & Chang, 2012).
- Back-up battery degradation could be measured by discharging them and measuring capacity.

These measures of condition, and potentially others, could be incorporated into the model to allow further calibration of the model to represent a specific asset. They have not been included as these measurements are not routinely taken, and as such the data is neither available to estimate how they affect statistical parameters for modelling failure, nor to determine the condition of any given asset. Should the collection of this data become routine, its inclusion within the model will reduce the uncertainty when calibrating it a specific AHBC crossing.

7.2.18 Model Convergence

Convergence and the number of simulations required to achieve it was tested using the central limit theorem, described in chapter 2. First convergence of the total replacement parts cost over fifty years of operation was explored. Figure 7-38 shows the rolling average cost and 95% confidence interval for 500,000 simulations. On average, only 260 simulations were required to obtain a 95% confidence interval within +/- 1% of the total cost.

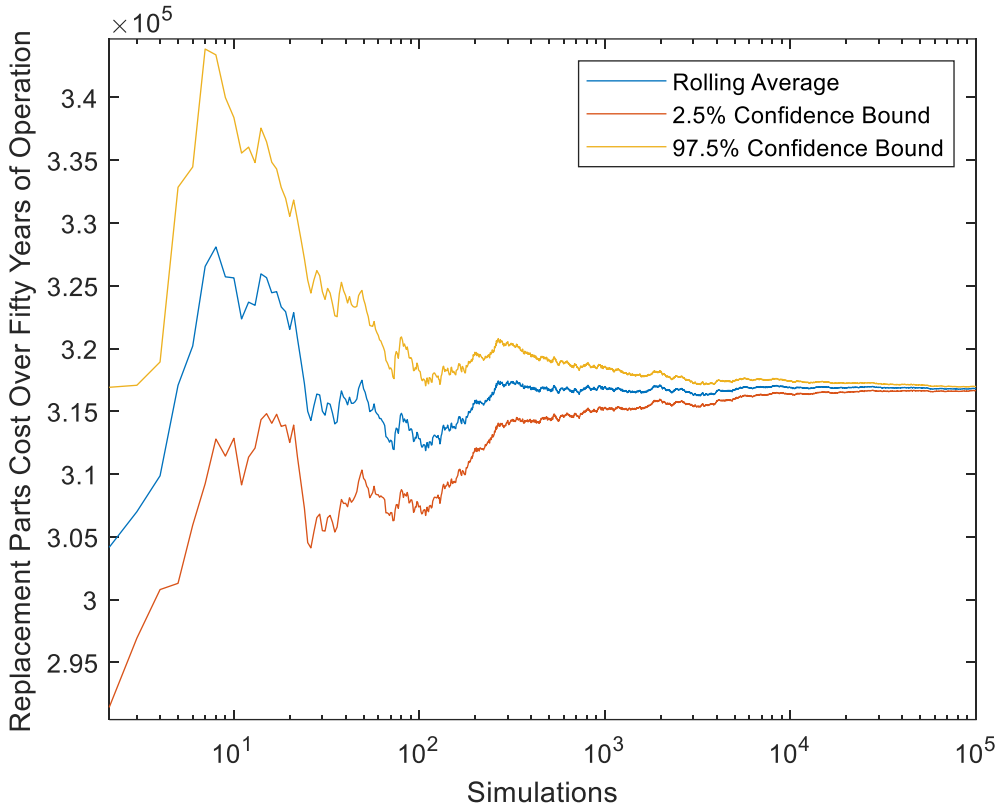

Figure 7-38 Convergence of Replacement Parts Cost for Fifty Years of Operation.

Next the convergence of a single year's replacement parts expense was explored. As this comprises a much smaller time frame, many more simulations were required to achieve convergence. On average, 34,000 simulations where required to achieve a 95% confidence interval within +/- 1% of the cost of replacement parts for the 50th year only.

Rare hazardous failure events require far more simulations to converge in comparison. Figure 7-39 shows convergence of the total number of trains passing a crossing whilst the Y side lights are off over a 20 year period. On average it required 120,000 simulations to achieve a 95% confidence internal within +/- 1% of the number of trains which pass the crossing without lights operating over a 20 year period. Achieving the same level of convergence for a single year of operation required approximately 1.5 million simulations, however this remained computationally feasible using contemporary computing hardware.

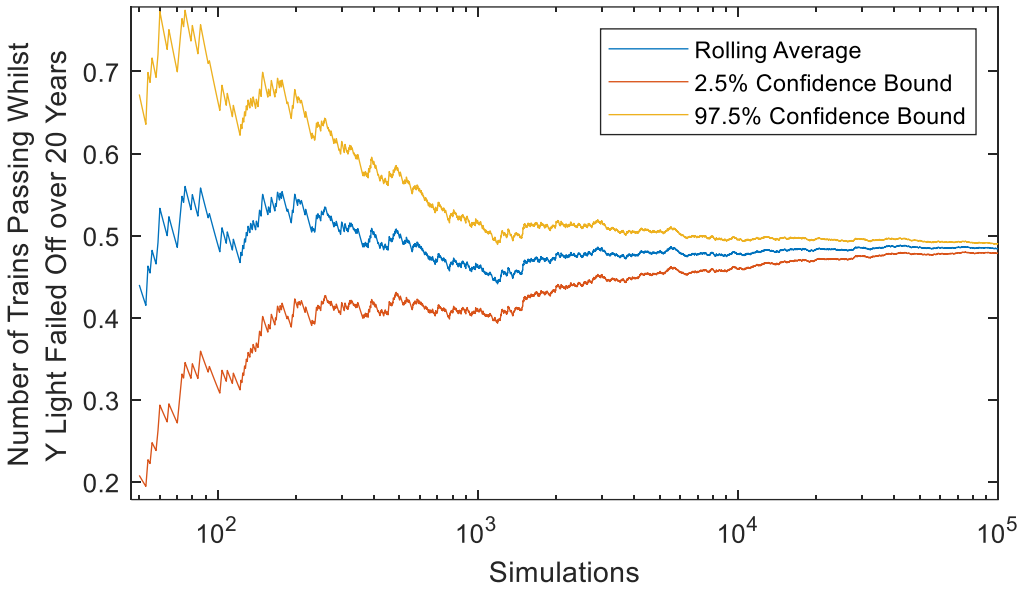

Figure 7-39 Convergence of Trains Passing Whilst Y Lights Failed Off Over 20 Years.

Multiple crossing assets were simulated within the same model by duplicating the Petri net structure for each asset. The convergence rate of these multiple asset models was found to be governed by the total number of crossing assets simulated, regardless of whether they were grouped together or simulated separately. Figure 7-40 shows a graph of the convergence of the size of the 95% confidence interval for the 50 year replacement parts costs against the product of the number of assets per model and number of times the model was simulated. It shows a similar convergence rate, regardless of the number of assets per model. Demonstrating that the number of assets per model does not affect the convergence rate. This is to be expected, as all assets are identical and there is no interaction between them. It would not be expected however, if the assets were different or had some interaction.

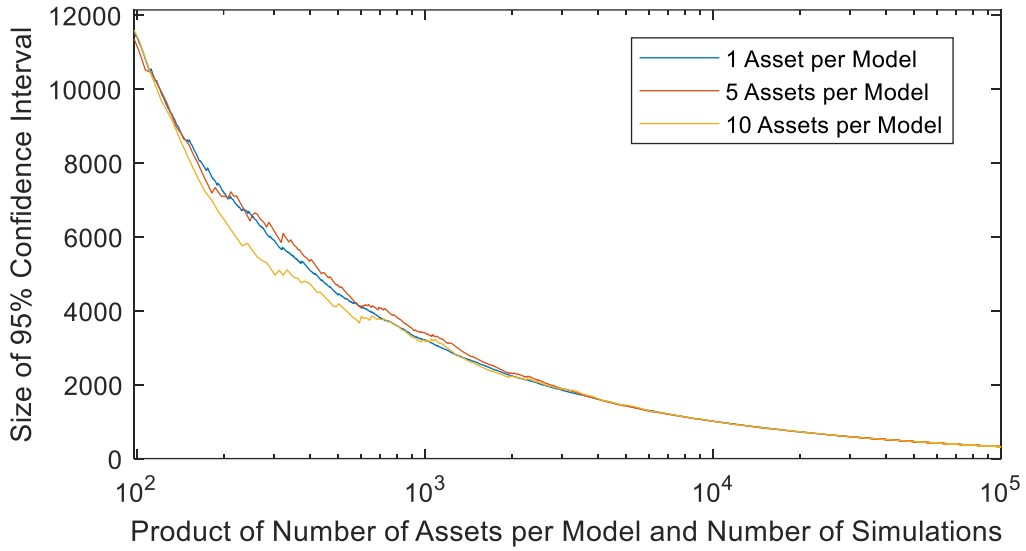

Figure 7-40 Graph of the Size of the 95% Confidence Interval for 50 Year Replacement Parts Costs For Varying Numbers of Assets Per Model.

7.2.19 Stochastic Petri Net Simulation Software

Monte Carlo solutions can be computationally taxing, owing to the large number of times the model must be simulated in order to attain a suitable level of confidence in the solution produced. When simulating single assets this is unlikely to pose an issue unless a complex parameter optimisation is performed. However, consider the application of an asset management framework to the creation and solution of a route, or network level model. Such a model would be dramatically larger than a single asset model and potentially much more complex. Due to this, the effects of scaling the model to include increasing numbers of assets have been explored for this stochastic Petri net model and each of the other Petri net methodologies considered.

The computational requirements for Petri net simulation depend heavily on the software used for simulation. Software to simulate stochastic Petri nets was written specifically for this work. It was designed to offer the best simulation performance possible within the scope of this work. This was also done for the other Petri net methodologies considered. This is to enable a fair comparison between each methodology. However, it must be acknowledged that the quality of software developed may not be equal for each methodology, and this may affect any conclusions drawn. To mitigate this, the general architecture of each Petri net methodologies software will be described to facilitate a fair comparison.

The choice of programming language can have a large effect on the computational times and memory requirement for software. Generally, comparing the computational requirements

for a language is difficult, as the task and context have a large impact on the results. Fourment and Gillings studied the computational requirements of common programming languages for bioinformatics applications finding C and C++ to be both the fastest and most memory efficient languages (Fourment & Gillings, 2008). A later paper applied common programming languages to solving a small stochastic economics model found broadly similar results (Aruoba & Fernandez-Villaverde, 2015). For these reasons, C++ was chosen to implement the software.

The core loop of this software is shown in Figure 7-41. It comprises four main functions: Pre-Processing; Transition Queuing; Advance Time; and Fire Transitions. In brief, the Pre-Processing function creates a table which is used to reduce the computational requirements of the simulation. The Transition Queuing function determines which transitions are enabled and their respective firing time delays. The Advance Time function determines which transition(s) will fire next. And finally the Fire Transitions function performs the addition and subtraction of tokens when a transition fires.

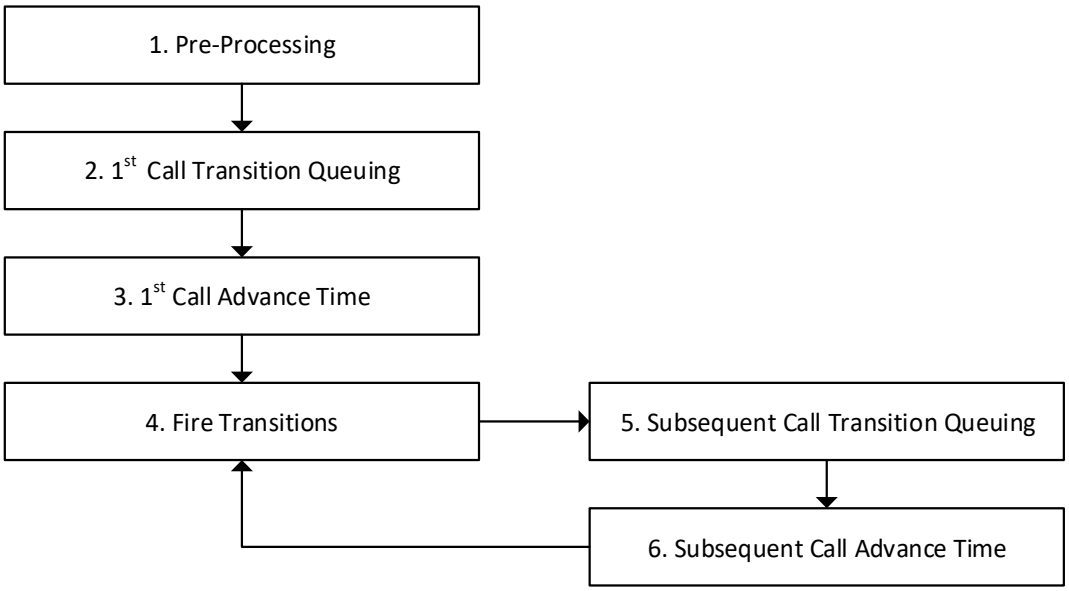

Figure 7-41 Simulation Software Core Loop

These functions are described in greater detail below.

1. Pre-processing is performed once at the start of the simulation. It analyses the structure of the Petri net to find which transitions may be enabled or disabled when any given place's token count changes. This table is used by the transition queuing function described later to reduce its computational requirement.

2. The first time Transition Queuing is called it iterates through each transition within the model. The state of each transition is determined, if it is enabled the time at which the transition fires is determined and recorded. This is referred to as queuing. Whether currently queued or not, all transitions are divided into blocks of 1000. The number and time of the soonest transition to fire is stored for each block. Should the number of blocks be greater than 1000, this process is repeated to determine the soonest transition to fire and its time within the block of blocks. This process is repeated until the highest level of blocks within the hierarchy numbers less than 1,000.

3. The first time the function Advance Time is called the block hierarchy has a complete list of when of the soonest transitions will fire. It therefore only need iterate through the highest level blocks in the hierarchy to find the time the next transition fires at. Once this time is found, the simulation time is advanced to this time.

4. This function fires the transition passed to it from the Advance Time function. Tokens are added and removed from places as defined by the transition being fired. A table of all the places whose token count was affected when the transition fired is created.

5. The Transition Queuing function behaves slightly differently now the first simulation cycle has completed. It uses the table of all the places which were affected when the last transition fired, and the table created from the earlier pre-processing to determine which transitions were affected when the previous transition fired. Now, it only rechecks the state of any transitions which were affected. It does not waste resources checking the state of transitions whose state could not possibly have changed. Simulating large network level models which may contain many millions of transitions would not be possible without optimisations such as this.

Any transitions which are no-longer enabled are disabled. If and when a transition is queued to fire, its firing time is compared against the current soonest for the block(s) it resides within. If it is queued to fire sooner it replaces the previously stored soonest transition.

In the block which contains the transition which fired the previous cycle, this comparison cannot be made. This is because the soonest transition to fire in these blocks is no longer known.

6. On subsequent callings, the Advance Time function has more tasks to perform. One block from each level in the hierarchy will no longer have a known soonest firing transition. The Advance Time function must iterate through each lower level affected

block in the hierarchy to determine the soonest transition firing time. It may then iterate through the highest level blocks to determine the next transition to fire and its firing time.

This represents a considerable computational saving compared to not grouping the transitions in blocks. Consider a model with 1,000,000 transitions, divided into 1,000blocks. Without dividing into blocks: 1,000,000 comparison operations must be performed to determine the next transition to fire every simulation cycle. With blocks: all 1,000 transitions within the block containing the previous transition to fire must checked, along with the 1,000 blocks themselves to find the next transition to fire, for a total of 2,000 comparison operations only.

7.2.20 Computational Test Results

Computer RAM is a potentially limiting factor in computational simulation, and may limit the maximum number of assets a framework can model. The amount of RAM memory required to simulate the model was plotted by progressively increasing the number of level crossings simulated within the same model and recording the peak memory usage. The plot can be seen in Figure 7-42. It shows a linear increase in memory used as the number of crossings are increased. At 4,000 level crossings the memory requirement reached 5.7 Gigabytes, corresponding to approximately 1.4 Megabytes per crossing. This is equivalent to 6.7million transitions and 5million places in total. At this point the experiment was terminated, taking 5Gigabyte of memory usage per simulation as a reasonable upper limit for memory usage on contemporary high-end computers.

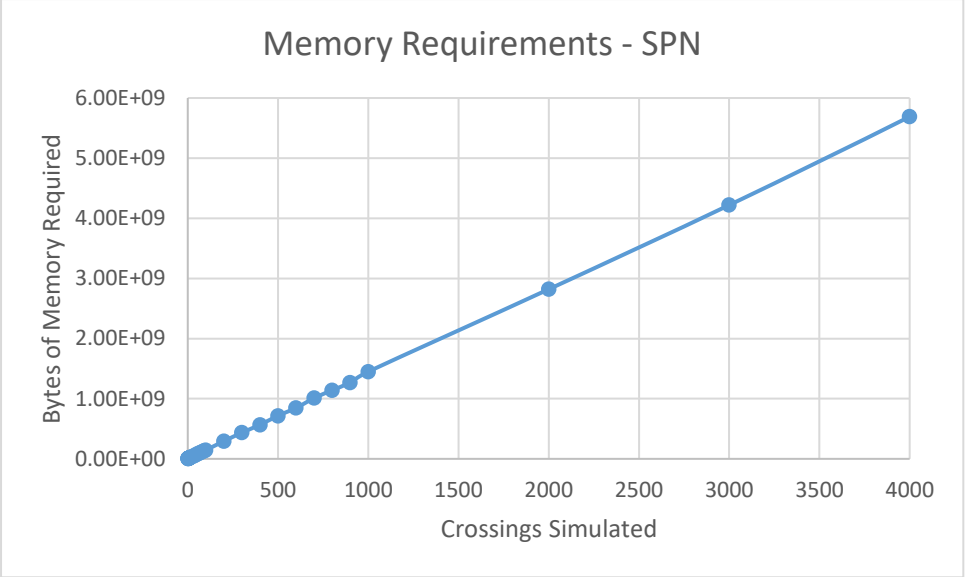

Figure 7-42 SPN Memory Test Results

A breakdown of the data structures and memory requirements to store the Petri net is provided in Table 7-5.

Table 7-5 SPN Memory Breakdown.

Petri Net Element	Memory Requirement
Mapping of arcs from Places to Transitions	Each arc is stored in an STD integer Vector, the vector uses 1 row per transition, and this row is populated by pairs of integers detailing the place the arc originates from and its multiplicity. This requires 3828 integers per crossing.
Mapping of arcs from Transitions to Places	As above, however now the row contents relates to the output place. This requires 20252 integers per crossing.
Inhibit arcs	Each inhibit arc is stored in an STD integer Vector, with an integer value for the originating place, the terminating transition and its multiplicity for a total of 3 integers per inhibit arc. With 2073 inhibit arcs within the model, this requires 6219 integers.
Transition details	The details of each transition, namely the parameters determining its type and delay are stored in an STD double Vector. The number of parameters per transition varies based on the type. There are 1587 transitions within the model, requiring 4116 doubles. In many instances the precision obtained using the double type is unnecessary here, a more complex data structure with multiple types would marginally reduce memory requirements.
Tokens	An STD integer Vector stores the number of tokens held within each place.
Optimisation Tables	The table which links places whose token state has changed to transitions which may be affected is stored in an STD integer Vector, requiring 3987 integers per crossing.

Next the computation time was tested for increasing numbers of crossings within a single model. The resulting model was simulated for 50years of operation. The results of which can be seen in Figure 7-43. This shows a linear increase in the time required for the simulation to complete against the number of level crossings simulated. At 5,000 level crossings, 357seconds was required to complete 1 simulation of the model. During this time, the total number of transition firings simulated was 80,000,000 which averages at 12 firings per

transition. No larger numbers of crossings were simulated owing to memory limitations outlined previously. From these results, it appears that the RAM memory requirement is the limiting factor for this methodology, given the software developed and the model itself.

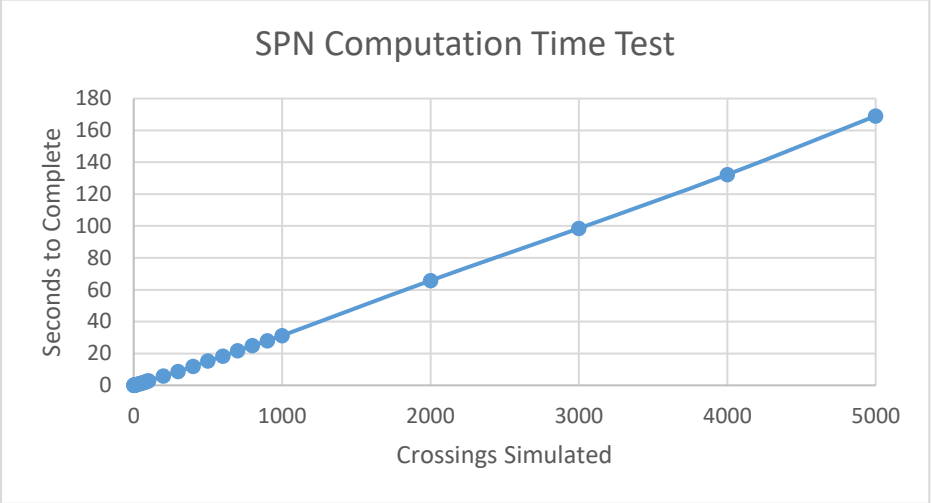

Figure 7-43 SPN Computation Time Test

Omitting the SPN Simulation optimisations described earlier produces a dramatic increase in computation times. Figure 7-44 shows the time for a single simulation to complete with all software optimisation removed. Only 30 crossings were simulated in ~160seconds, whereas the optimised software was able to simulate 5,000. It also shows an exponential increase in computation time, making simulating larger numbers of crossings increasingly infeasible. This occurs because only one transition fires per simulation loop iteration, but without the optimisations implemented each transition in the model is evaluated each iteration.

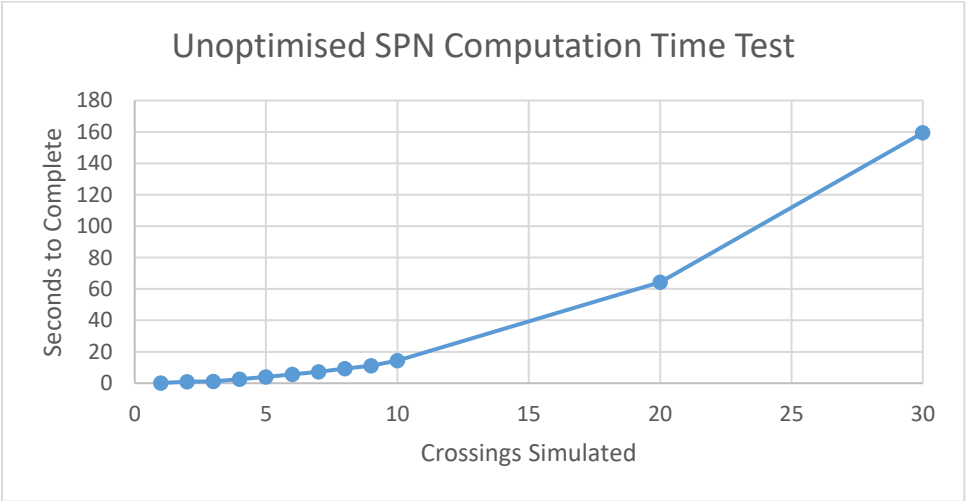

Figure 7-44 SPN Computation Time Test without Software Optimisations.

There was only one large SPN model for railway assets found in literature with which to validate the efficacy of the simulation software developed. Litherland (2019) created an SPN model for rail track maintenance. This model used 134k transitions and 134k places, and was simulated for 35years of operation of the rail line. It required 1,000seconds per simulation to complete. The memory required per simulation is not stated.

An equivalent size SPN level crossing model would contain 100 level crossings. This required 100MBs per simulation, and took 6.5seconds to complete. A fair comparison is difficult, as Litherland does not state how often each transition in their model fires. If it is assumed that on average both model's transitions fire a similar number of the times the large discrepancy in simulation times can be explained by the optimisations the software in this work uses. Litherland does not use any kind of search algorithm or structure to reduce the computational cost of finding the next transition to fire chronologically. Thus, their software must perform 134k comparison operations every simulation loop in order to find the transition with the lowest firing time, in contrast to the software in this work which requires a hundred times less.

7.3 Colour Petri Net Base Model

A CPN model has been created to allow a comparison of the computational requirements of CPN and SPN models. To reduce the work required to create and simulate the CPN, only the following modules were included: control system relay contacts; inspections; reactive repair; protection system states. This will be compared with a version of the SPN model presented earlier, containing the same modules. As both models are logically identical, this will allow a fair comparison of the computational requirements of each methodology. Unlike the SPN model, additional crossing assets may be simulated without duplicating the Petri net structure, instead an additional set of tokens need only be added to the model.

7.3.1.1 Relay Component Failure

The only components considered in this limited model are relay contacts. The first module presented in Figure 7-45 models the failure of these elements. Place P0 holds tokens for each working element, and P1 for each failed element. Transition T0 models failure of relay contacts, either through high resistance or welding. Place P2 contains tokens which serve an informative role, which T0 references to model relay contact failures being triggered by the passage of trains, as per the SPN model.

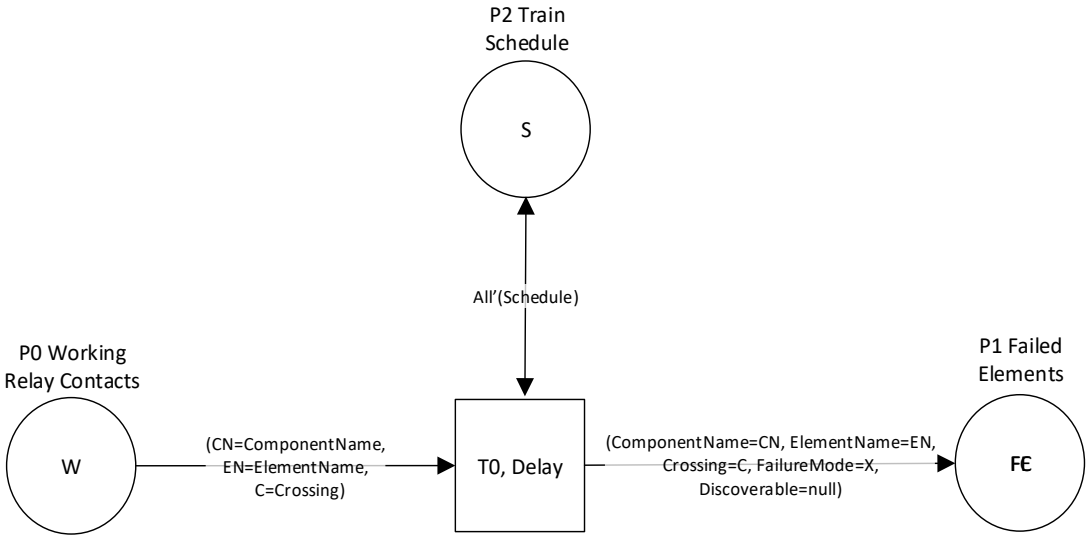

Figure 7-45 Complex CPN Transitions T0 and T1

When the model is initialised, Place P0 contains a token for each relay contact. These tokens conform to composite Colour Working, defined in Figure 7-46. This Colour contains: the component name which refers to the relay the contacts are housed in; the element name, which refers to the relay contacts specifically; and which level crossing asset the element is installed in.

```
539. Colour Working As ComponentName x ElementName x Crossing
540. Colour ComponentName As String
541. Colour ElementName As String
542. Colour Crossing As Integer
543. Colour Schedule As Double
```

Figure 7-46 Colour Working, Schedule Definitions.

Transition T0 models the time to failure for each relay contact. This transition is enabled by the tokens contained in P0, but also uses a test arc connected to place P2. A transition mode is enabled for each token within place P0. The function which determines the delay for each transition mode is shown in Figure 7-47. Lines 546 to 548 generate random numbers for the time till high resistance and welding relay contact failures occur. Line 550 compares both to find the lowest delay. Should the high resistance delay be lower, lines 552 to 557 will be executed. These lines of code first generate a random number between 1 and 50 to determine how many trains pass till the failure occurs, as per the SPN model. Next a function compares this against the schedule from place P2 to obtain the exact delay. This delay is then used as the firing delay time for the transition mode, and the variable X set to "HR" and used later. Should the random number generated representing the time till the relay contacts weld be lower, a similar set of code in lines 562 to 567 will execute.

```
T0 Delay (Schedule) {
        //generate Weibull distributed random variable for time till high resistance failure
        Double HRDelay = Weibull(1.5, 12000)
        //generate exponentially distributed random variable for time till welding failure
        Double WeldingDelay = Exponential(3650000)
        //does high resistance failure occur first?
        If (HRDelay < WeldingDelay) {
                //generate random number between 1 and 50
                Integer TrainsPassing = Uniform(1,50)
                //find time till nth train passes
                Double Delay = FindNextTrain(HRDelay, TrainsPassing, Schedule)
                Return Delay
                //set failure mode variable
                String X = "HR"
        }
        //welding failure occurs first
        Else {
                //generate random number between 1 and 50
                Integer TrainsPassing = Uniform(1,50)
                //find time till nth train passes
                Double Delay = FindNextTrain(WeldingDelay, TrainsPassing, Schedule)
                Return Delay
                //set failure mode variable
                String X = "W"
        }
}
```

Figure 7-47 T0 Delay Function.

When this time elapses, the transition fires the transition mode, removing the appropriate token from place P0 and placing a new token in P1. The new token has several additional data fields, detailed in Figure 7-48 as Colour FailedElement. This colourset includes the specific failure mode and whether the failed element is discoverable or not.

When T0 fires, and the new token added to place P1, these fields are populated. Component name, element name and crossing retain the same values as the token removed from P0. Failure mode is set to the variable X, which may be either "HR" or "W", depending on the outcome of the delay function. Finally, Discoverable is set to the value null, this is triggers a further transition to fire which will be shown next.

```
Colour FailedElement As ComponentName x ElementName x Crossing x FailureMode x RepairTime x Discoverable
Colour FailureMode As String
Colour Discoverable As Boolean
```

Figure 7-48 Colour FailedElement Definition.

7.3.1.2 Protection System Effects Updating Module

This next module concerns the operational state of the crossing's protection systems, ensuring the crossing's operational state always reflects the effects of failed elements, or combination of elements which have failed. This module uses only one transition, T1, shown Figure 7-49. This transition interacts with: P3, a place containing tokens describing the state

of each simulated level crossing; P1, containing all the failed component elements previously described; and P4, a place containing tokens which list the minimal cutset sets and system failure effects they produce.

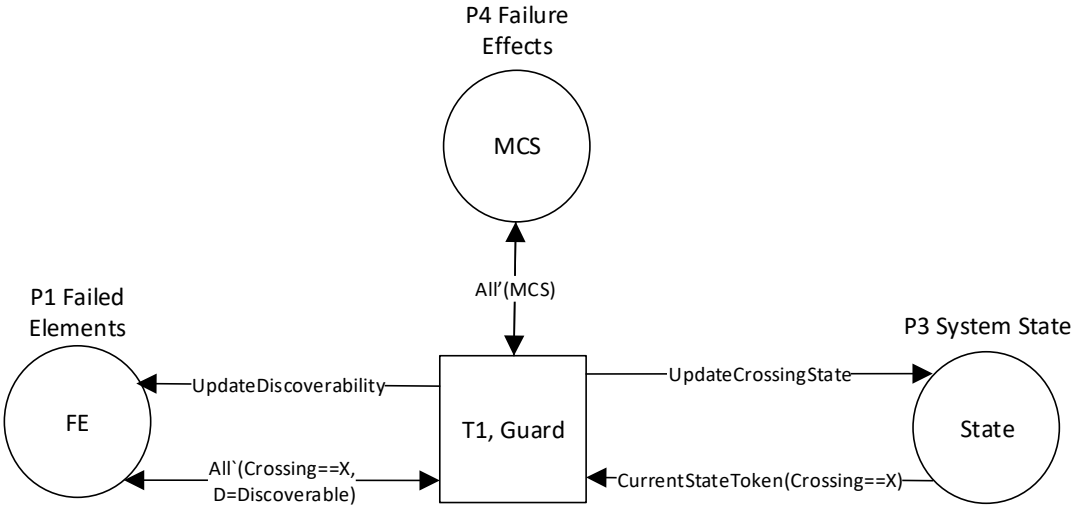

Figure 7-49 Protection System Effects Updating Module

The guard within T1 allows a transition mode of T1 to be enabled whenever a token within place P1 has its Discoverability colour set to null. The presence of such a token means that the system state for the crossing which contains that component element may no longer be valid, as the failure the null token represents has not been included in any prior protection system state checks. The arc functions from P1 to T1, and P3 to T1 specify that each transition mode of T1 is limited to tokens from the same crossing, as failures of components in different crossings cannot affect each other.

Transition T1 fires immediately upon becoming enabled. When T1 fires it checks all the failed components in a crossing against all the minimal cutsets stored in place P4. For this reason, T1 requires all the tokens in place P1 which belong to the crossing, regardless of whether the failed component has been included in a previous check. As it may now match a minimal cutset of failed components in combination with the most recent failed component. Upon finding a match for a minimal cutset among the failed components place P3 is updated to reflect the effects that minimal cutset has upon the crossing's protection systems. If the matching minimal cutset's effects are detectable during an inspection or other intervention then the respective component failure tokens Discoverability colour is set to true, otherwise false. Similarly, if no minimal cutsets include the recently failed component, its token's Discoverability colour is set to false – as a failed relay contact which has no effect on the crossing's protection systems cannot be guaranteed to be discovered during an intervention.

Now the functioning of the Protection System Effects Updating Module will be explained more explicitly. Place P4 contains all the minimal cutsets for all protection system failure states, these apply to all level crossing's simulated. Place P4 uses Colour MinimalCutSets, defined Figure 7-50. This comprises the union of two colours: ElementsStates and SystemEffects. The Colour ElementStates is an n by 2 array where n is the number of failed components in the minimal cutset, the first column of the array refers to the failed element, the second to the failure mode. The Colour SystemEffects is an m by 2 array, where m is the number of protection system effects the minimal cutset causes. The first column referring to the system, the second to the state. Place P3 uses the Colour States, this comprises a union of the SystemEffects Colour and the crossing Colour, allowing it to store the current state of the protection systems all the crossing's under simulation.

```
573. Colour MinimalCutSets As  ElementStates x SystemEffects
574. Colour SystemEffects As Array(m, 2)
575. Colour ElementStates As Array(n, 2)
576. Colour State As SystemEffects x Crossing
```

Figure 7-50 Colour Definitions for MCS and State.

The operation of this module begins with the guard function in transition T1, defined in Figure 7-51. This function iterates through all the tokens in place P1 over lines 579-586, each token is checked if the Discoverability Colour is set to null, line 581. If this is found, a transition mode is enabled for all tokens whose Crossing Colour matches the null token's, lines 583-584. This is achieved by setting the variable X to the crossing ID in the FailedElement token from P1. This same variable specifies that only tokens with matching Crossing Colour values are valid for a single transition mode via the arcs entering transition T1 from places P1 and P3.

```
577. T1 Guard(FE) {
578.            //loop through each Failed Element token
579.            For Each FE Token {
580.                        //is discoverability parameter unset?
581.                        if (D == null) {
582.                                    //enable transition mode for this crossing
583.                                    X=Crossing
584.                                    Transition = enabled
585.                        }
586.            }
587. }
```

Figure 7-51 T1 Guard Function.

Now Transition T1 is enabled in a transition mode, with access to all the tokens from the same crossing from Failed Elements Place P1 and the System Effects Place P3. When the transition fires, arc functions UpdateCrossingState and UpdateDiscoverability run

simultaneously. Both functions uses nested loops as they search for valid minimal cutsets through the tokens present.

UpdateDiscoverability will be described first, this function is defined in Figure 7-52. The first of the nested loops cycles through each token in place P4, line 590, representing each minimal cutset. The second nested loop cycles through each of the element failures comprising the minimal cutset, line 595. The third nested loop, line 598, cycles through each of the tokens from Place P1 representing actual element failures present. The aim of this triple nested loop is to find which minimal cutsets are valid for the crossing. The If statements across lines 602-612 first check if the actual failed component element name matches that within from the minimal cutset. If both ElementName and ElementState match, the component failure token is added to a temporary list. Should any component failures not be present to complete the minimal cutset, the list is discarded and the next minimal cutset tested.

If a complete minimal cutset is found, then all component failure tokens which met the conditions of the cutset are set to discoverable over lines 617-622. After the triple nested loop has completed searching, any component failure tokens which have not found matching cutsets must be set to undiscoverable. This is carried out over lines 625-629, by setting any components whose discoverability colour is not set to false.

Owing to the complexity of this function, a slightly more flexible notation has been adopted. Rather than explicitly declare and pass variables through arcs, colours are instead accessed directly through their containing token or colour. The first example of this appears on line 602, the colour ElementName of the FailureElement token is accessed using the syntax “FE(ElementName)”. In addition, a new variable type ‘PointerArray’ is declared on line 593, this stores the location of tokens allowing them to be accessed later in the function.

```
588. UpdateDiscoverability() {
589.            //loop through each Minimal Cut Set token
590.            For Each MCS Token {
591.                        bool CompleteMatch = true
592.                        //create array to store matching Failed Element Tokens
593.                        MatchingTokens() As PointerArray
594.                        //loop through each Element State in the Minimal Cut Set token
595.                        For Each ElementState In MCS {
596.                                    bool Match = false
597.                                    //loop though each Failed Element token
598.                                    For Each FE Token {
599.                                                bool NameMatch = false
600.                                                bool StateMatch = false
601.                                                //does element name match?
602.                                                if (FE(ElementName) == ElementState(ElementName) {
603.                                                            NameMatch = true}
604.                                                //does failure mode match?
605.                                                if (FE(FailureMode) == ElementState(ElementState) {
606.                                                            StateMatch = true}
607.                                                //both match?
608.                                                if (NameMatch == true && StateMatch == true) {
609.                                                            Match = true
610.                                                            //add to list
611.                                                            MatchingTokens.pushback(FE)
612.                                                }
613.                                    }
614.                                    //if no match was found minimal cut set cannot be valid
615.                                    if (Match == false) {CompleteMatch = false}
616.                        }
617.                        if (CompleteMatch == true) {
618.                                    //if match is found, update discoverability field
619.                                    For Each FE Token in MatchingTokens {
620.                                                FE(Discoverability) = true
621.                                    }
622.                        }
623.            }
624.            //for any Failed Element tokens not matched to a minimal cut set, set as undiscoverable
625.            For Each FE Token {
626.                        if (FE(Discovery) == null) {
627.                                    FE(Discovery) = false
628.                        }
629.            }
630. }
```

Figure 7-52 UpdateDiscoverability Arc Function.

The UpdateCrossingState arc function is defined in Figure 7-53. This function features the same triple nested loop as UpdateDiscoverability, which serves to match component failure tokens with minimal cutset tokens. When it finds a cutset which matches the failed elements present it must update the tokens in P3 which describe the current operational state of the crossing. This occurs over a nested loop in lines 655 to 670. The first loop iterates over each system effect caused by the minimal cutset. The second loop iterates over each token representing current active systems states. If the crossing is not already experiencing a given system failure effect, a token is added to place P3 describing the effect. This prevents

multiples tokens describing the same system effects caused by different minimal cutsets being added to place P3.

```
631. UpdateCrossingState() {
632.            //loop through each Minimal Cut Set token
633.            For Each MCS Token {
634.                        bool CompleteMatch = true
635.                        //loop through each Element State in the Minimal Cut Set token
636.                        For Each ElementState In MCS Token {
637.                                    bool Match = false
638.                                    //loop though each Failed Element token
639.                                    For Each FE Token {
640.                                                bool NameMatch = false
641.                                                bool StateMatch = false
642.                                                //does element name match?
643.                                                if (FE(ElementName) == ElementState(ElementName) {
644.                                                            NameMatch = true}
645.                                                //does failure mode match?
646.                                                if (FE(ElementState) == ElementState(ElementState) {
647.                                                            StateMatch = true}
648.                                                //both match?
649.                                                if (NameMatch == true && StateMatch == true) {
650.                                                            Match = true}
651.                                    }
652.                                    //if no match was found minimal cut set cannot be valid
653.                                    if (Match == false) {CompleteMatch = false}
654.                        }
655.                        if (CompleteMatch == true) {
656.                                    //if complete match found, loop through each system effect from
    minimal cut set
657.                                    For Each SystemState In MCS Token {
658.                                                Bool StateAlreadyActive = false
659.                                                //loop through each crossing state token present
660.                                                For Each State Token {
661.                                                            If (State(SystemState) == MCS(SystemState)) {
662.                                                                        StateAlreadyActive = true
663.                                                            }
664.                                                }
665.                                                //if a token matching the crossing state is not found, create one
666.                                                If (StateAlreadyActive == false) {
667.                                                            Output 1'State(Crossing=X, SystemEffects =
    SystemState)
668.                                                }
669.                                    }
670.                        }
671.            }
672. }
```

Figure 7-53 UpdateCrossingState Arc Function

7.3.1.3 Protection System Reporting Module

The Protection system reporting module enables the level crossing to self-report failures. The Petri net structure of this module can be seen in Figure 7-54. This module introduces place P5, this place contains all the urgent work orders generated for the crossings within the model. Should a crossing self-report a failure, a token is placed in P5. This token enables a later module to perform a repair.

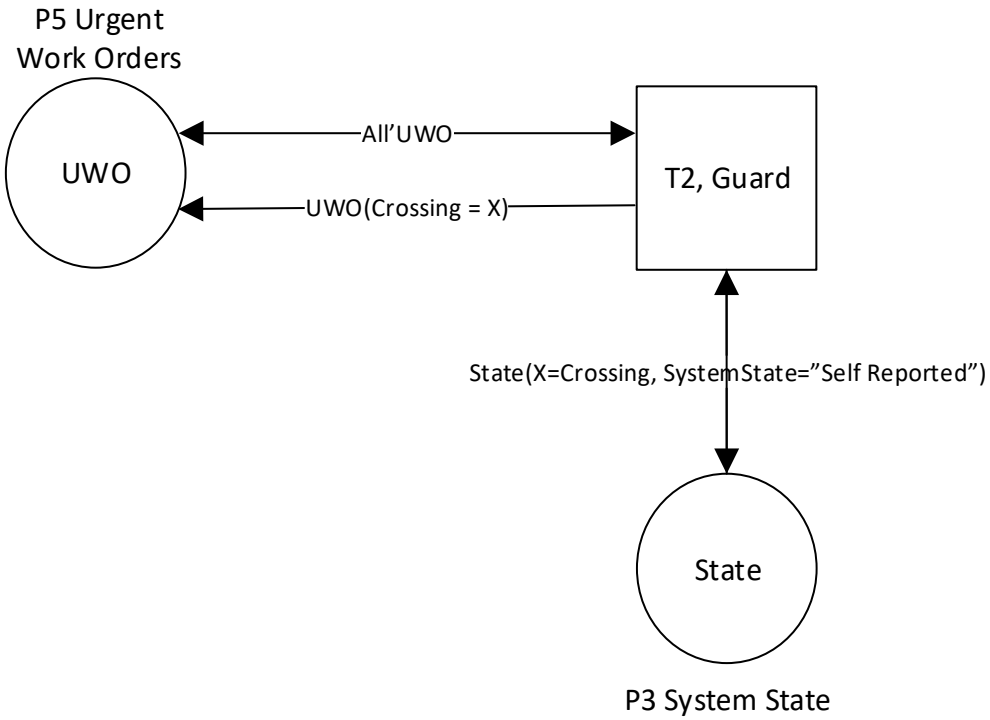

Figure 7-54 Self Reporting Module

The arc from P3 to T2 specifies only crossing's which are presently self-reporting a failure may enable transition T2. The guard function for T2 is defined in Figure 7-55. This function searches through all the tokens already present in place P5 to determine if an urgent work order has already been generated for the crossing. If it has, the transition will not be enabled as a duplicate work order would serve no purpose. If a work order is not present, the arc from T2 to place P5 creates one.

```
673. Colour UWO As Crossing
674. T4 Guard(X) {
675.            Bool UWOPresent = false
676.            //loop through each Urgent Work Order token present searching for one matching the
     selected crossing
677.            For Each UWO Token {
678.                    If (UWO(Crossing) == X) {
679.                            UWOPresent = true
680.                    }
681.            }
682.            //if a match is not found, enable a transition mode
683.            if (UWOPresent == false) {
684.                    Transition = Enabled
685.            }
686. }
```

Figure 7-55 T4 Guard Function and Colour UWO Definition.

7.3.1.4 Inspection Module

The inspection module allows crossing protection system failure states to be discovered via a regular inspection. The inspection module can be seen in Figure 7-57. Place P6 holds tokens describing the inspection requirements for each crossing modelled, defined in Figure 7-56.

```
687. Colour CrossingInpsectionState As Crossing x InspectionState
688. Colour Inspection State As String
```

Figure 7-56 Colour CrossingInpsectionState Definition.

The Colour CrossingInspectionState is a string which may be set to: “Not Required”, meaning there is currently no plan or need to inspect the crossing; “Opportunistic”, where the crossing may be inspected opportunistically; “Scheduled”, where an inspection has been scheduled to occur; and lastly, “Inspecting”, indicating an inspection is currently taking place.

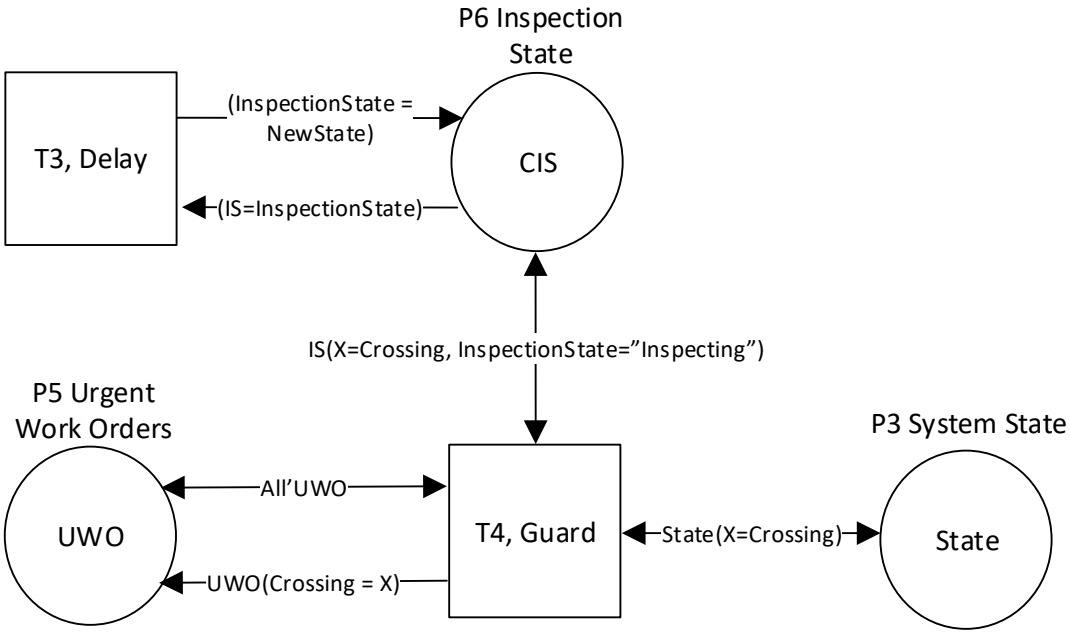

Figure 7-57 Inspection Module.

The function which controls this progression of inspection states is shown in Figure 7-58. The behaviour of the function depends on the current value of InspectionState within the tokens passed to the function. Lines 691 to 694 are executed if the inspection state is “Not Required”, in this case there is a firing delay of 28days, after which the InspectionState value will be updated to “Opportunistic” when T3 fires.

Now the same token will enable transition T3 again, with an InspectionState value of “Opportunistic”. This executes lines 695 to 708 of the function. Over the next 14 days an opportunistic inspection may take place negating the need to schedule one. An exponentially distributed random number with a mean of 28 days is generated on line 697. If this random number is less than 14, an opportunistic inspection will take place using the random number

as the firing delay. When the transition fires the InspectionState value will be set to "Inspecting". Should the random number be greater than 14, the transition firing delay will be se to 14, and the InspectionState value set to "Scheduled" when the transition fires.

```
689. T3 Delay(IS) {
690.            //find delay for transition mode, dependent on current Inspection State
691.            If (IS == "Not Required") {
692.                        Delay = 28
693.                        NewState = "Opportunistic"
694.            }
695.            If (IS == "Opportunistic") {
696.                        //generate exponentially distributed random variable for time till an opportunistic
    inspection occurs
697.                        OpportunisticDelay = ExponentialDistribution(28)
698.                        //if the delay is less than 14, an opportunistic inspection will occur
699.                        If (OpportunisticDelay < 14) {
700.                                    Delay = OpportunisticDelay
701.                                    NewState = "Inspecting"
702.                        }
703.                        //otherwise an inspection is scheduled after 14 days elapse
704.                        Else {
705.                                    Delay = 14
706.                                    NewState = "Scheduled"
707.                        }
708.            }
709.            If (IS == "Scheduled") {
710.                        //generate log normal distributed random variable for time till a scheduled
    inspection occurs
711.                        Delay = LogNormalDistribution(1.36,0.697)
712.                        NewState = "Inspecting"
713.            }
714.            If (IS == "Inspecting") {
715.                        //carry out 30minute inspection
716.                        Delay = 0.02
717.                        NewState = "Not Required"
718.            }
719. }
```

Figure 7-58 T3 Delay Function.

Tokens whose InspectionState value is set to "Scheduled" will cause lines 709 to 713 to be executed. Here, a random number is generated from a log-normal distribution to model the delay till the scheduled inspection occurs. After the transition fires the value of InspectionState is set to "Inspecting". The final portion of the function is executed when a token's InspectionState is "Inspecting", lines 714 to 718. The transition firing delay is set to 0.02 (30minutes), modelling the time for the inspection to complete. When the transition fires the token's InspectionState is set to "Not Required".

Whilst the token's InspectionState is set to "Inspecting", transition T4 may be enabled. This transition operates nearly identically to transition T2 which models self reporting. The guard function for T4 is the same as that used in T2. If any abnormal system states are present for the crossing an urgent work order will be generated.

7.3.1.5 Repair Crew Dispatch Module

This module controls the dispatch of maintenance crews to a level crossing to fix some issue. This typically occurs when a fault is self-reported, or discovered following an inspection.

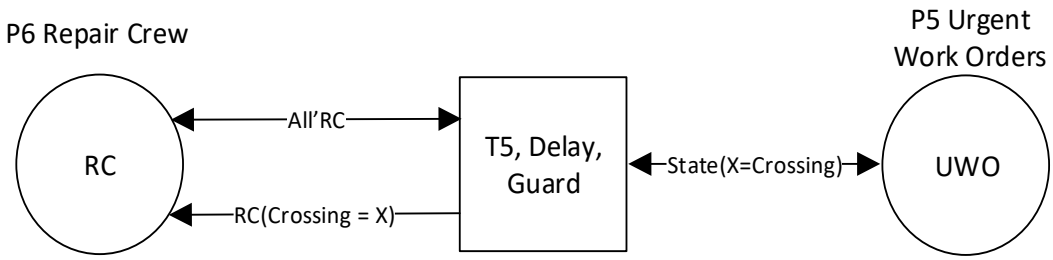

Figure 7-59 Repair Crew Dispatch Module.

The module is shown in Figure 7-59. Transition T5 operates very similarly to T2 and T4. It is enabled by the presence of any UrgentWorkOrder token in place P5. The guard function checks that a repair crew is not already present at the crossing. The transition delay function is shown in Figure 7-60, it simply generates a random number from a log-normal distribution to model the time till a repair crew arrives at the crossing. When the transition fires it places a token in place P6 signifying that a repair crew is now onsite.

```
720. Colour RepairCrew As Crossing
721. T5 Delay() {
722.            //generate log normal distributed random variable for time till a repair crew arrives
723.            Delay = LogNormalDistribution(-1.909, 0.145)
724. }
725. T5 Guard(All'RC, X) {
726.            //loop through each Repair Crew token
727.            For Each RC Token {
728.                        //if a repair crew is already at the crossing, disable the transition mode
729.                        If RC(Crossing == X) {
730.                                    Transition = disabled
731.                        }
732.            }
733. }
```

Figure 7-60 Colour RepairCrew Definition and T5 Delay Function.

7.3.1.6 Repair Module

The purpose of the repair module is fix crossing component failures. Only components with discoverable element failures can be replaced, i.e. those which are directly responsible for malfunctioning of the crossing. This is carried out by a single reset transition, T6, shown Figure 7-61. This in stark contrast to the regular Petri net model which required many reset transitions.

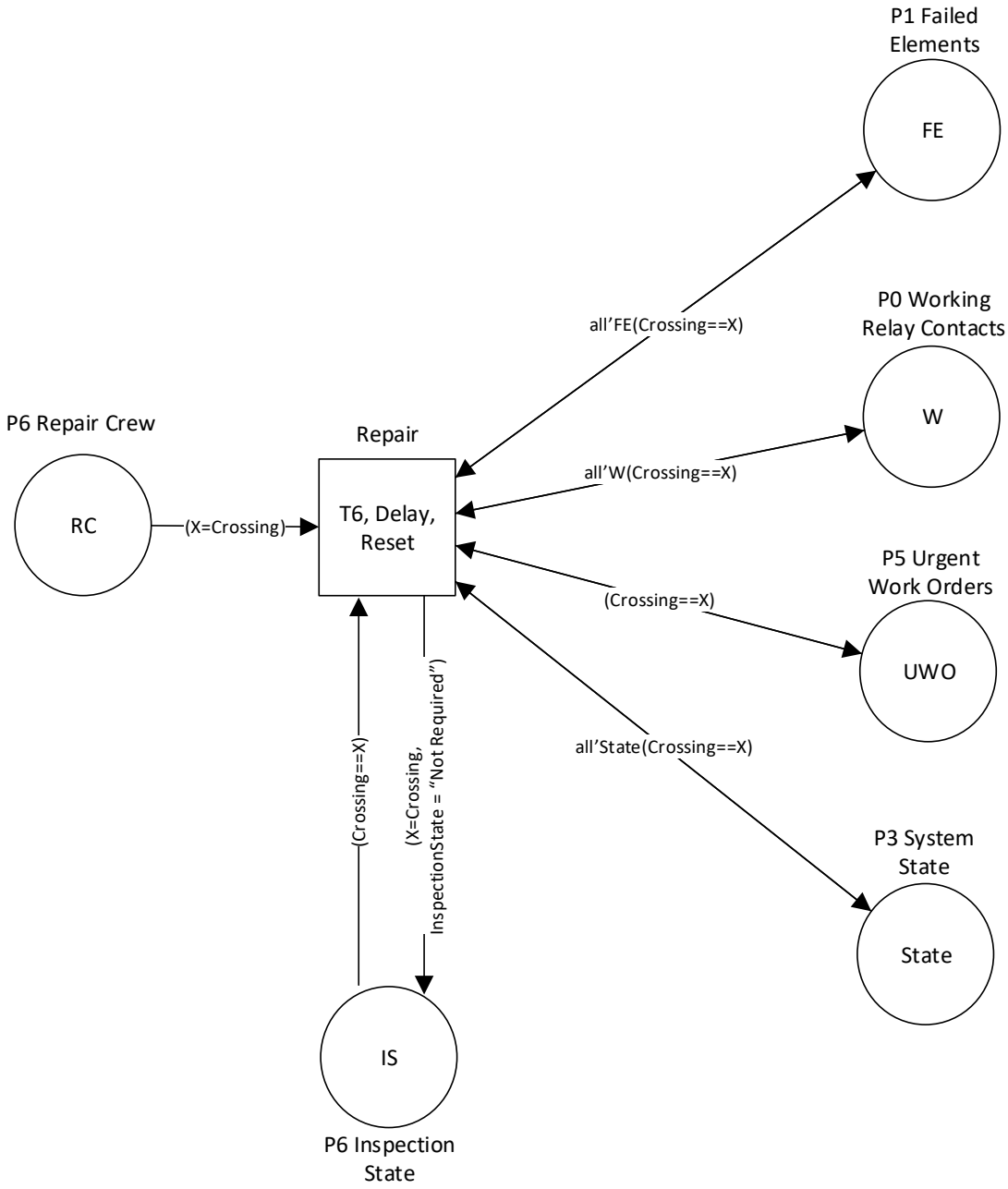

Figure 7-61 CCPN Repair Module

A transition mode is enabled by the presence of a token in place P6, meaning that a repair crew is ready and waiting to repair a crossing. Transition T6 uses a delay defined in Figure 7-62. This delay function sums the repair time of each discoverable component failure and returns this value as the delay time. Between lines 738 and 753, the function iterates over each element failure token, and on line 740 checks if the failure is discoverable. Recalling that components contain multiple elements, but are replaced as whole components, a loop between lines 743 and 747 checks if the component the failed element resides in has already been added to the replaced list. If not, line 750 adds the component to the list. Finally, line 755 calculates the repair time used for the transition's delay as the number of relay's

requiring replacement plus a fixed time for diagnosis and preparation. Each replaced relay requires 40minutes to replace, as per the SPN model.

```
734. T6 Delay(all'FE) {
735.        //create a string array to store the names of all failed components
736.        String Array FailedComponentsList
737.        //loop through each Failed Element token
738.        For Each FE Token {
739.                //is the failed element discoverable?
740.                If (FE(Discoverable) == true) {
741.                        Bool InList = false
742.                        //check it's not already in the list
743.                        For Each String In FailedComponentsList {
744.                                If (FE(ComponentName) == String) {
745.                                        InList = true
746.                                }
747.                        }
748.                        //if not, add to the list
749.                        If (InList = False) {
750.                                FailedComponentList.push_back(FE(ComponentName))
751.                        }
752.                }
753.        }
754.        //calculate the repair time from size of the list
755.        Delay = 0.0225 + FailedComponentsList.size/36
756. }
```

Figure 7-62 T6 Delay Function.

When the delay elapses the transition mode fires and runs its reset function, defined Figure 7-63. This function restores all discoverable component failures in a given crossing to their original working state and resets the protection system states to working. The function first builds a list of all components which will be replaced, over lines 759 to 775. This is the same code which was used in T6's delay function. Next, over lines 777 to 790, the function iterates over each failed element token comparing it against the list of components to be replaced. If the element resides within a component that is being replaced, it is restored to the place P0 which holds the working elements and removed from P1.

```
757. T6 Reset(All'FE, All'W, All'UWO, All'State, All'RC) {
758.        //create a string array to store the names of all failed components
759.        String Array FailedComponentsList
760.        //loop through each Failed Element token
761.        For Each FE Token {
762.                //is the failed element discoverable?
763.                If (FE(Discoverable) == true) {
764.                        Bool InList = false
765.                        //check it's not already in the list if not, add component name to the list
766.                        For Each String In FailedComponentsList {
767.                                If (FE(ComponentName) == String) {
768.                                        InList = true
769.                                }
770.                        }
771.                        If (InList == False) {
772.                                FailedComponentList.push_back(FE(ComponentName))
773.                        }
```

```
774.                    }
775.          }
776.          //loop through each Failed Element token
777.          For Each FE Token {
778.                    //loop through each  component name in the list
779.                    For Each String In FailedComponentsList {
780.                              //check for matches
781.                              If (FE(ComponentName) == String) {
782.                                        //Found a failed element to be repaired!
783.                                        //create token to be added to working place
784.                                        Create.Token.W(ComponentName = FE(ComponentName),
    ElementName = FE(ElementName), Crossing = FE(Crossing))
785.                                        Add Token to P0
786.                                        //remove Failed Element token
787.                                        Remove FE from P1
788.                              }
789.                    }
790.          }
791.          //reset the time till failure on all otherwise working elements in replaced components
792.          For Each W Token {
793.                    For Each String In FailedComponentsList {
794.                              If (W(ComponentName) == String) {
795.                                        Reset Delay
796.                              }
797.                    }
798.          }
799.          //remove Urgent Work Order token for crossing
800.          For Each UWO Token {
801.                    If (UWO(Crossing) == X) {
802.                              Remove UWO from P5
803.                    }
804.          }
805.          //remove crossing failure state tokens
806.          For Each State Token {
807.                    If (State(Crossing) == X) {
808.                              Remove State from P3
809.                    }
810.          }
811.          //remove repair crew from crossing
812.          For Each RC Token {
813.                    If (RC(Crossing) == X) {
814.                              Remove State from P3
815.                    }
816.          }
817. }
```

Figure 7-63 T6 Reset Function.

As entire relays are replaced, some working relay contacts will be replaced despite not having failed. To account for this, lines 792 to 798 iterate through all working relay contacts to determine if they are in a relay which is being replaced. If a match is found the transition mode firing delay is re-sampled. Lines 800 to 816 search for and remove tokens for the specific crossing from places P3, P5 and P6 representing the restoration of the crossing system, the completion of the work order and the work crew leaving.

7.3.1 Colour Petri net Simulation Software

As with the stochastic Petri net, software was written specifically for this work to simulate colour Petri nets. The core simulation loop for the CPN software is shown in Figure 7-64.

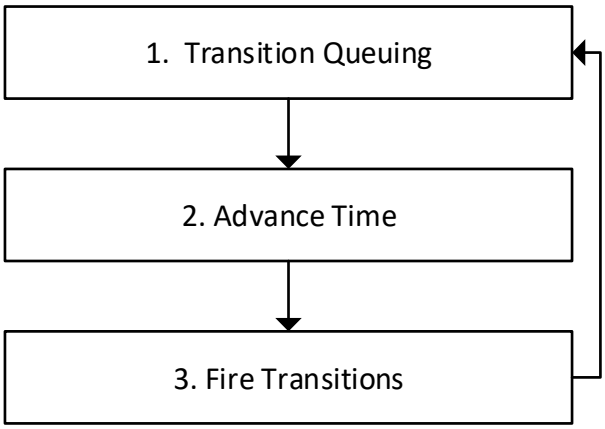

Figure 7-64 Colour Petri Net Simulation Loop.

Each function is briefly described:

1.	This function tests each transition to determine if it is valid to fire. It first extracts all sets of tokens which meet the specified incoming arc conditions, these are evaluated against any guard conditions for the transition. If met and not previously queued, a transition mode will be queued to fire. Any previously enabled transition modes are checked to determine if they are still valid by checking all tokens are still present and the guard function conditions are still met.
2.	The next function, Advance Time, finds the next transition mode queued to fire. This is achieved by simply iterating through each transition mode and selecting that with the smallest remaining delay. No optimisations have been applied here, as the list of transition modes is typically small, and comparatively little computational time is spent running this function compared to the others.
3.	The last function fires queued transition modes. It first evaluates if a previously queued transition mode is still valid to fire. Should this be true, it removes the incoming tokens which enabled the transition mode and evaluates the outgoing transition arc functions.

This is much simpler than that for the SPN. This is because many of the optimisations which improved SPN simulation cannot be applied to CPN simulation. The conditions which enable a CPN transition are variable and potentially complex functions, as are the guard functions

and output arc functions. The optimisations applied to the SPN simulation relied upon the rigidly defined input and output arcs of an SPN. As such, it is difficult to apply any optimisations to the simulation of the CPN as a whole.

7.3.2 Comparison of Computational Requirements

Figure 7-65, shows the CPN memory requirement for increasing numbers of level crossings assets simulated. This is compared with the results of the SPN model which uses the same set of modules and logically identical. Its shows the CPN uses much less memory, and is thus able to simulate models containing up to 70,000 crossings for a budget of 5GB of RAM. In comparison, for the same amount of RAM the equivalent SPN model was only able to simulate 5,000 crossings. This was achieved by minimising the structure of the CPN. The SPN has a rigid structure where each transition has defined inputs and defined outputs requiring more memory to store, whereas the CPN has little defined structure and instead must search to determine its inputs and outputs.

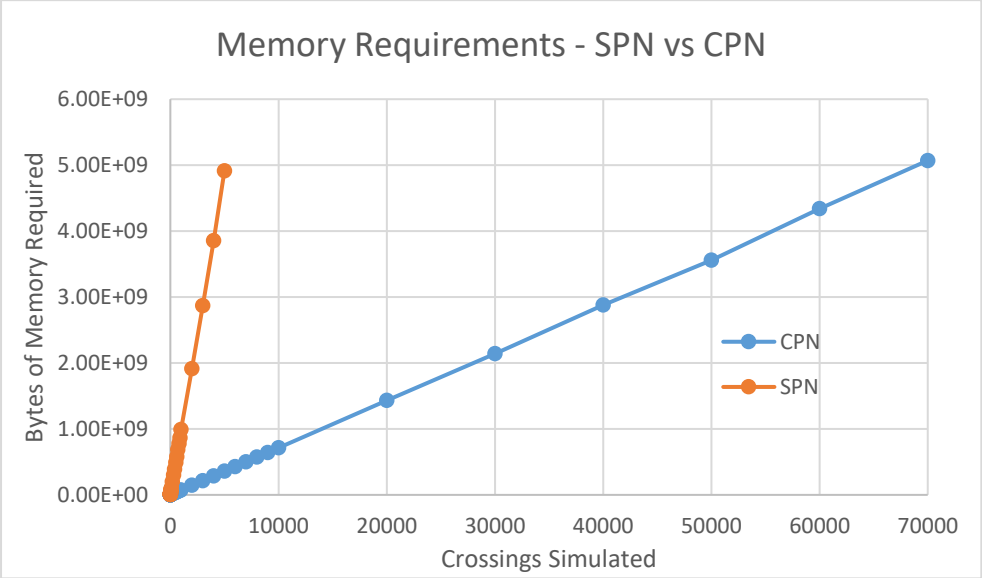

Figure 7-65 Memory Requirement Comparison between SPN and CPN Models

To further reduce the memory requirements of this model, the strings stored within tokens were replaced with integer values which can be mapped to strings, known as enumeration. The results of this are shown in Figure 7-66. Each additional crossing modelled using this method then requires only 23KB of additional programme size per crossing modelled, where previously 72KB was needed.

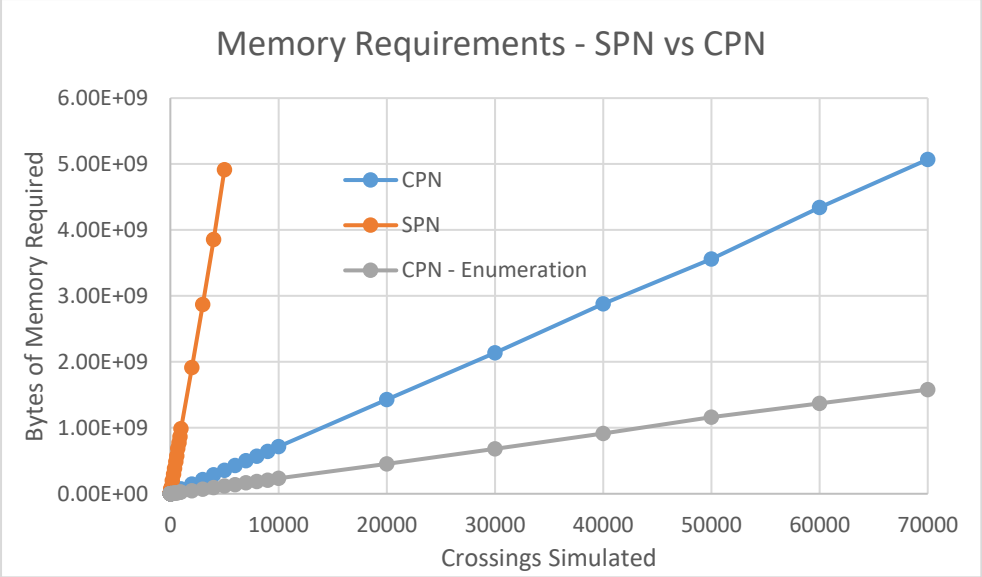

Figure 7-66 Updated Memory Requirement Comparison.

However, a different outcome is observed when computational time is considered. Figure 7-67 shows the computation times for a single simulation for increasing numbers of level crossing assets within each model. The CPN is much slower, showing an exponential increase in computation time. This is because of the loosely defined structure of the CPN. It must search through each token in each place to find the sets of tokens which enable a transition, some transitions then have to search to determine what actions are performed when they fire. As the number of crossings increases, so does the number of tokens which must be searched through, creating an exponential increase in computation time

In contrast, the SPN need only perform simple comparison operations on the places which enable a transition followed by addition and subtraction operations. Many of these transitions can be ignored entirely on any given simulation loop by use of the optimisations described earlier. Such optimisations are not possible for the CPN model due to its loosely defined structure.

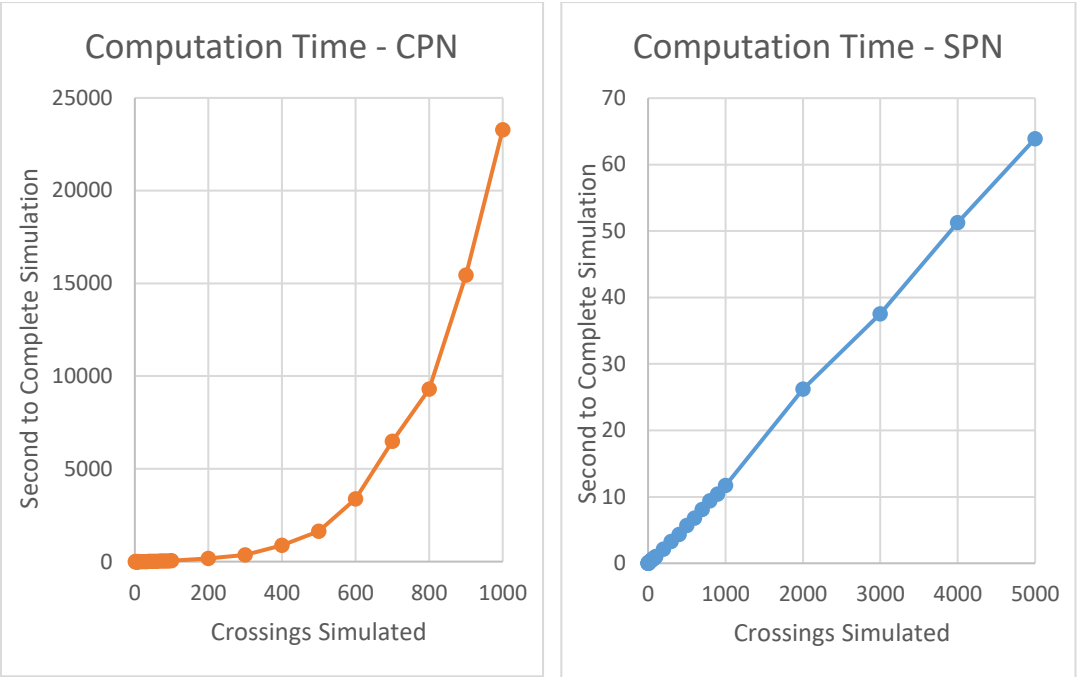

Figure 7-67 Comparison of Computation Times for SPN and CPN models.

The poor simulation speed of this methodology appears to due to its inherent lack of structure preventing the implementation of any meaningful optimisations. Rebuilding with additional structure will require greater memory usage, but also facilitate optimisation. This is the aim of the next Petri net methodology presented.

7.4 Simple Colour Petri Net

The Simple Colour Petri Net (SCPN) was developed for this work and is intended to be a balance between the high memory requirement of the SPN and the high computation requirement of the CPN. It is identical to the SPN, however does not require duplication of the SPN for each additional asset. Each level crossing model added to the SPN contains large amounts of duplicated structure, the SCPN only adds the data that is needed to simulate the additional crossing, using the original crossing Petri net as a template to be referred to as required. It is in effect a colour Petri net with a single colour identifying the asset the token belongs to. However it is simulated in a manner which is in between an SPN and a CPN. For comparison, the model was built using the same modules at the SPN and CPN.

The nature of the differences between the simulation and scaling of the SPN and SCPN models are illustrated in Figure 7-68. The top portion of the figure shows how the SPN methodology would model the same process for three different crossing assets by simply duplicating the entirety of the Petri net. The SCPN achieves the same outcome by creating sub-places and sub-transitions for each place and transition. Sub-places hold tokens, sub-transitions store their firing times, however all other information such as arc relationships

and transition firing delays are stored within the higher level place and transition. In a normal CPN these sub-transitions would not be explicitly defined, and the sub-places would instead take the form of token colours.

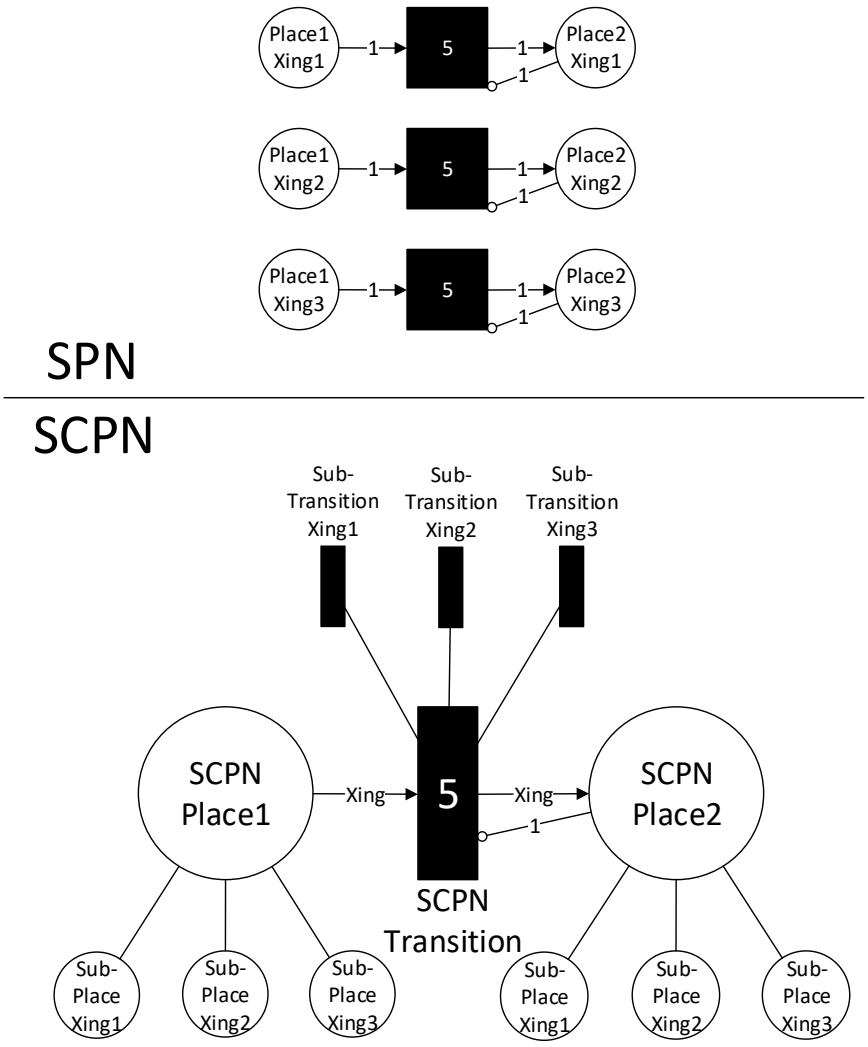

Figure 7-68 Diagram Illustrating Differences between SPN and SCPN Models.

When implemented in software, the SCPN will treat each sub transition as if it were a normal SPN transition. The higher level transition acts as a template, which must be referenced to determine if the sub-transition is enabled, its firing delay and the effects of it firing. This required only minor changes to the SPN software to enable it to model the SCPN, retaining all the previously described optimisations. This was made possible because its structure allows the creation of a list of all the potential transition modes which could be enabled, and the exact conditions which would enable them.

7.4.1 Simple Colour Petri Net Computational Requirements

The SCPN used much less memory than the SPN it was derived from, and approximately twice as much memory as the CPN, a graph of which is shown in Figure 7-69. Each additional

crossing requires 61 KB of memory. The SCPN simulation software uses a similar data structure as that described for the SPN. The vectors for place – transition mappings remains unchanged, as are the optimisation tables. The vector storing the number of tokens held per place was expanded so each sub-place's tokens can be stored. A similar expansion was used to store the firing delay for each sub-transition.

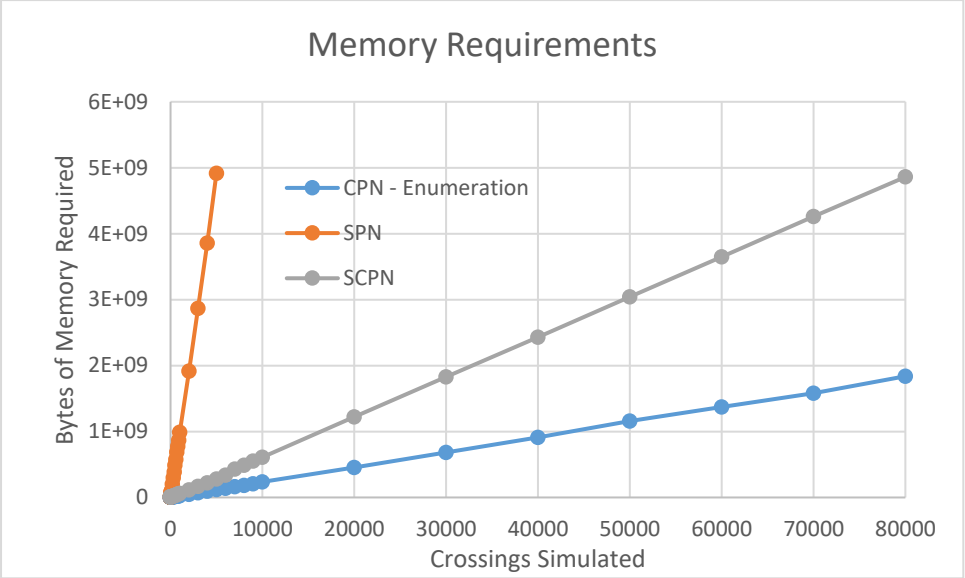

Figure 7-69 Memory Requirement Comparison of all Models.

The computational cost of the memory reduction was found to be acceptable. Simulations using the SCPN took twice as long as the SPN, the result of which can be seen in Figure 7-70. The linear increase in computational cost continued ultimately allowing 100,000 level crossing assets to be simulated simultaneously for 50 years, taking just under 70 minutes for the computation of a single simulation to complete.

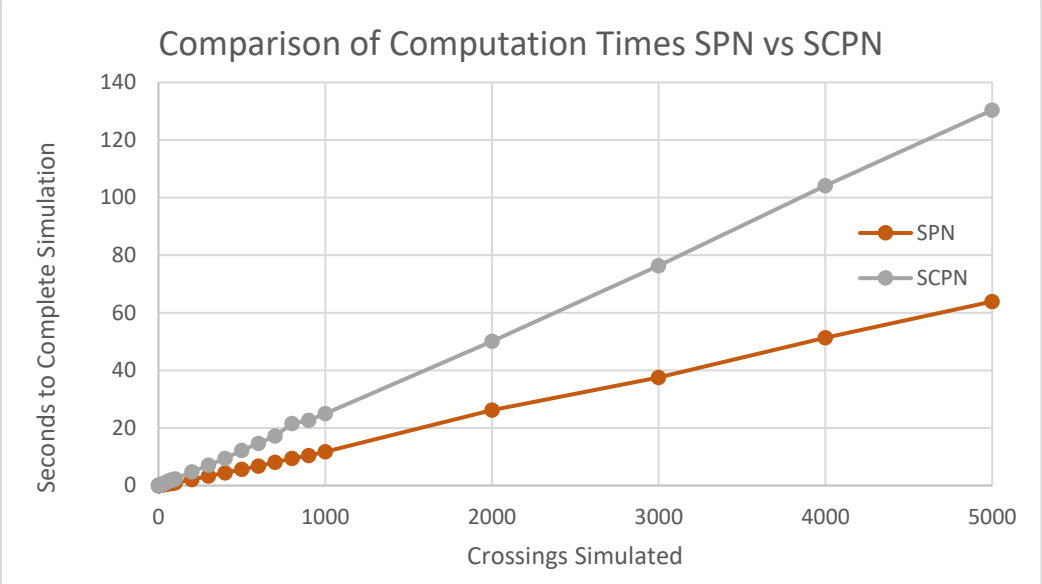

Figure 7-70 Comparison of Computation Times between SPN and SCPN.

7.5 Conclusions

This chapter has presented a model for asset management of level crossings. Computation times required for simulating increasing numbers of level crossing assets simultaneously were determined. The stochastic Petri net model was simulated faster than the other methodologies explored, but the computer memory required to do so limited this model to no more than 5,000 level crossing assets, assuming contemporary desktop computer memory limits. This is a large number of level crossings, more than realistically would be simulated in a route level model. However, the total number of assets which may be simulated in a route level model could easily exceed this number. Thus the methodology should be able to comfortably simulate this number of assets and greater.

A colour Petri net model was designed to minimise the amount of memory required to simulate each additional level crossing asset with the goal of pushing beyond the 5,000 crossing asset limit. The result of this was a model with dramatically reduced memory requirements, but a similarly dramatic increase in computational time. This was attributed to the inability to optimise the simulation of the model due to its lack of structure.

The simple colour Petri net model was created to strike a balance between the colour Petri net model and the stochastic Petri net model. It achieved this, enabling up to 100,000 crossing assets to be simulated on contemporary computer hardware within a reasonable timeframe. This methodology has the added benefit of being compatible with stochastic Petri nets models in literature without modification, as this is achieved in software alone.

It must be acknowledged that the validity of these results depends upon the relative efficiency of the software authored during this work to simulate each Petri net methodology. Despite this, the results obtained were as would be expected. Stochastic Petri nets require duplication of entire Petri nets for each asset and thus have a higher memory requirement than colour Petri nets which do not. Colour Petri simulation is complex, and thus slower to simulate than the simple operations that comprise simulation of a stochastic Petri net. And finally, the simple colour Petri net which was designed to balance between to the other two was shown to do exactly that.

8. Modelling Shared Resources

Single asset models frequently use stochastic models to simulate the time until maintenance occurs. Where the delay for a component to degrade is generated stochastically, and the delay for that degraded component to be repaired is also generated stochastically. The degradation delay is often modelled as such as it is not feasible to predict when railway components will fail. The delay between discovery and repair of a degraded component depends upon factors such as the availability of spare parts, engineering crews and rail track possessions. Single asset models typically have a narrow scope and therefore these factors are unknown, necessitating the use of a stochastic delay.

Whole asset models can have a much wider scope, where all assets on a rail line could be modelled. Under these circumstances, the factors which affect the time until maintenance occurs can be included within the model. If all the assets for a region are modelled, then the volume of work orders for that region can be included within the model. Combined with a method to model the work load of maintenance crews, and availability of spares, the time until maintenance begins need no longer be determined by generation of a random number. Instead, the movement and activity of maintenance crews as they complete work orders can be modelled to determine when a maintenance activity begins. Earlier literature review highlighted that such an asset management model has not yet been developed.

Using a framework, asset model templates can be used to rapidly create models for each asset. Assets of the same design, or class will share core features enabling a single template to be applied to many assets. Templates may be calibrated to specific assets by using unique statistical distribution reflecting their environmental conditions, and through the inclusion of optional extensions to the template to reflect structural differences between otherwise similar assets, such as the provision of additional road traffic lights in an AHBC.

The primary advantage of such a model is that it allows exploration of the effects of staffing and spares availability on asset management. A model which uses a stochastic variable to model the time until maintenance begins relies upon either an assumed distribution or one taken from maintenance scheduling data. Whilst this data may be available, it is only truly descriptive of the asset management strategy in place at the time, and the volume of maintenance activity required at that time. This limits the capacity of a model to explore asset management plans which directly affect scheduling and the availability of maintenance resources. This chapter explores how Petri nets can be used to create a whole asset model

using a scheduling system which models maintenance crews shared between large numbers of assets.

Maintenance engineers will be the sole shared resource modelled. This is because all maintenance tasks in some way depend upon the availability of engineers to carry out or oversee them. As such, modelling the availability of engineers is core to achieving the aim outlined above. The logistics for these engineers is also explored, as the travel time from their present location to a failed asset could have a significant effect on asset management key performance indicators if that asset is affecting service provision. Travel time is also significant when modelling maintenance actions which are carried out relatively quickly, such as inspections.

8.1 Timed Petri Net Model for Scheduling

As there are no stochastic elements required for the scheduling system alone, the stochastic Petri net devolves to a timed Petri net (TPN) with high level extensions. This is backwards compatible with stochastic Petri nets as only stochastic transitions are removed.

An example of a TPN modelling only the movement of a maintenance crew to and from assets from a base location has been created, shown Figure 8-1. The tokens in place P1 represent the number of available maintenance crews. The tokens in places P2 and P5 indicate that both assets in this example have faults requiring a maintenance crew to resolve. This will enable both transitions T2, and T3, modelling the maintenance crew's travel time to each asset. As Asset1 is closest, transition T3 will fire, resulting in this asset being repaired first. After repairs are completed, transition T4 returns the maintenance crew to the depot, where they can be dispatched to repair Asset2.

This system of repairing the closest assets to the maintenance depot maximises the number of assets which can be repaired during times where there is insufficient staff resources to carry out all repairs by minimising travel time. This is not intentional, but a convenient property of Petri nets.

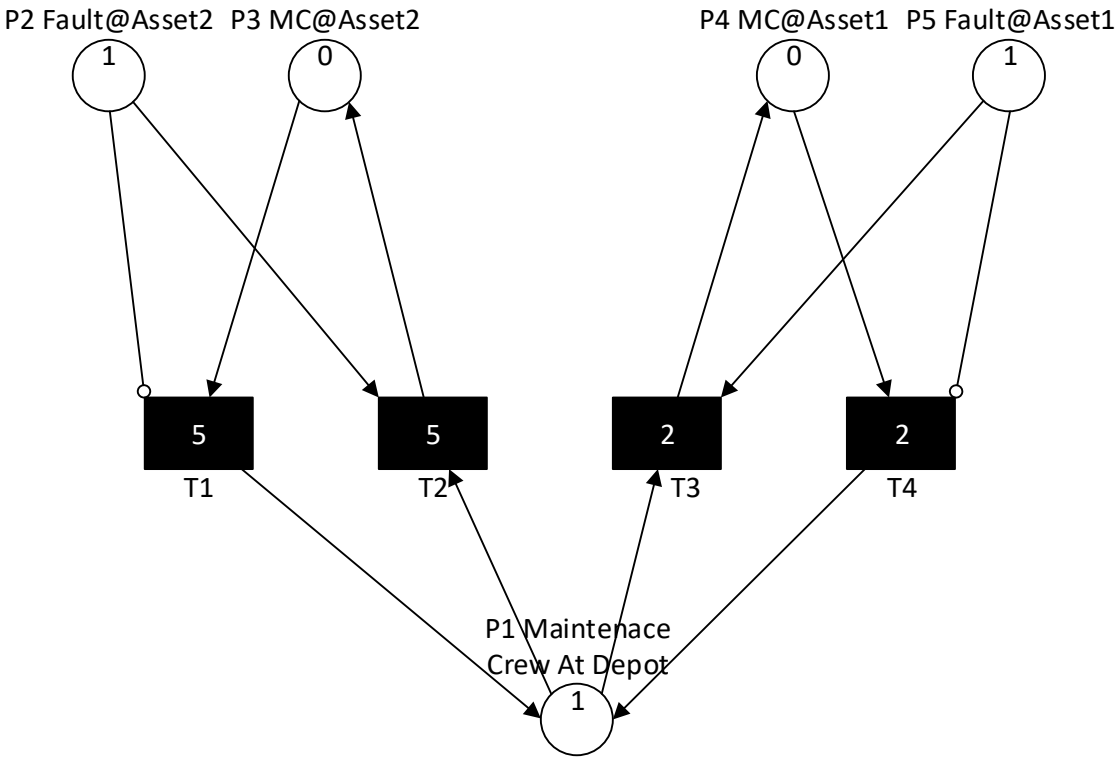

Figure 8-1 Staff Travelling to, and Returning From Assets from a Base Location Implemented Using a TPN

Different levels of urgency for work orders are explored next. A TPN to demonstrate how this can be achieved is shown in Figure 8-2. This example expands on the previous one by adding a place indicating the level of urgency of the work order. There is a pair of transitions for each asset and level of urgency modelling the movement of maintenance crews. Transitions 3 and 4 model urgent work orders for asset2, with Transitions 5 and 6 modelling routine work orders. Should an urgent work order not be fulfilled in a timely manner, perhaps because it is located far from other assets, there is a further pair of transitions, T1 and 2 which model overdue work orders. This type of work order has the highest priority. For Asset1, urgent work orders are modelled by T9 and 10, routine work orders by T7 and T8, and overdue urgent work orders by T11 and 12.

Simulating the net as drawn, both assets require a maintenance crew. Asset2 has the higher priority work order, indicated by the token in place P3. An inhibitor arc from this place disables transition T7 which would move a maintenance crew to carry out a routine repair on Asset1. Thus Asset1's urgent work order is carried out before the routine work order for Asset2. Should both assets have equal priority work orders, the closest is carried out first.

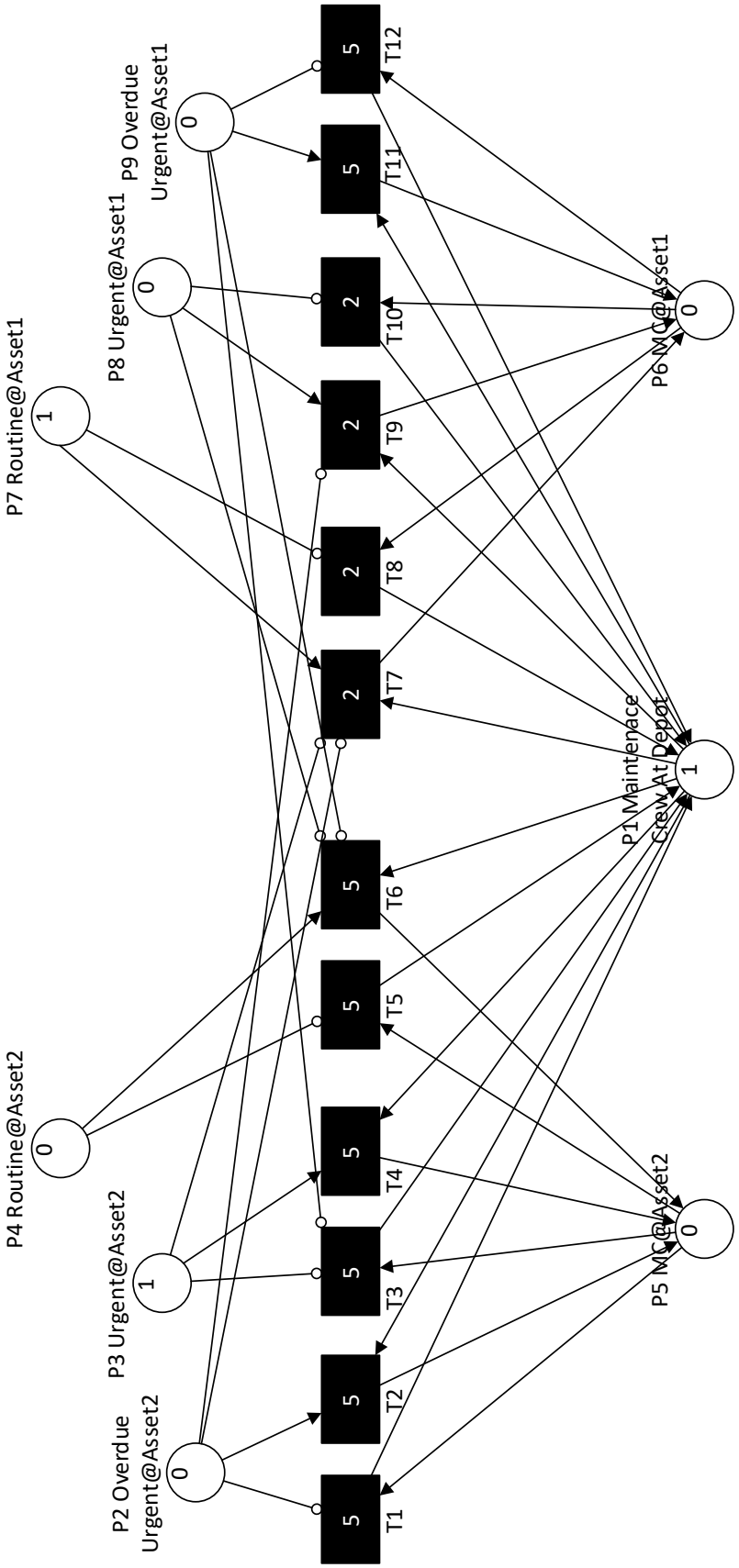

Figure 8-2 Staff Travelling to, and Returning From Assets from Base Location Implemented Using a TPN with Priority System.

When scaling this system to include further assets, each urgent work order place, such as P2, must have an inhibitor arc disabling all transitions which fulfil lower priority work orders. More precisely, any transition of lower priority which has the effect of moving a repair crew further away from the asset must be inhibited. This is illustrated in Figure 8-3, it shows the permissible movement paths for the model as described so far. There is an urgent work order present at Asset 1. The red arrows show movement paths which will be inhibited, and the green arrows movement paths which are permissible.

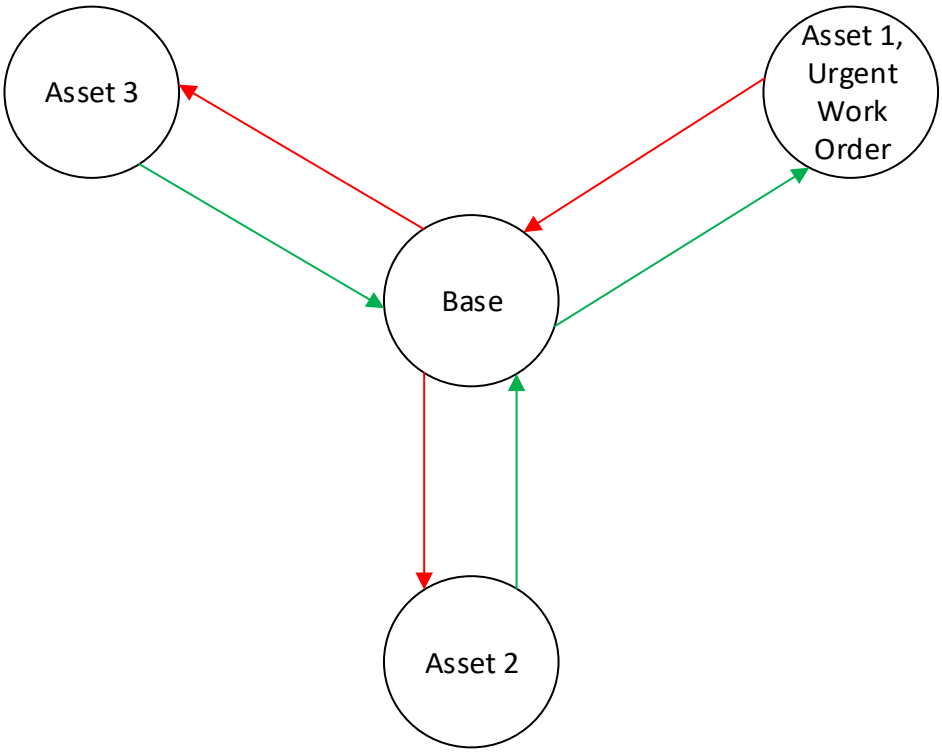

Figure 8-3 Permissible Maintenance Crew Movement Paths for Return to Base Configuration.

There must be an inhibitor arc from each Urgent Work Order place to each transition which facilitates moving a repair crew away from a more urgent work order. The number of pathways increases with each additional asset.

$$Total\ Travel\ Routes = 2A$$
$$Where\ A = Number\ of\ Assets\ in\ Model$$

8-1

Each additional asset introduces two further pathways for maintenance crews to travel.

$$Routes\ Away\ from\ Specific\ Asset$$

$$= Total\ Travel\ Routes - Routes\ Toward\ Specific\ Asset$$

$$Routes\ Away\ from\ Specifc\ Asset = 2A - A = A$$

8-2

However, only one of these routes will move a maintenance crew away from an asset with an urgent work order. The total number of inhibitor arcs required is the product: of the number of routes away from a given asset; the number of assets; and the number of inhibitor arcs required per asset.

$$Total\ Inhibitor\ Arcs = A * A * 3$$

8-3

For the return to base configuration described so far, this a quadratic function. This presents a computational issue, as the memory requirement increases quadraticly with the number of assets. Assuming an inhibitor arc requires three integers to store: place number; transition number; and multiplicity, and that each integer requires 4bytes of memory, the scaling of the memory requirement is shown in Figure 8-4.

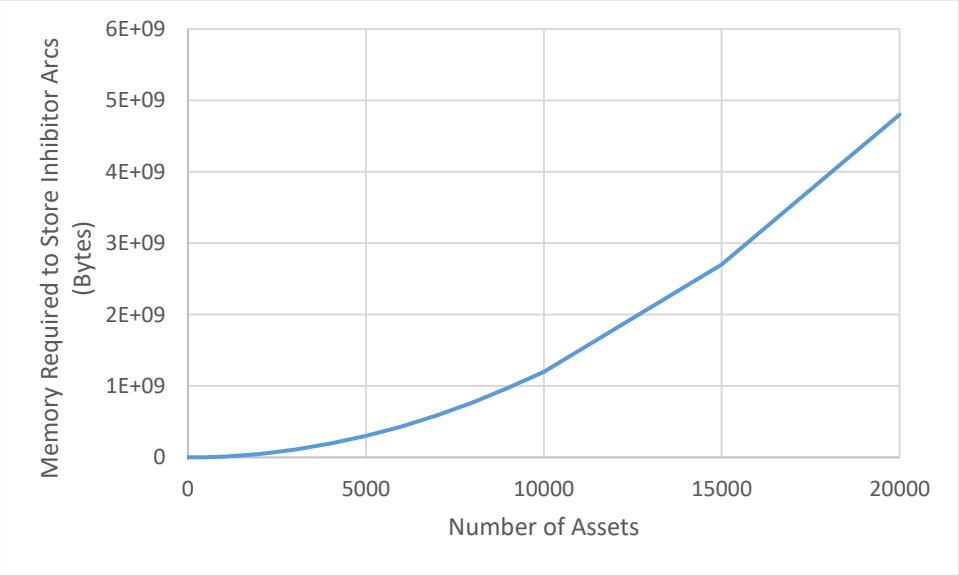

Figure 8-4 Inhibitor Arc Memory Requirement for Return to Base Configuration.

This limits the number of assets which can be modelled on contemporary desktop hardware to around 20,000 assets before the memory requirement becomes a problem. However, this considers only movement of maintenance crews to assets via a base. A maintenance crew may be required to move from any asset, to any other asset in carrying out their duties. Such

movement is most relevant when inspecting assets, where an engineer will travel from asset to asset spending little time at each.

An example of asset to asset movement paths is shown in Figure 8-5. Once again, the effects of an urgent work order for Asset 1 are represented in the colour of the movement path arrows, red indicating a path which will be inhibited, and green a permissible path. There are many more asset to asset paths, a higher proportion must be inhibited compared to the return to base configuration.

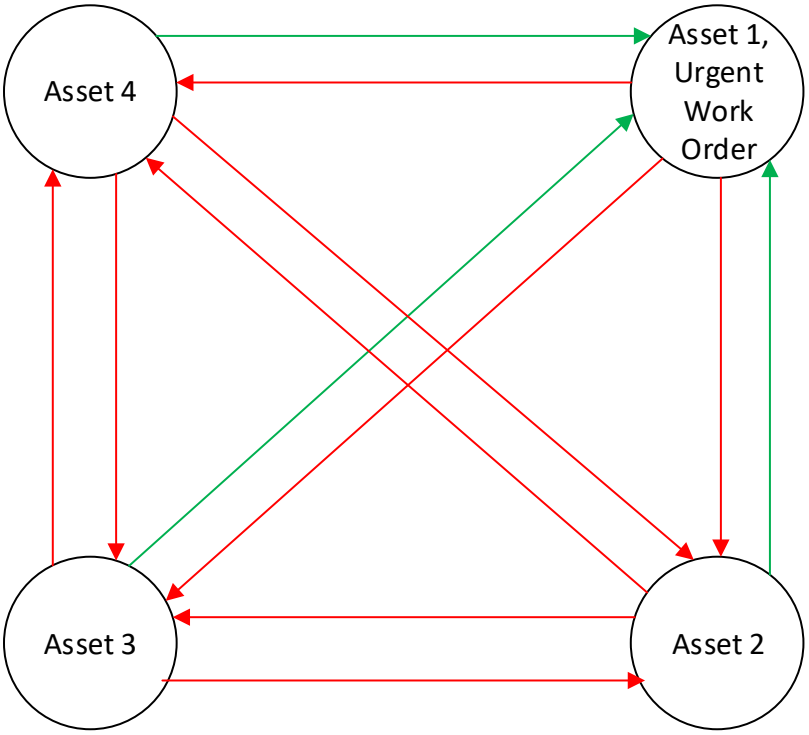

Figure 8-5 Permissible Maintenance Crew Paths for Asset to Asset Configuration.

The asset to asset configuration creates a quadratic increase in the number of paths between assets for maintenance crews as the number of assets increases. This ultimately results in a cubic increase in the number of inhibitor arcs required with increasing asset numbers.

$$Total\ Travel\ Routes = A(A-1)$$
$$Where\ A = Number\ of\ Assets\ in\ Model$$

8-4

$$Routes\ Away\ from\ Specific\ Asset = Total\ Travel\ Routes - Routes\ Toward\ Specific\ Asset$$
$$Routes\ Away\ from\ Specific\ Asset = A(A-1) - (A-1)$$

8-5

$$Total\ Inhibitor\ Arcs = 3A * (A(A-1) - (A-1))$$

8-6

Figure 8-6 shows the effect of increasing the number of assets on the inhibitor arc memory requirement for the asset to asset configuration. Due to the cubic increase in inhibitor arcs there is a much steeper increase in the memory requirement. This limits its use to around 700 assets only, again assuming contemporary desktop computer hardware.

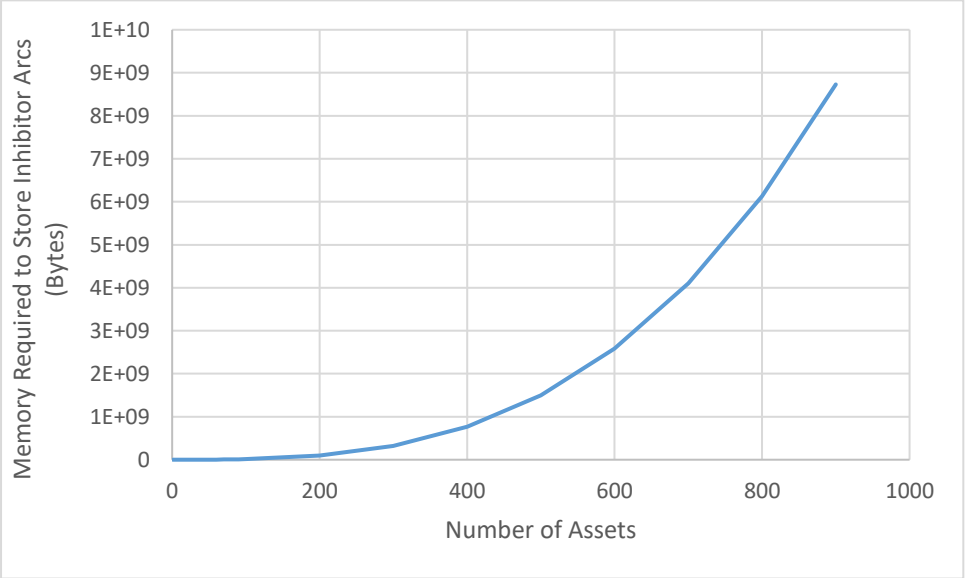

Figure 8-6 Inhibitor Arc Memory Requirement for Asset to Asset Configuration

The memory requirement of the asset to asset configuration can be reduced by introducing a hierarchy. At the top of the hierarchy assets can be grouped by location into regions, the lower levels occupied by sub regions, and the lowest levels by the actual assets. This is illustrated in Figure 8-7. In this configuration, inhibitor arcs do not need to be applied from each asset to every movement path which does not bring a maintenance crew to the specific asset. Instead, inhibitor arcs need only be applied to each asset within the sub region.

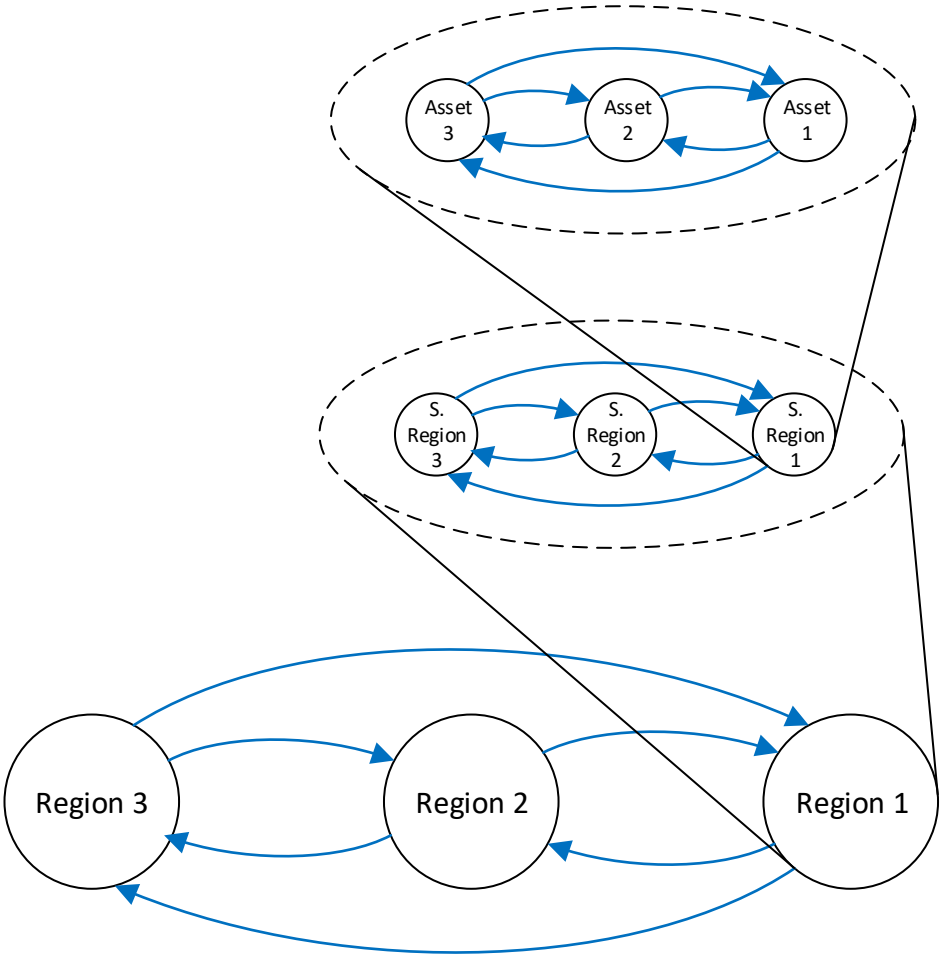

Figure 8-7 Movement paths for Asset to Asset Hierarchy.

This improves the memory scaling, shown Figure 8-8, allowing around 20,000 assets to be simulated using contemporary desktop hardware. This assumes a three level hierarchy comprising four regions, four sub-regions per region, with assets on the third level. Whilst this greatly improves the computational cost, it is still quite limiting in terms of the maximum number of assets compared to the stochastic approach used in the previous chapter.

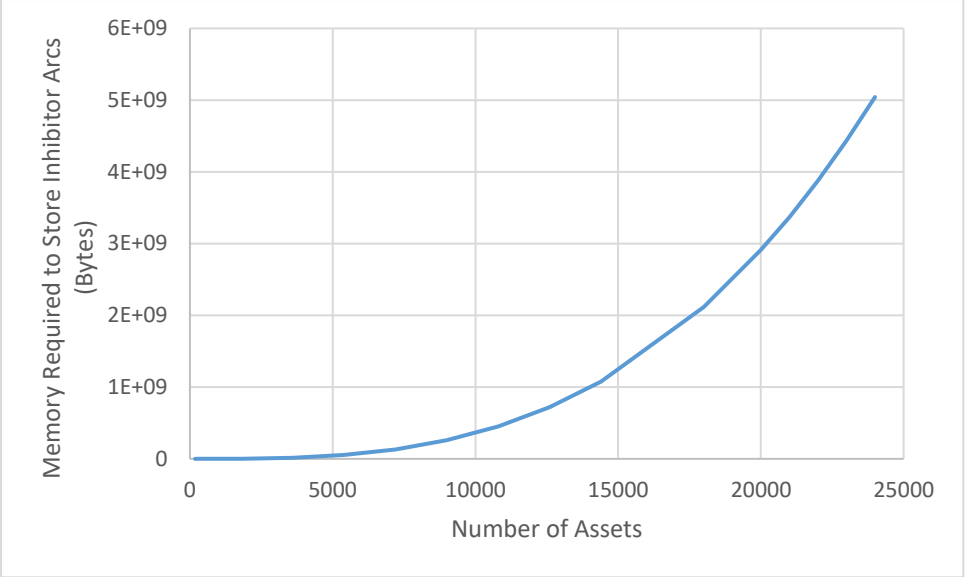

Figure 8-8 Inhibitor Arc Memory Requirement for Asset to Asset Hierarchy Configuration

However, it is not without downsides. Using such a hierarchy necessitates that a maintenance crew must travel through each level in the hierarchy to arrive at an asset. This means asset to asset travel requires the following movements: Asset to sub-region, sub-region to region, region to new region, new region to new sub region, new sub-region to new asset. As illustrated in Figure 8-9, this can result in a very indirect path when moving between assets in different regions, despite their being geographically close. This may be inaccurate if there is easy access between assets, resulting in the modelled time to move between assets being much higher than reality.

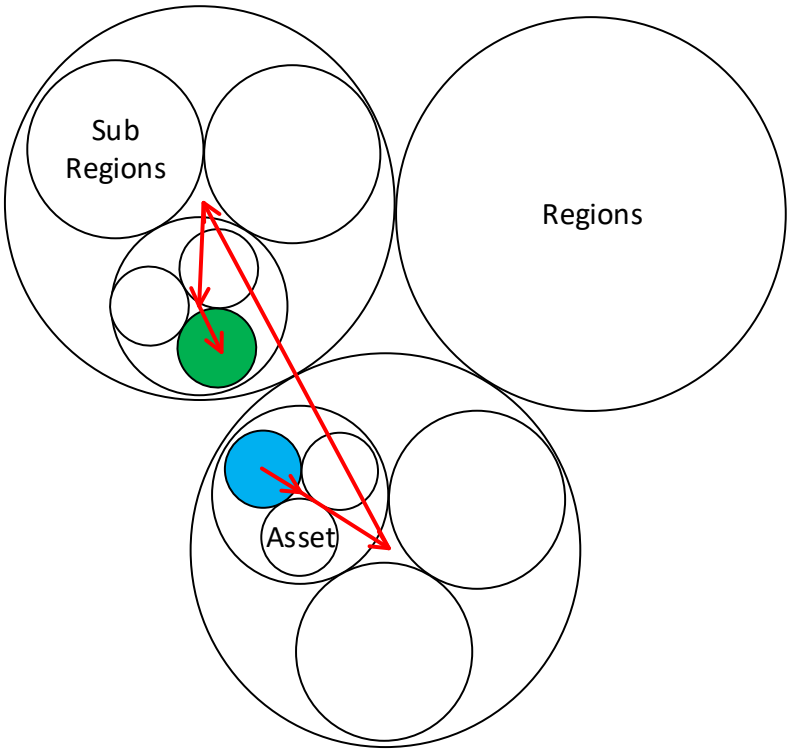

Figure 8-9 Movement between Regions in Asset to Asset Hierarchy Model.

This is not the only way to structure a TPN to carry out movement of maintenance staff. In the asset to asset example, maintenance crews moved directly from where they were to where they were needed. An alternatively structured TPN can be considered where staff move between each asset. Moving from their current location to the desired location, via each asset in-between. This takes advantage of rail assets being concentrated along lines.

A simple example of this is shown in Figure 8-10. It shows how maintenance crews may be modelled moving through each asset in a single direction to arrive at an asset with a fault. In this example, Asset3 has a fault indicated by a token in place P1. A maintenance crew is at Asset1, indicated by a token in place P9. This will cause transition T1 to fire, placing a token in P5. This further enables T4 to fire, placing a token in place P6. Now transition T6 is enabled, this delay transition models the time for the crew to move between Asset1 and Asset2. It fires, consuming the tokens in P9 and P6. Transition T5 fires modelling the same process, moving the crew between Asset2 and Asset3, allowing maintenance to begin.

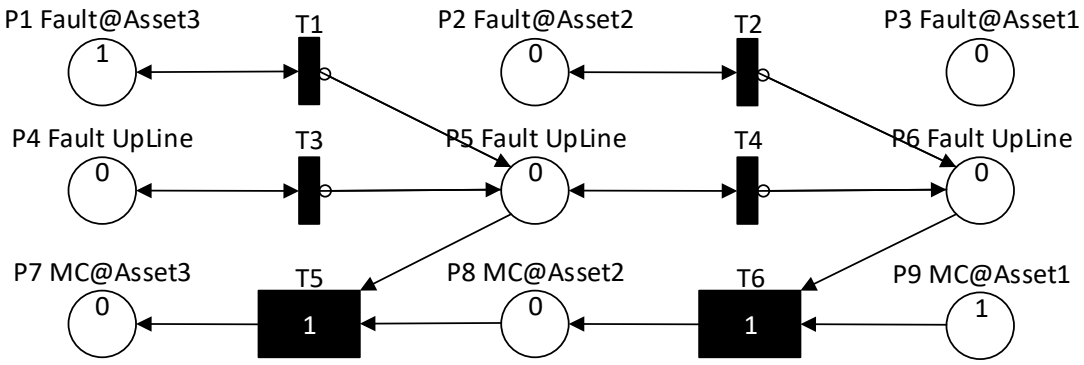

Figure 8-10 Alternative Inter Asset Movement TPN.

This system model is more restrictive and underdeveloped compared to the previous TPN, allowing only one way movement of a maintenance crew for a single level of work order priority. However, it can already been seen that whilst this will only require a linear increase in TPN size as additional assets are added, it will result in a large increase in the number of transitions which must fire for a maintenance crew to travel to an asset. This TPN structure is therefore not suitable either.

Three TPNs have been explored for modelling the movement of maintenance crews, however none have been found which are acceptable. The return to base model ignores asset to asset movement which whilst acceptable for lengthy work orders is unsuitable for quick routine work orders where several assets are likely to be visited before the crew returns to base. The asset to asset model has an unacceptable memory requirement, and therefore can be discounted. The inter asset movement model, though undeveloped, appears to trade an excessive memory requirement for an excessive computational requirement.

The most suitable TPN model explored was the asset to asset hierarchy model. Movement issues notwithstanding, it had what might be considered an acceptable memory requirement as presented. However, it lacks basic features such as control of maintenance crew shifts, such as being able to recall a maintenance crew when their shift is over. Implementing such a feature will require additional TPN structure, pushing the memory requirement even higher. Coloured Petri nets, with their enhanced feature set may be able to improve upon the TPN models, and will be explored next.

8.2 Coloured Petri Net Model for Deterministic Scheduling

Having determined that TPNs are not sufficient for a resource sharing system, a Coloured Petri Net deterministic scheduling model was developed and is presented below.

8.1.1 Work Orders

The first part of this Petri net concerns the generation of urgent work orders, shown Figure 8-11. Work orders are generated within the level crossing asset model and are passed to the CPN model for fulfilment.

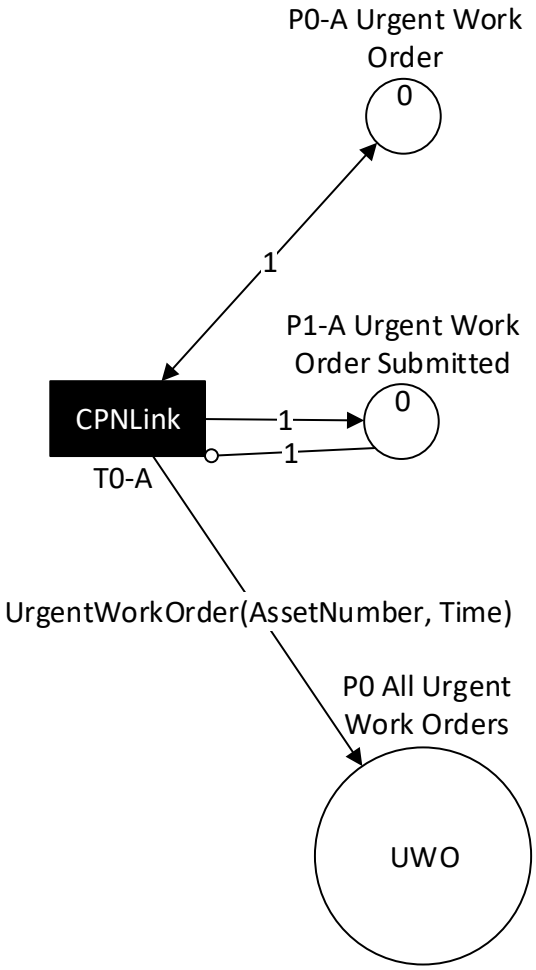

Figure 8-11 Generation of Urgent Work Orders.

When an urgent work order is generated for an asset, a token is created and placed in place P0-A. This enables transition T0-A, which is a special transition connecting the asset Petri net with the scheduling CPN. When this transition fires it creates an Urgent Work Order token and places it in place P0 of the scheduling CPN. This token's colours are defined in Figure 8-12, comprising: a unique reference number for the asset which generated the work order, and the generation time.

```
818. Colour UrgentWorkOrder As AssetNumber x GenerationTime
819. Colour AssetNumber As Integer
820. Colour GenerationTime As Double
```

Figure 8-12 Urgent Work Order Token Colours.

Transition T0-A also places a token in the asset model's place P1-A which prevents the transition from firing a second time till the urgent work order has been fulfilled. Only one urgent work order per asset can be created using this system. It is assumed that all urgent works are completed in one repair session, as per the single asset Petri net models presented in previous chapters.

The next part of the CPN scheduling model allocates the closest available engineer to fulfil a work order at an asset. This is achieved using three new token colours defined in Figure 8-13. The token colour Asset Information holds information on the position of each asset relative to the engineer's base. The token colour Engineers contains the position of the engineer, the asset it is/was last at, what they are currently doing (travelling to an asset, carrying out a work order, returning to base, or off duty) and the start and end times of their shift. The Assigned Urgent Work Order token colour contains a mixture of the asset information colours and the engineer's colours, in addition to the travel time of the engineer to the asset.

```
821. Colour AssetInformation As AssetNumber x XPosition x YPosition
822. Colour AssignedUrgentWorkOrder As AssetNumber x XpositionAsset x YpositionAsset x
     GenerationTime x AssignmentTime x XpositionEngineer x YpositionEngineer x TravelTime x
     EngineerNumber
823. Colour GenerationTime As Double
824. Colour ArrivalTime As Double
825. Colour Engineers As Xposition x Yposition x Location x Status x ShiftStart x ShiftEnd
826. Colour Xposition As Double
827. Colour Yposition As Double
828. Colour Location As Integer
829. Colour Status As Integer
830. Colour ShiftStart As Double
831. Colour ShiftEnd As Double
```

Figure 8-13 Asset Information, Assigned Urgent Work Orders and Engineers Token Colours.

The CPN structure which assigns engineers to urgent work orders is shown in Figure 8-14. There are two conditions which must be met to enable transition T0. Any Urgent Work Order token in place P0 fulfils the first condition. The arc from P0 to T0 sets the variable AN to be asset number for the urgent work order. The arc from P1 to T0 uses variable AN to select the appropriate token holding information about that asset, it sets variable XPA and YPA to the X and Y co-ordinate of the asset respectively.

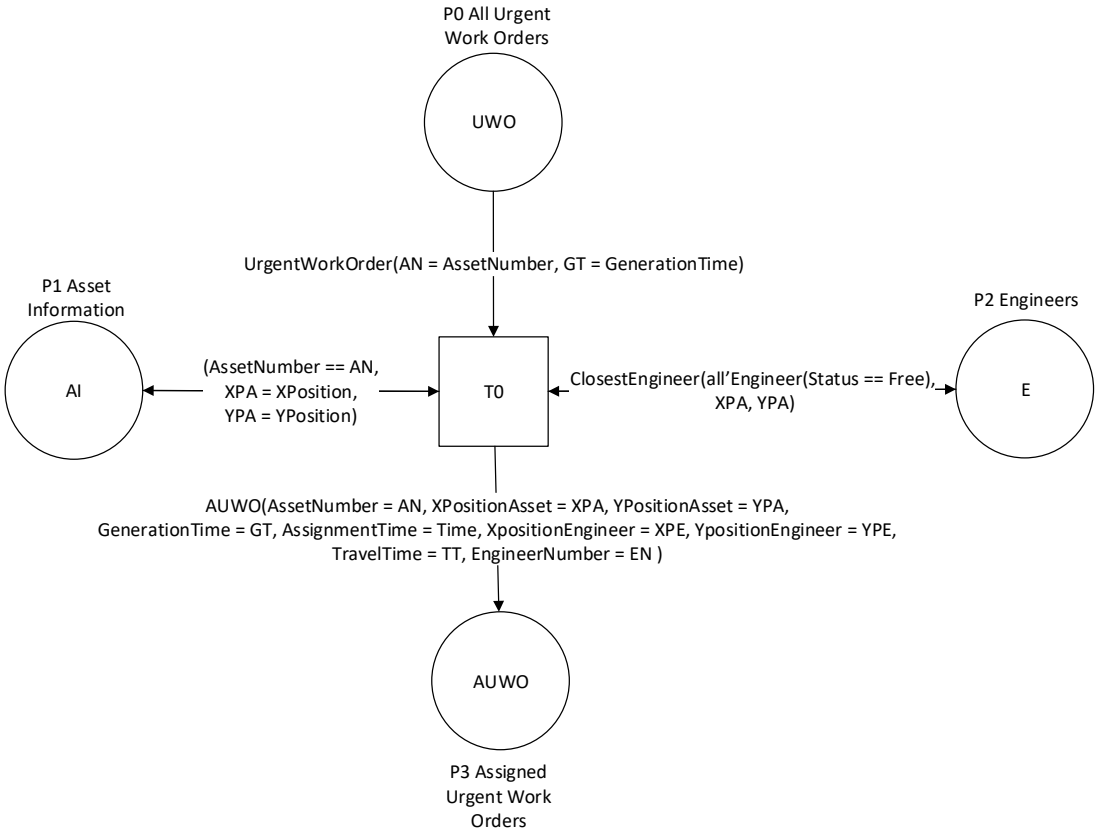

Figure 8-14 Assignment of Engineers to Urgent Work Orders.

The second condition comes from the function within the arc from P2 to T0. This function calculates the distance to the asset for each available engineer, selecting the engineer closest to the asset. If no engineers are available, the transition will not be enabled. This function is detailed in Figure 8-15. The function accepts all engineer tokens within P2 whose status is set to 'free', and the X and Y asset position variables set by the arc from P1 to T0.

Lines 840 to 853 calculate the distance from each engineer to the asset using Pythagoras theorem, using a comparison operator to track the closest engineer as the loop progresses through each engineer. A comparison operator on line 855 checks if any free engineers were found, if true, line 857 calculates travel time assuming an engineer moves at 50kph. Lines 858 to 861 set and return variables to be used by the final arc from T0 to P3. Lastly, line 862 changes the engineer's status to assigned, preventing it from being assigned simultaneous work orders.

```
832. ClosestEngineer(all'Engineer(Status == Free), XPA, YPA) {
833.            //create a pointer to refer to the token for the closest engineer to the crossing
834.            TokenPointer ClosestEngineer
835.            Double ClosestEngineerDistance = -1
836.            Double EngineerDistance
837.            //create variable for travel time to be passed out of function
838.            Double TT
839.            //loop through each Engineer token to find closest
840.            For Each Engineer Token
841.                        //calculate distance to crossing asset
842.                        EngineerDistance = ((Engineer(Xposition) - XPA)^2 + (Engineer(Yposition) -
    YPA)^2)^0.5
843.                        //if first engineer token, distance is closest by default
844.                        if (ClosestEngineerDistance = -1) {
845.                                    ClosestEngineerDistance = EngineerDistance
846.                                    ClosestEngineer = Engineer
847.                        }
848.                        //if less that current closest distance, update
849.                        if (ClosestEngineerDistance < EngineerDistance) {
850.                                    ClosestEngineerDistance = EngineerDistance
851.                                    ClosestEngineer = Engineer
852.                        }
853.            Next Engineer
854.            //if any free engineers were found
855.            if (ClosestEngineer != null) {
856.                        //create and set variables for other arc functions
857.                        TT = (ClosestEngineerDistance / 50)/24
858.                        Integer EN = EngineerNumber
859.                        Double XPE = Xposition
860.                        Double YPE = Yposition
861.                        Return TT, EN, XPE, YPE
862.                        ClosestEngineer(Status = Assigned)
863.            }
864.            else {Transition = disabled}
865. }
```

Figure 8-15 Description of Function to Find Closet Engineer to Asset.

The last arc from T0 to P2 creates a new token of type 'Assigned Urgent Work Orders' using the variables created from the previous three arcs, and adds it to place P3. The next portion of the CPN models the engineer travelling to the asset and initiates the repair process, the structure of which is shown in Figure 8-16.

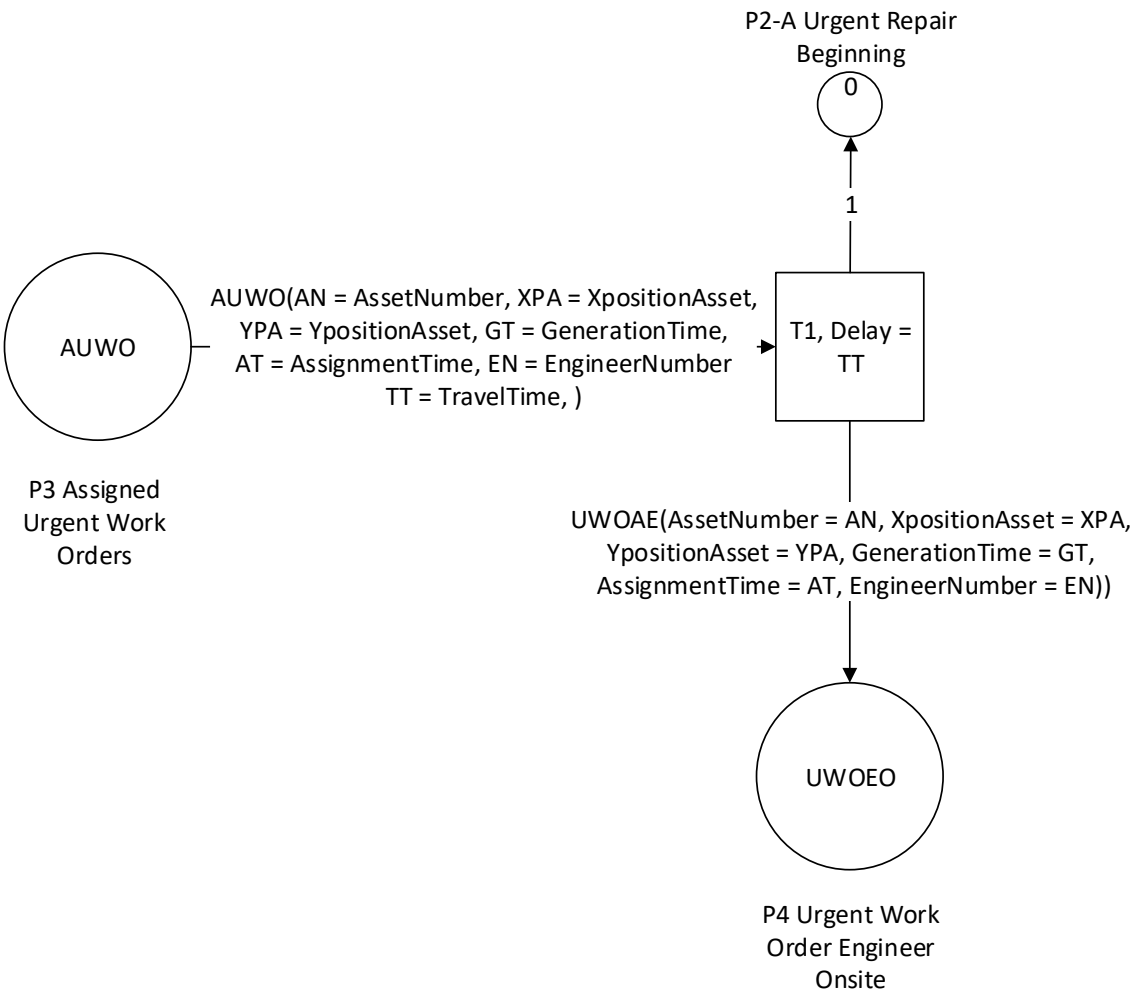

Figure 8-16 Modelling Travel Time of Engineers to Assets.

Any tokens in place P3 will enable a transition mode of T1. The arc from P3 to T1 assigns many variables, which are later used to create the new token being passed to P4. One variable created is TT, set to the TravelTime colour. This is passed to transition T1 and is used as the transition mode's firing delay, to model the time for the engineer to travel to the asset. When T1 fires it places a normal token in place P2-A of the appropriate level crossing asset model, allowing the repair process itself to be modelled by the asset Petri net. A new type of token is then generated using the variables set earlier and placed in P4. This token type is named Urgent Work Order Engineer Onsite, and is defined in Figure 8-17.

```
866. Colour UrgentWorkOrderEngineerOnsite As AssetNumber x XpositionAsset x YpositionAsset x
     GenerationTime x AssignmentTime x EngineerNumber
867. Colour CompletedUrgentWorkOrders As AssetNumber x CompletionTime
868. Colour CompletionTime As Double
```

Figure 8-17 Urgent Work Order Engineer Onsite and Completed Urgent Work Order Token Colours

The repair process is modelled within the asset model. When it is completed the asset Petri net adds a token to the scheduling CPN to show that the work is completed. This is shown in

Figure 8-18. When the repair is completed within the asset model a token is added to place P3-A, signifying that the asset is being brought back into service, and the maintenance crew are free to perform other tasks.

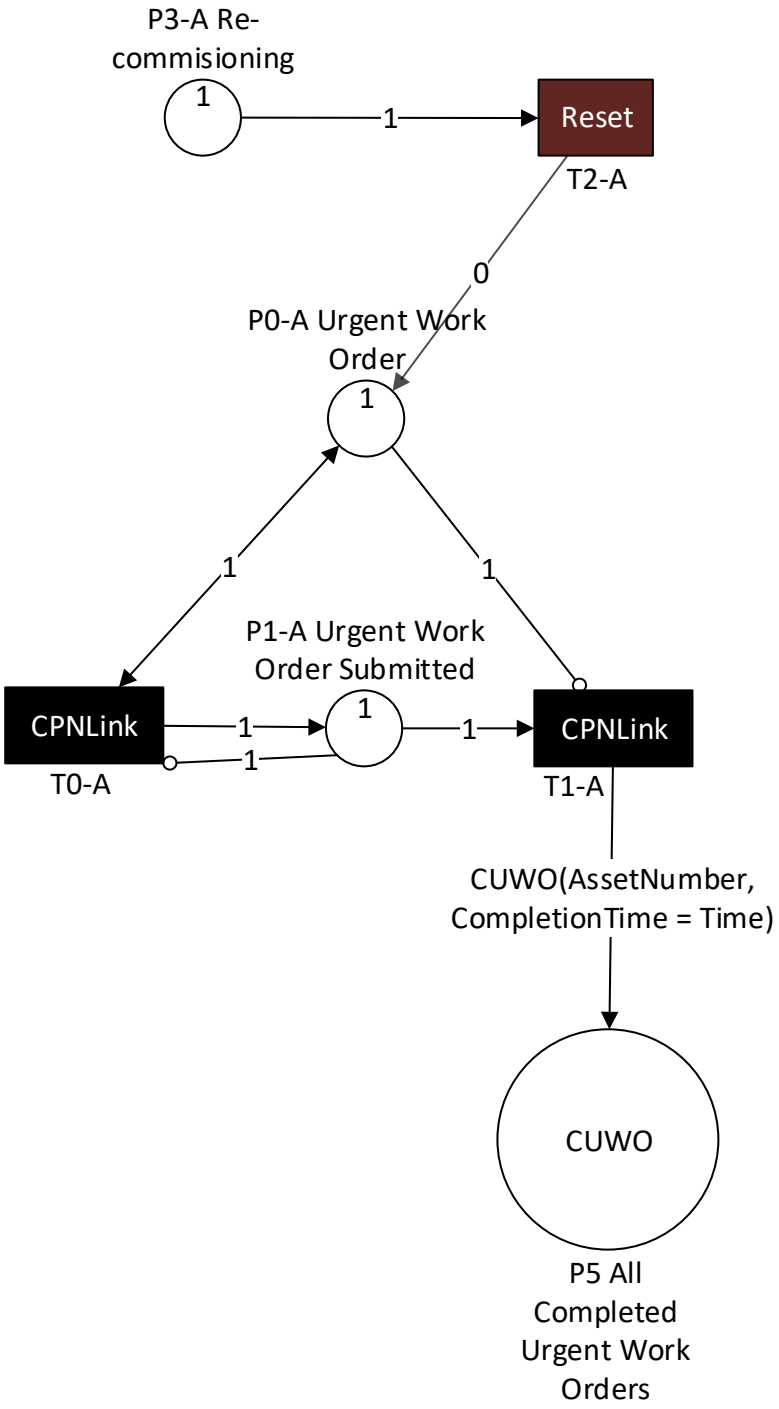

Figure 8-18 Completion of Urgent Work Orders in Asset Model.

This enables transition T2-A, which performs various functions on the asset Petri net model, but of importance here is that it removes the token in P0-A. This allows transition T1-A to

fire, adding a 'Completed Urgent Work Order' token to CPN place P5. This contains the asset number and the time the work order was completed, defined in Figure 8-19.

```
869. Colour CompletedUrgentWorkOrder As AssetNumber x CompletionTime
870. Colour CompletionTime As Double
```

Figure 8-19 Token Colour Completed Urgent Work Order.

Finally, this allows the engineer to become idle, where upon they can be assigned a further work order or return to base. The CPN structure for this is shown in Figure 8-20. Each token representing a completed urgent work order in place P5 enables a mode of transition T2. The arc from P5 to T2 assigns an asset number to the variable AN within the token. There will exist an equivalent token in place P4 for the same asset, the arc from P4 to T2 specifies the token must have the same asset number. It also sets the variable EN to refer to the engineer's unique ID.

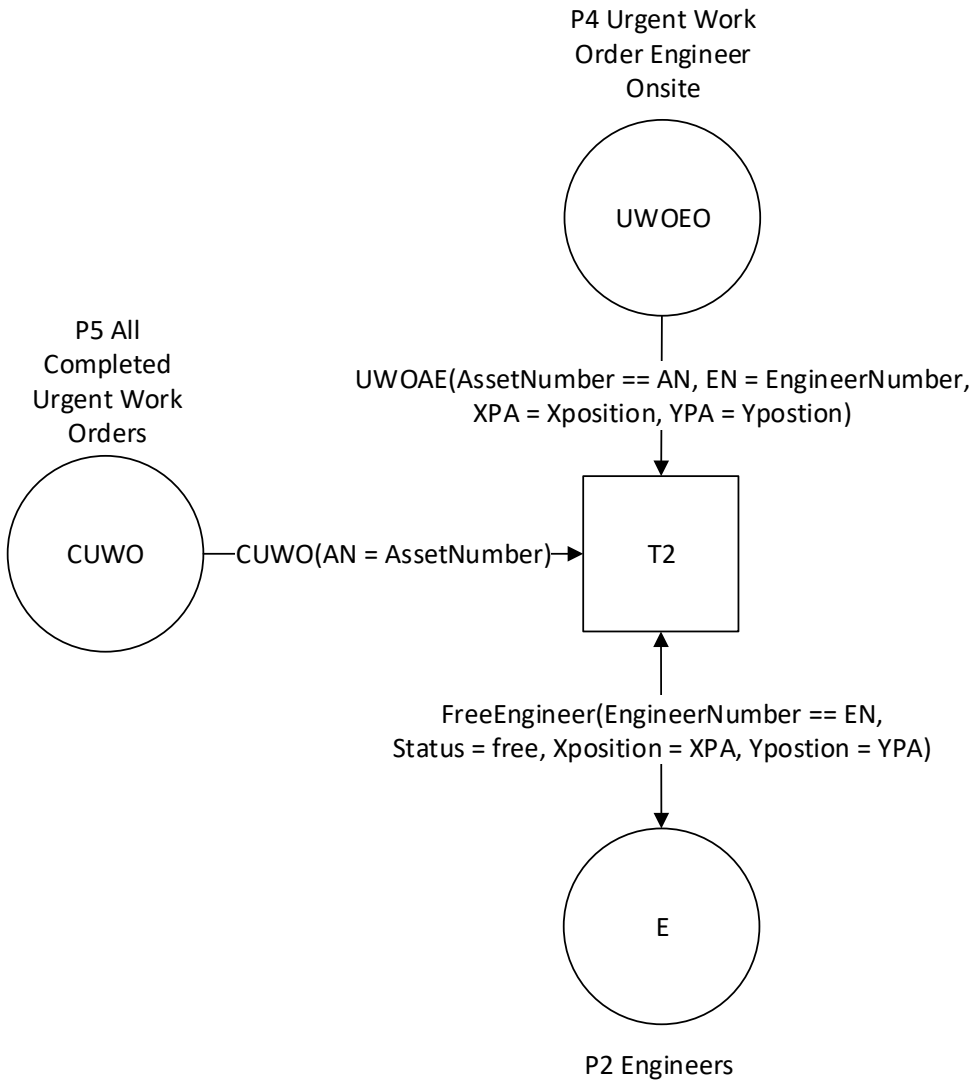

Figure 8-20 Completion of Urgent Work Orders within Framework.

Transition T2 fires immediately. The arc from T2 to P2 selects the engineer token which matches the engineers unique ID held by variable EN. The engineer token's status is then set to 'free', and their position set to that of the asset they have just repaired.

The CPNs for routine work orders and inspections are of an identical structure to that of the urgent work orders, and thus not shown for brevity. However the routine work order CPN transitions have a lower priority, inspections are have a lower priority further. Thus, high priority urgent work orders will be fulfilled before all others.

8.1.2 Renewals

Renewals are modelled as being carried out by contractors, and therefore do not use the same engineers who carry out other work orders. An example CPN for generating renewal work orders is shown in Figure 8-21.

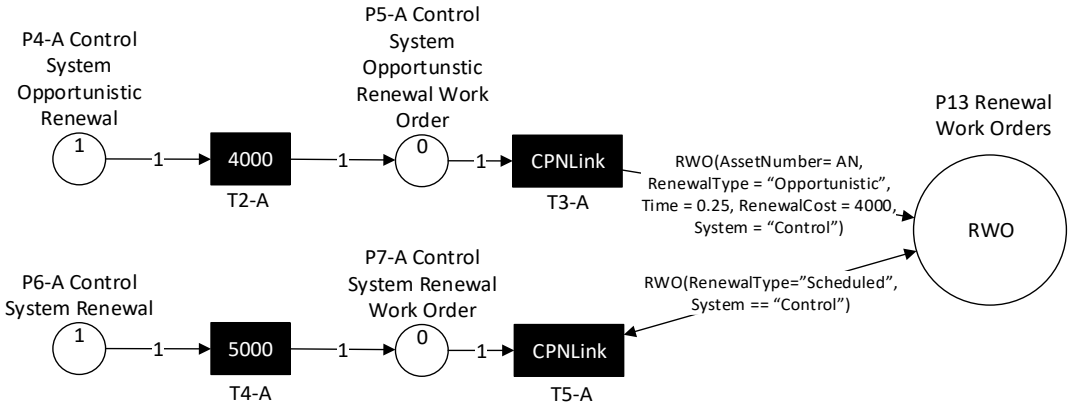

Figure 8-21 Generation of Renewal Work Orders

This Petri net links the asset model renewal work order generation with the resource sharing CPN. The presence of a single token in P4-A enables opportunistic renewals for a specific asset. After a fixed delay transition T2-A removes a token from P4-A and adds a token to P5-A, this then enables a CPN Link transition. This fires immediately, creating a token of colour RenewalWorkOrder with parameters describing the renewal, adding this token to place P13. The colour RenewalWorkOrder is defined in Figure 8-22.

Opportunistic renewals occur if budget allows, described later. Should an opportunistic renewal not occur a scheduled renewal may, if enabled. These occur regardless of budgetary limitations. This is enabled by the presence of a token in place P6-A. This enables transition T4-A, which fires after a delay removing a token from P6-A and adding a token to P7-A. This enables a CPN Link transition, T-5A, which fires immediately. This transition updates the previously generated opportunistic renewal token to a scheduled renewal. If scheduled

renewals are the only type enabled, the Petri net can be modified such that transition T-5A creates a new token.

```
871. Colour RenewalWorkOrder As AssetNumber x RenewalType xRenewalCost x System x Time
872. Colour RenewalCost As Double
873. Colour RenewalType As String
874. Colour System As String
875. Colour Budget As ReplacementPartsCost x AnnualFixedCost x RenewalsCost x AnnualBudget
876. Colour ReplacementPartsCost  As Double
877. Colour AnnualFixedCost  As Double
878. Colour AnnualBudget As Double
```

Figure 8-22 Definition of RenewalWorkOrders and Budget Colours.

Place P14 tracks the budgetary expenditure for the assets within the model, shown Figure 8-23. Transition T13 determines the expenditure of all assets to date. It fires with a delay which reflects the frequency with which expenditure is evaluated. The cost of replacement parts is determined by summing all the number of tokens in the cost tracking places within each asset model. For simplicity, fixed costs are assumed to occur at a constant rate, and are therefore calculated as a function of how much time has passed. However, this could be made much more complex to reflect the dates and times salaries, rent and other expenses are paid. When T13 fires a single token within P14 is updated, though multiple tokens could be used to model separate budgets.

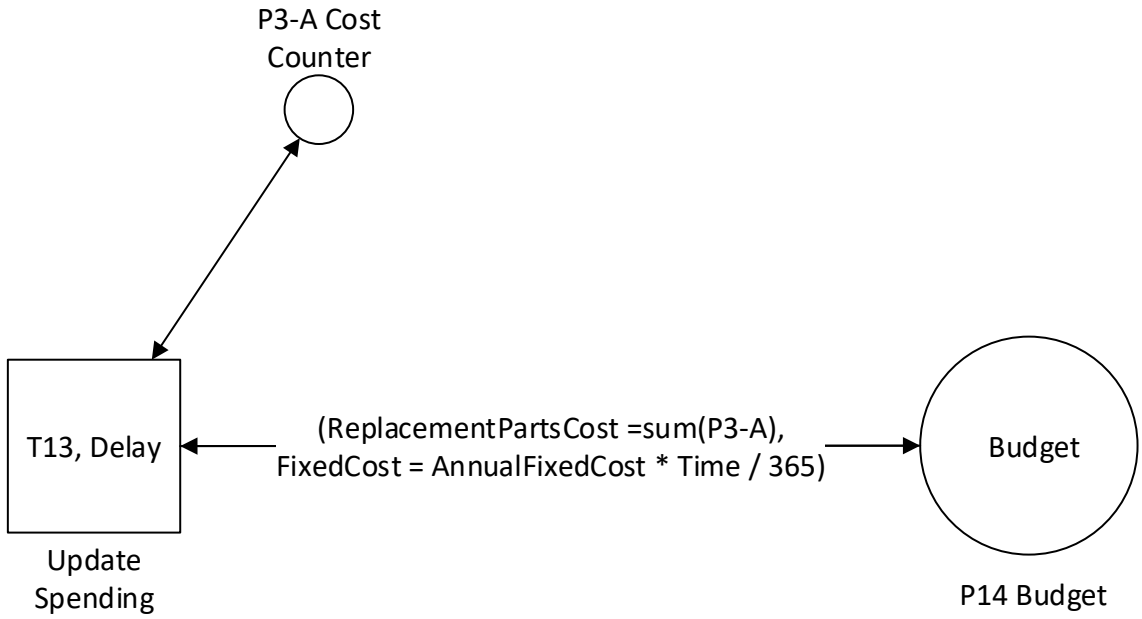

Figure 8-23 CPN for Tracking Budget

The Petri net in Figure 8-24 models the occurrence of opportunistic renewals. A token in place P13 whose RenewalType Colour is set to Opportunistic is required to enable the transition. A guard function evaluates the state of the budget to determine if an

opportunistic renewal can occur, defined in Figure 8-25. This function requires a large number of variables passed to it from an arc from P14 to T14. It uses these variables to calculate the total expenditure to date, line 881. The total expenditure is compared against the budget to determine if the transition shall be enabled, line 883. For simplicity, it is assumed the budget is released daily, however a more complex function could be substituted.

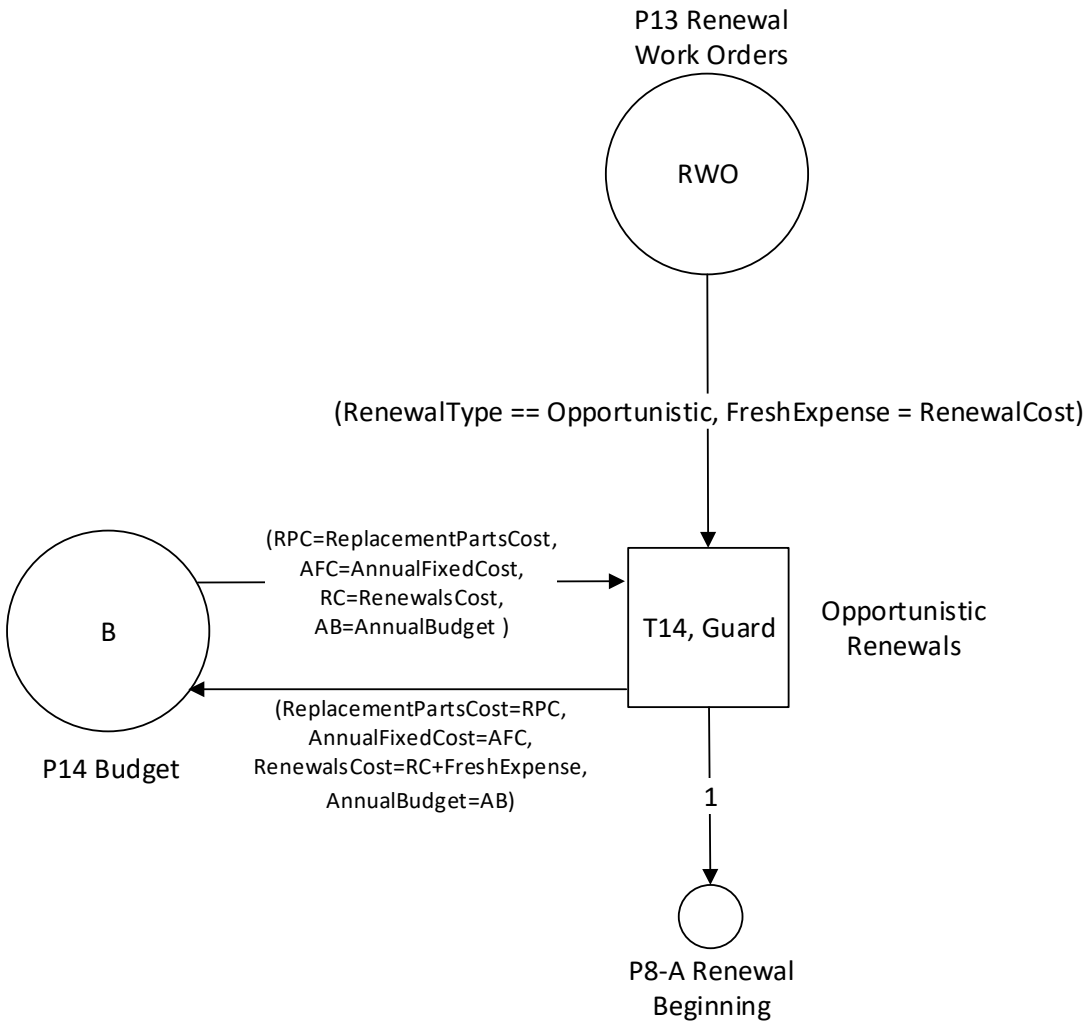

Figure 8-24 Model for Opportunistic Renewals.

If the guard function is satisfied, the transition fires immediately. Upon firing it updates the Budget token in place P14 to reflect the cost of the renewal. It also places a token in the asset model place P8-A allowing the renewal to begin. The renewal token which enabled the transition is then removed from P13.

```
879. T14 Guard (FreshExpense, RPC, AFC, RC, AB, Time) {
880.            //calculate pending total expense amount
881.            double TotalExpense = AFC + RPC + RC + FreshExpense
882.            //if not over budget, enable transition
883.            If (TotalExpense  > AB * IntegerPart(Time/365)) {
884.                        Transition = Enabled
885.            }
886. }
```

Figure 8-25 T14 Guard Function.

The Petri net which models scheduled renewals is similar, but simpler than that for opportunistic, shown Figure 8-26. Transition T15 is enabled by a Renewal token in place P13 with RenewalType Colour set to 'Scheduled', unlike T14, this transition has no guard function as budget does not affect scheduled renewals. T15 fires immediately, updating the Budget token in place P14 and adding a token to the asset model Petri net to begin the renewal.

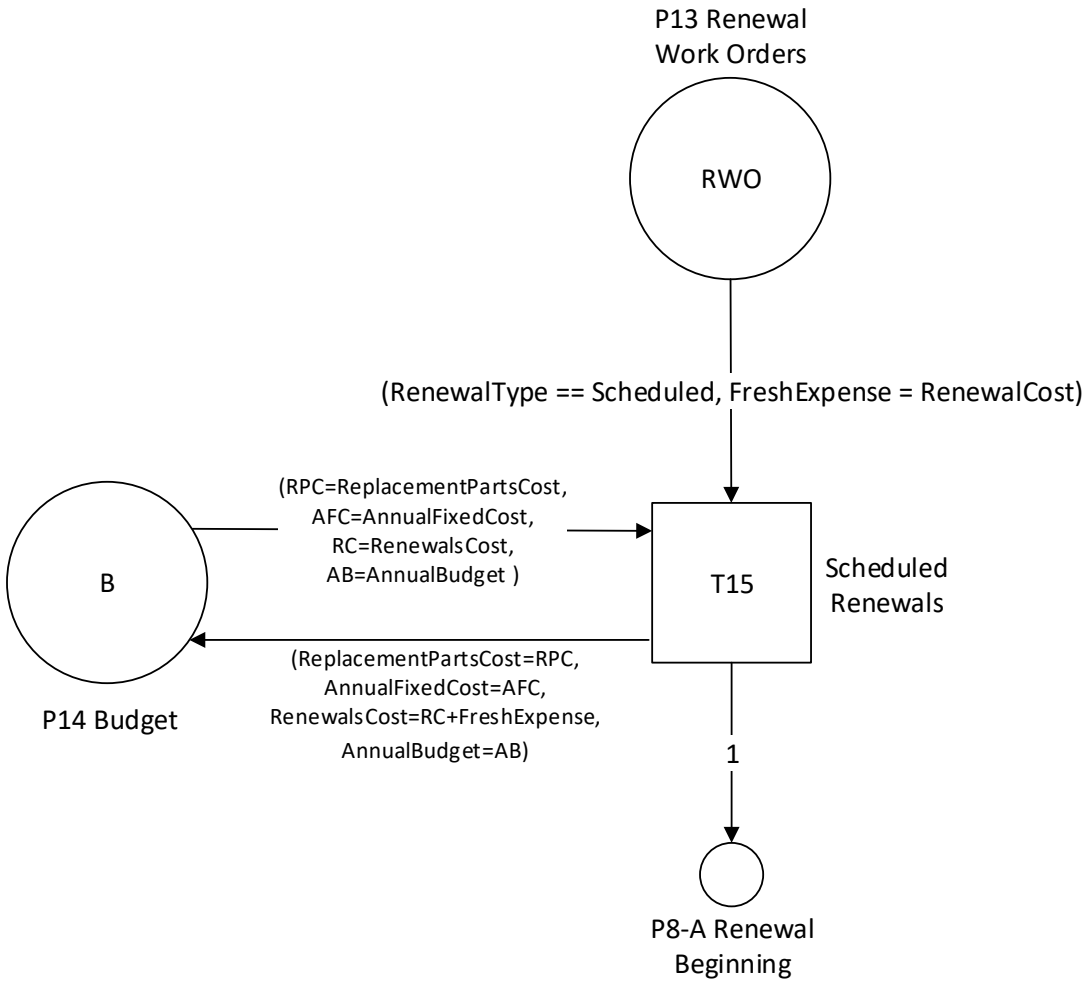

Figure 8-26 Model for Scheduled Renewals.

8.1.3 Staff Management

The CPN modelling elements presented next control the staff working hours. Engineers are modelled as having shift start and end times. They are unable to fulfil work orders outside of

these times. The CPN structure for taking engineers off duty when their shifts end is shown in Figure 8-27. This transition checks the shift start and end times for all available engineers. Should an engineer's shift have ended, they are either taken off duty, or marked as Returning To Base – Off Duty (RTBOD).

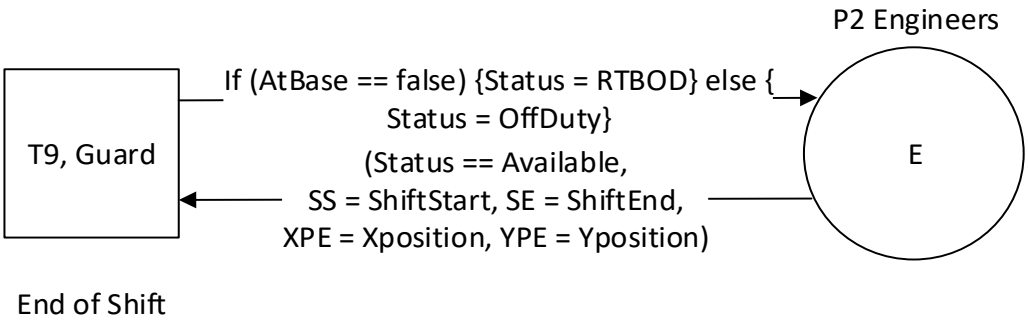

Figure 8-27 CPN Framework End Shift.

The arc from P2 to T9 specifies that only engineers whose status is Available can enable a mode of this transition. It also creates several variables which are passed to the transition's guard function, defined in Figure 8-28. The guard function accepts the shift start time, end time, x and y co-ordinate of the engineer. Line 889 discards the integer component of the time, to get simply the time of day. Line 893 checks if the shift is a normal day shift, where the shift starts and ends on the same day. If true, the following line asks if the current time is between the shift start and end times, setting the variable ShiftValid to true is so.

If an engineer's shift ends the day after it started, as may occur with an evening or night shift the logic is slightly different. Between lines 900 and 905 it is determined if the engineer's night shift is currently active. Line 907 determines if the engineer is currently at the base of operations, setting variable AtBase to true if correct. Lastly, if the engineer is not currently on shift the transition is enabled and the variable AtBase returned so it may be accessed by the arc from the transition to place 2. This arc sets the Status token colour to either RTBOD if the engineer is not at the base, and to OffDuty if the engineer is at the base.

```
887. T9 Guard(SS, SE, XPE, YPE) {
888.         //get time of day by removing integer component of time
889.         double ModTime = Remainder(Time / 1)
890.         Bool AtBase
891.         Bool ShiftValid = False
892.         //if shift starts and ends on same day
893.         If (SE > SS) {
894.                 If (ModTime > SE && ModTime < SS) {
895.                         ShiftValid = true
896.                 }
897.         }
898.         //if shift starts and ends on different days
899.         else {
900.                 If (ModTime > SE && ModTime > SS) {
901.                         ShiftValid = true
902.                 }
903.                 If (ModTime < SE && ModTime < SS) {
904.                         ShiftValid = true
905.                 }
906.         }
907.         If (XPA == 0 && YPA == 0) {
908.                 AtBase = true
909.         }
910.         else { AtBase = false }
911.         //if shift not current, enable transition to allow engineer to go off duty
912.         If (ShiftValid == False) {
913.                 Return Transition = enabled
914.                 Return AtBase
915.         }
916. }
```

Figure 8-28 Transition T9 Guard Function

The CPN in Figure 8-29 models off-duty engineers returning to base. The arc from P2 to T10 specifies that only tokens whose status is RTBOD can enable transition T10, the arc also sets variables for the engineers X and Y co-ordinates and passes them to T10's delay function.

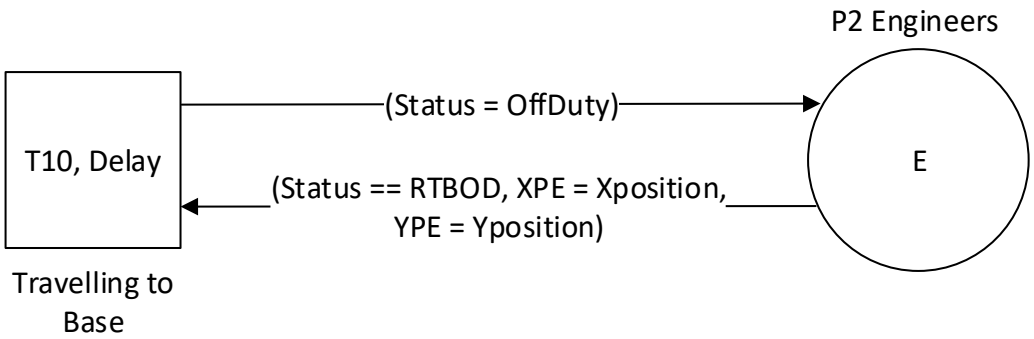

Figure 8-29 CPN Framework Offduty Engineers Returning to Base.

The Delay function for transition T10 is defined in Figure 8-30. It calculates the distance of the engineer to the base using Pythagoras's theorem. It then assumes a 50kph travel speed to calculate the arrival time. The same as engineer travel between assets.

```
917. T10 Delay(XPE, YPE) {
918.        Double EngineerDistance
919.        Double TT
920.        //calculate distance from engineer to base
921.        EngineerDistance = (XPE^2 + (YPE^2))^0.5
922.        //calculate travel time
923.        TT = (EngineerDistance / 50)/24
924.        //return travel time
925.        Delay = TT
926. }
```

Figure 8-30 Transition T10 Delay Function.

The CPN in Figure 8-31 controls the start of an engineer's shift. The arc from P2 to T11 specifies that only engineer token's whose status is set to Offduty can enable a mode of this transition.

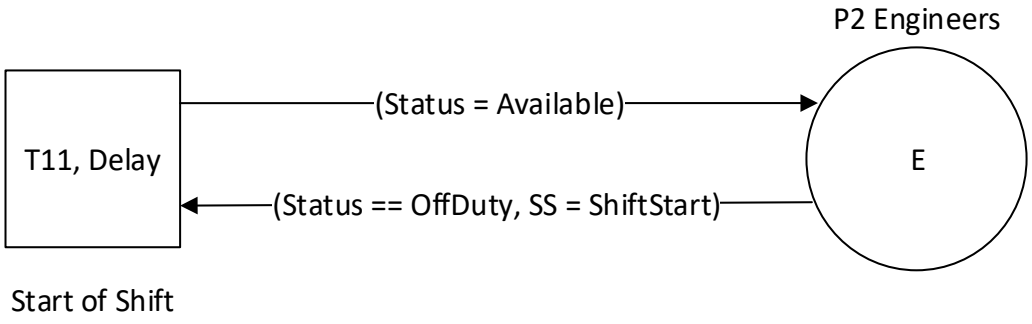

Figure 8-31 CPN Framework, Engineers Starting Shifts.

The transition firing delay within T11 models the time till the engineer's shift starts. This function is defined in Figure 8-32. Line 929 removes the integer part of the current time which represents the date, leaving only the time of day. The following line tests if the shift starts after the current time and calculates the delay accordingly. Should the shift start before the current time, as might occur if a shift starts early in the morning line 935 calculates the delay. When the transition fires the engineer token's status is set to Available.

```
927. T11 Delay(SS) {
928.        //get time of day by remove integer component of time
929.        Double ModTime = Remainder(Time/1)
930.        //if shift starts after current time
931.        If (SS > ModTime) {
932.                //set delay till shift starts
933.                Delay = SS – ModTime }
934.        //other set delay for next day
935.        Else { Delay = 1 + SS – ModTime }
936. }
```

Figure 8-32 Transition T11 Delay function.

The final staff management part of the CPN framework is shown in Figure 8-33. This portion of the net returns engineers to base if there are no further work orders to complete. As such,

the first arc from P2 to T12 specifies that the engineer must be available and not already at base to enable a transition mode.

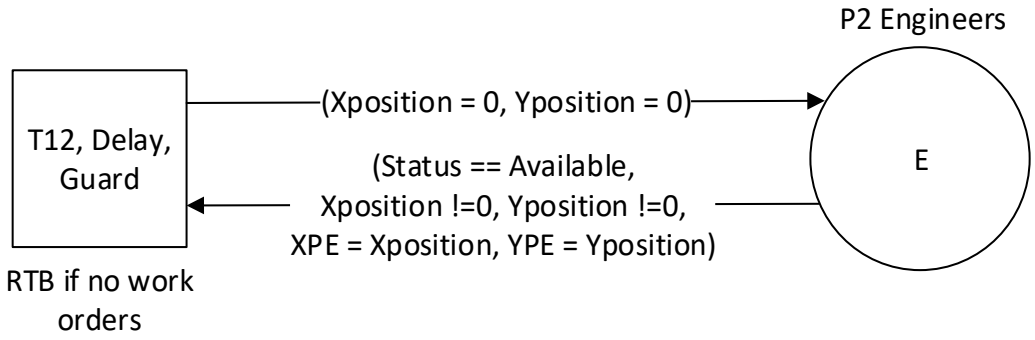

Figure 8-33 CPN Framework Returning Engineers to Base when No Work Orders are Outstanding.

Next, a guard function must be satisfied, defined Figure 8-34, lines 945 to 950. This guard function simply checks that all three places which hold outstanding work orders are empty. Should this condition be met, the transition mode is enabled. Next the transition delay is calculated over lines 937 to 944 in an identical manner to previous transitions modelling engineers returning to base. When the transition fires, the engineer token's position colours are updated reflecting their new location at the base.

```
937. T12 Delay(SS) {
938.            Double EngineerDistance
939.            Double TravelTime
940.            //calculate distance and travel time for engineer
941.            EngineerDistance = (XPE^2 + (YPE^2)^0.5
942.            TravelTime = (EngineerDistance / 50)/24
943.            Delay = TravelTime
944. }
945. T12 Guard(P0, P3, P6) {
946.            //if no work orders, enable transition mode so engineer can return to base
947.            If (P0 == Empty && P3 == Empty && P6 == Empty) {
948.                        Transition = Enabled
949.            }
950. }
```

Figure 8-34 CPN Framework Transition T12 Delay and Guard Functions.

8.3 Work Order Fulfilment Time Analysis

The relationship between the number of engineers on staff at any given time and the time required to fulfil an urgent work order was investigated. The framework was simulated for 2,500 crossing assets, recording the work order fulfilment times. The number of engineers within the model was set to be constant over the day, ie not varying with day or night shifts. In this scenario each crossing begins in new condition.

Table 8-1 shows the results of this test. As the number of engineers is reduced from 12 to 3 there is a linear increase in the average fulfilment time for urgent work orders. However from 2 engineers this breaks down and there is a rapid increase in the average fulfilment time. The same can be seen for routine work orders, but the increase is much more exaggerated

Table 8-1 Average Work Order Fulfilment Times for Varying Numbers of Engineers.

Number of Engineers	Urgent Work Order Fulfilment Time	Routine Work Order Fulfilment Time	Inspection Work Order Fulfilment Time
12	0.026522	0.02891	0.033544
10	0.035984	0.036452	0.049024
9	0.046644	0.042537	0.080913
8	0.05883	0.047033	0.572284
7	0.063401	0.048266	4.31478
6	0.066864	0.049873	11.2438
5	0.07248	0.052976	21.7802
4	0.085379	0.066133	40.2105
3	0.120189	0.133137	79.4069
2	0.586449	278.114	142.301
1	173.749	11281.3	143.76

The relationship between inspection work orders and engineer numbers is slightly different. Inspection work order fulfilment times break down much sooner. However, the increase in times is much less dramatic here. In fact, it appears to show inspection work orders occurring faster than routine work orders, despite the higher scheduling priority of routine work orders. This is due to the frequent nature of inspections, and their lowest work order priority. Inspection work orders are generated at a constant rate of every 28days from the completion of the last inspection. The generation of urgent and routine work orders however, increases with the age of the asset. This results in many inspections occurring whilst the assets are young, but as the assets age the urgent and routine work orders consume greater portions of the engineer time. When an inspection work is not fulfilled for a long time due to this, no further inspection work orders are generated, skewing the average inspection work order fulfilment time towards lower values, when higher values compared to routine work orders would be expected.

Figure 8-35 shows the frequency distribution of fulfilment times for urgent work orders for a model compiled with 2,500 level crossing assets and 7 engineers. It shows a skewed distribution. By inspection it was not found to fit either a lognormal distribution which had been fitted to the sampled fulfilment times, parameters: mu = -2.979; sigma = 0.7831. A fitted Weibull distribution provided the closest approximation to the sampled fulfilment times, parameters: shape = 1.6753; scale = 0.0724. Goodness of fit testing was not required owing to the large numbers of maintenance response times generated by the model. This allows it to be determined by inspection that whilst the Weibull distribution provides a good approximation to the observed distribution, it is not the same distribution.

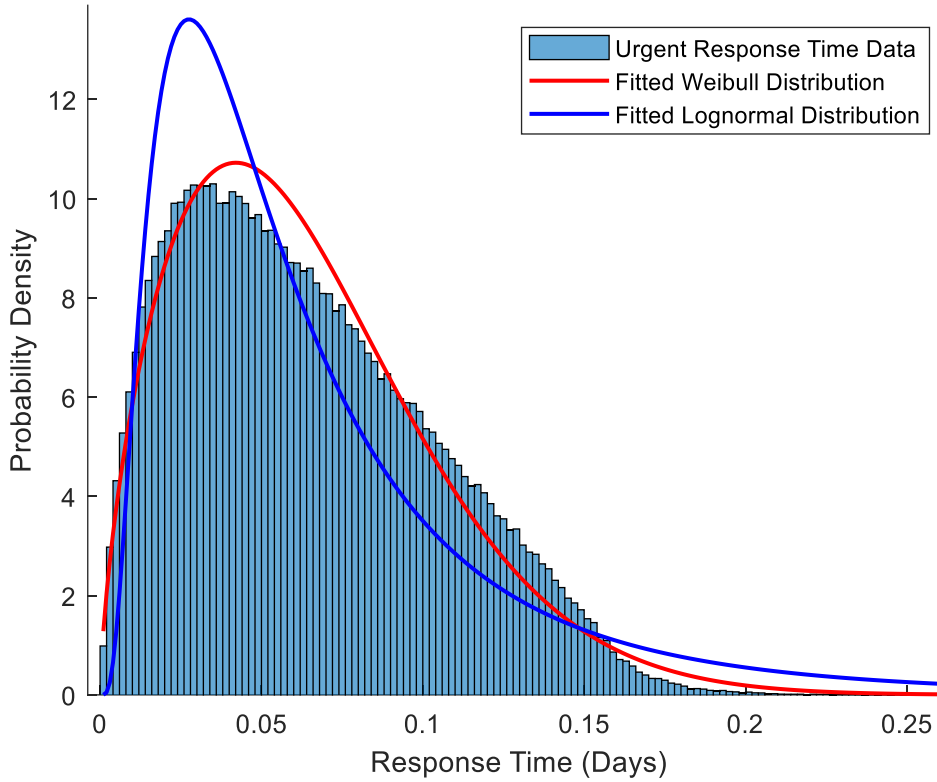

Figure 8-35 Urgent Work Order Response Time Distribution for 7 engineers servicing 2,500 crossing assets.

Reducing the number of engineers produces a distribution resembling lognormal. The test is repeated in Figure 8-36, this time with only 2 engineers. Whilst this appears close to the fitted lognormal distribution, testing did not prove so at the 5% significance level using either the Chi^2 or Kolmogorov-Smirnov test. The Lognormal distribution fitted to the data has parameters: mu = -1.2655; sigma = 1.1822. The Weibull distribution fitted to the data has parameters: shape = 0.7550; scale = 0.5243.

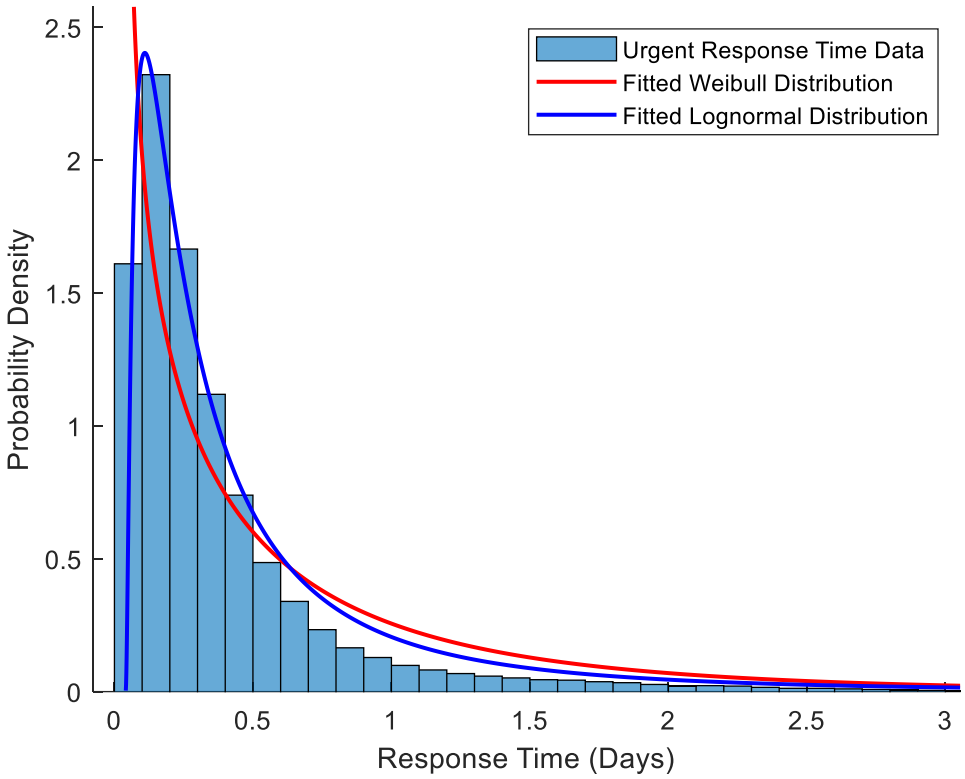

Figure 8-36 Urgent Work Order Fulfilment Time Distribution for 2 engineers servicing 2,500 crossing assets.

The distribution is not always represented by the same shape. When a large number of engineers are simulated relative to the number of assets a distribution emerges with a uniform composite appearance, shown Figure 8-37. This distribution resembles the supposition of a skewed distribution like that seem above and a uniform distribution. The uniform like portion occurs because the assets are assumed to be uniformly distributed along a straight rail line, creating a uniform range of travel times between maintenance depot and base. As there is an abundance of engineers, there is always an engineer at the depot ready to be dispatched to an asset. The initial skewed portion of the distribution stems from how engineers are assigned work orders. Engineers who are closest to an asset are assigned the work order. An engineer who has recently completed a work order will not be at base but will be available, and may have a shorter travel time.

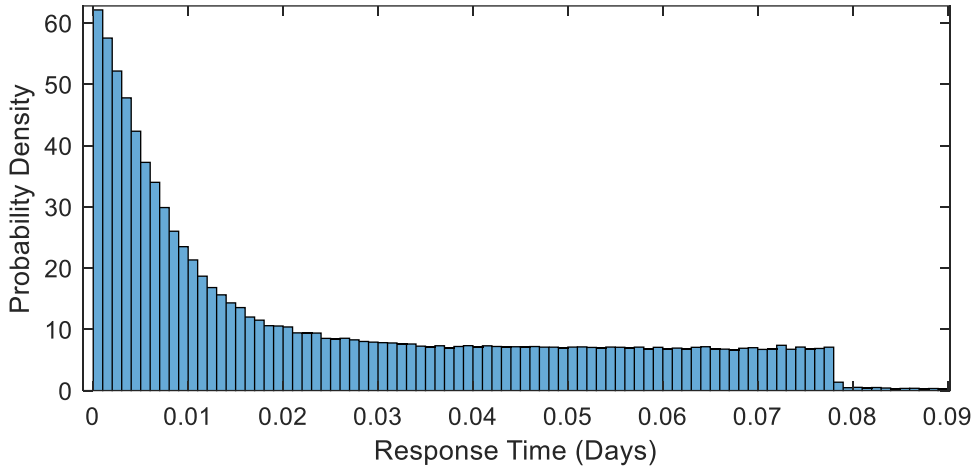

Figure 8-37 Urgent Work Order Fulfilment Time Distribution for 15 engineers servicing 2,500 crossing assets.

The equivalent distributions for routine work order fulfilment times are broadly similar, however inspections are not. When engineers are abundant, the inspection work order fulfilment time appears uniformly distributed. As the number of engineers decreases the fulfilment time becomes skewed to the right. However, at 7 engineers or fewer per 2,500 crossing assets the distribution appears to be a composite, Figure 8-38.

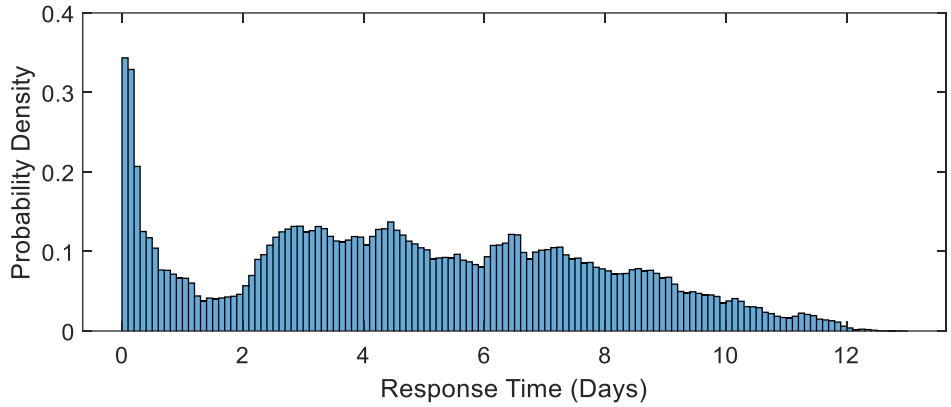

Figure 8-38 Inspection Work Order Fulfilment Time Distribution for 7 engineers servicing 2,500 crossing assets.

This occurs because of the asset life-cycle. In this configuration of the model, all assets begin in new condition. When the assets are young, they have a lower failure rate owing to the use of the Weibull distribution for component failures. This creates a comparatively low volume of urgent and routine work orders during the first few years of the simulation. This gives engineers sufficient time to carry out low priority inspection work orders without delay, producing the initial distribution between 0 and 1 day response time. From 1day response time and above, a second distribution is apparent. This corresponds to when the assets have aged and are exhibiting higher failure rates, occupying more of the engineers time with

urgent and routine work orders leaving them unable to fulfil inspection work orders promptly. This demonstrates that the distribution underlying inspection work order response times is not constant, and can be dependent upon the frequency of generation of other higher priority work orders.

The results of the model presented so far assume that the assets all begin the simulation in new condition. The model was re-configured to distribute asset conditions to explore the effects. This was achieved by bringing a crossing asset online every two days of simulation, for 5,000 days, for a total of 2500 assets. Data from the first 5,000 days was discarded as the availability of engineers would not be representative. First, the effect upon urgent work order response times was examined, comparing the new distribution, shown Figure 8-39, against that previously presented there is little difference.

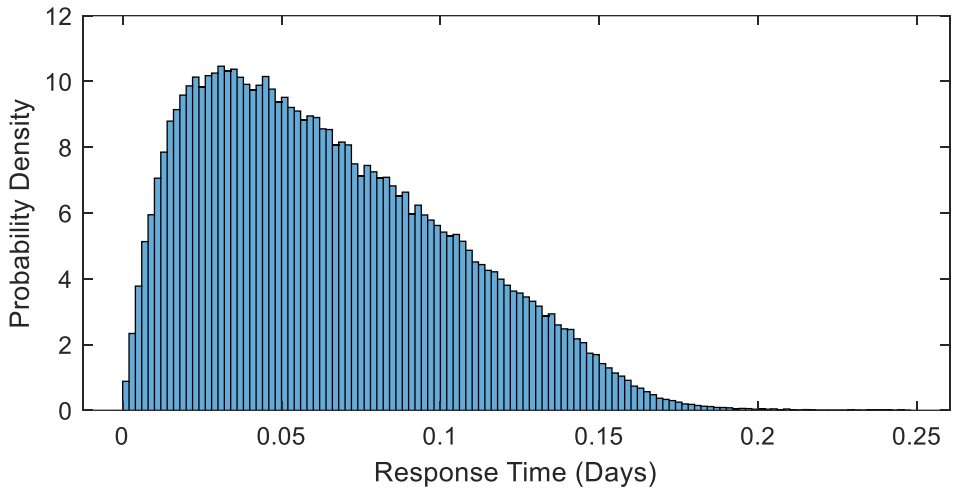

Figure 8-39 Urgent Work Order Response Time Distribution for 7 engineers servicing 2,500 crossing assets whose commission dates have been spread out over 5,000 days.

When the new inspection work order response times are examined, a clear difference can be seen, shown Figure 8-40. When the assets were all the same age, the same graph, Figure 8-38, displayed an initial peak, followed by a long rightward skew. The initial peak is now absent. With the ages of the assets somewhat distributed, there is now no quiet period where few urgent or routine work orders are being generated. The spread of inspection work order response times is also narrower now the asset ages are more distributed. This occurs as there is less variation in the volume of urgent and routine work orders over time, therefore there is more consistent engineer availability for the lower priority inspection work orders.

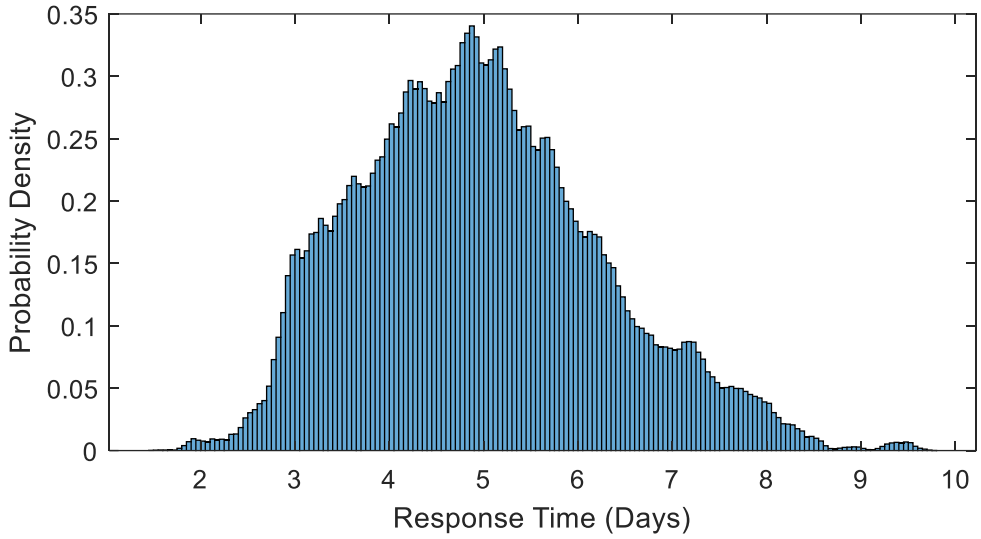

Figure 8-40 Inspection Work Order Fulfilment Time Distribution for 7 engineers servicing 2,500 crossing assets whose commission dates have been spread out over 5,000 days.

The results of this model show that as the availability of the maintenance resources changes so does the distribution of the maintenance response times. Similarly, if the rate of generation of work orders were to change, a change in maintenance response time would also change. The analysis of the model presented here has shown a variety of maintenance response time distributions can be produced as asset age profiles and engineer availability is changed.

8.4 Computational Results

To determine how efficiently the CPN resource sharing model scales with increasing numbers of assets, the model was repeatedly simulated with increasing numbers of assets. The number of available engineers was increased with the number of assets, with 1 engineer available for each 200 assets within the model. The model is configured such that all assets begin in a new state, and their maintenance is simulated for 50years.

As the number of assets within the framework increases there is a linear increase in the memory required by the software to run the simulation, shown Figure 8-41. This memory requirement was obtained from the peak memory usage reported by windows for the software as the model was simulated. This is similar to the memory requirements of SCPN model shown in the previous chapter, highlighting that the CPN scheduling model requires very little memory compared to the SCPN asset models. This is expected, as the CPN resource sharing model has a simple structure and few tokens.

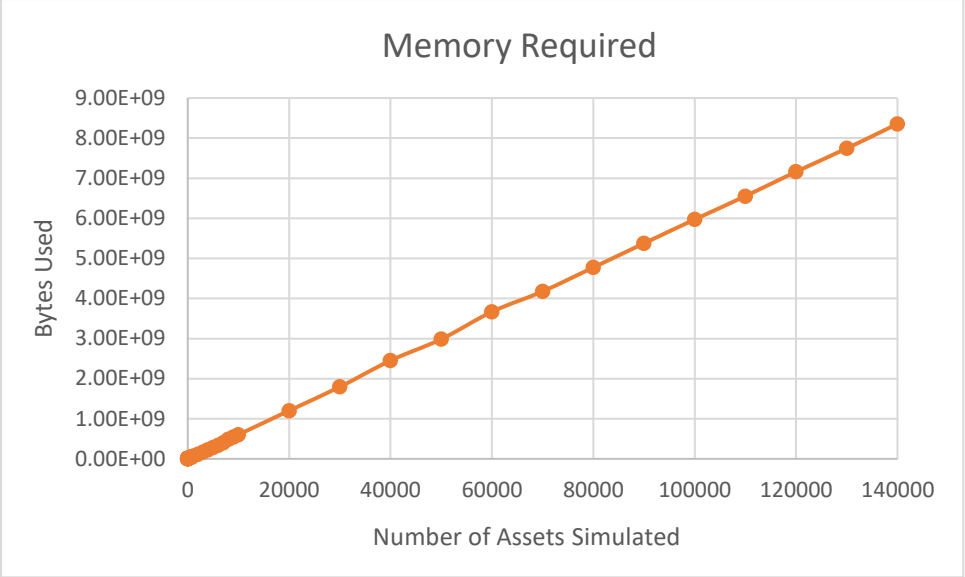

Figure 8-41 Memory Required for Framework Model for Increasing Numbers of Assets.

The computational cost of increasing the number of assets increases in a non-linear fashion, shown Figure 8-42. This is in contrast with the SCPN stochastic scheduling model presented in the previous chapter, which increased linearly. The cause of the non-linear increase is the increasing complexity of conducting an exhaustive search to find the closest engineer to an asset requiring maintenance. As the number of engineers within the model increases, so does the computational cost of finding the closest engineer to any given asset and therefore increases the time to perform each engineer work order assignment. Further work could be to create a heuristic to reduce the computational cost and potentially better represent real life scheduling processes.

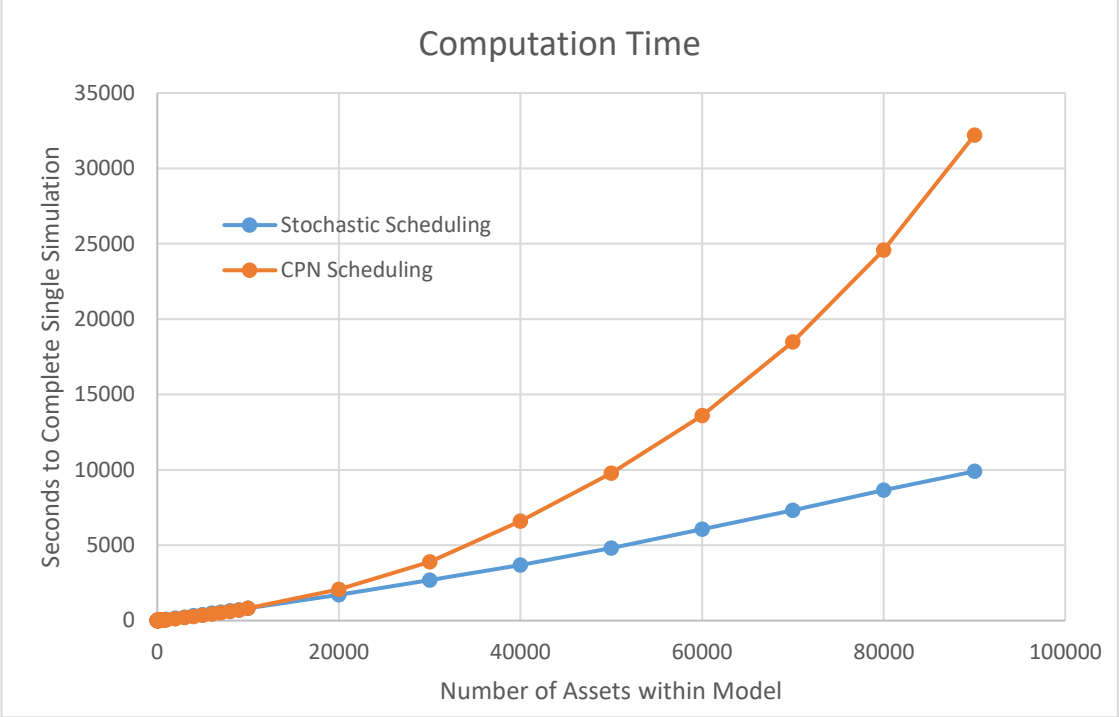

Figure 8-42 Computation Times for Framework Model for Increasing Numbers of Assets

This is a highly idealised framework model, as it features identical assets, and a single maintenance depot. As such, the convergence rate is largely dictated by the number of assets within the model. This is shown in Figure 8-43, which plots the replacement part costs 95% confidence interval against the product of the number of simulations and the number of assets within the model, for a variety of model sizes. It shows a similar rate of convergence for each model. Meaning that convergence rate is dependent on the number assets simulated in total, regardless of how many assets are in each simulation. This is unsurprising, as it would be expected that 5,000 simulations of a single asset model is broadly equivalent to 50 simulations of a 100 asset model in this case.

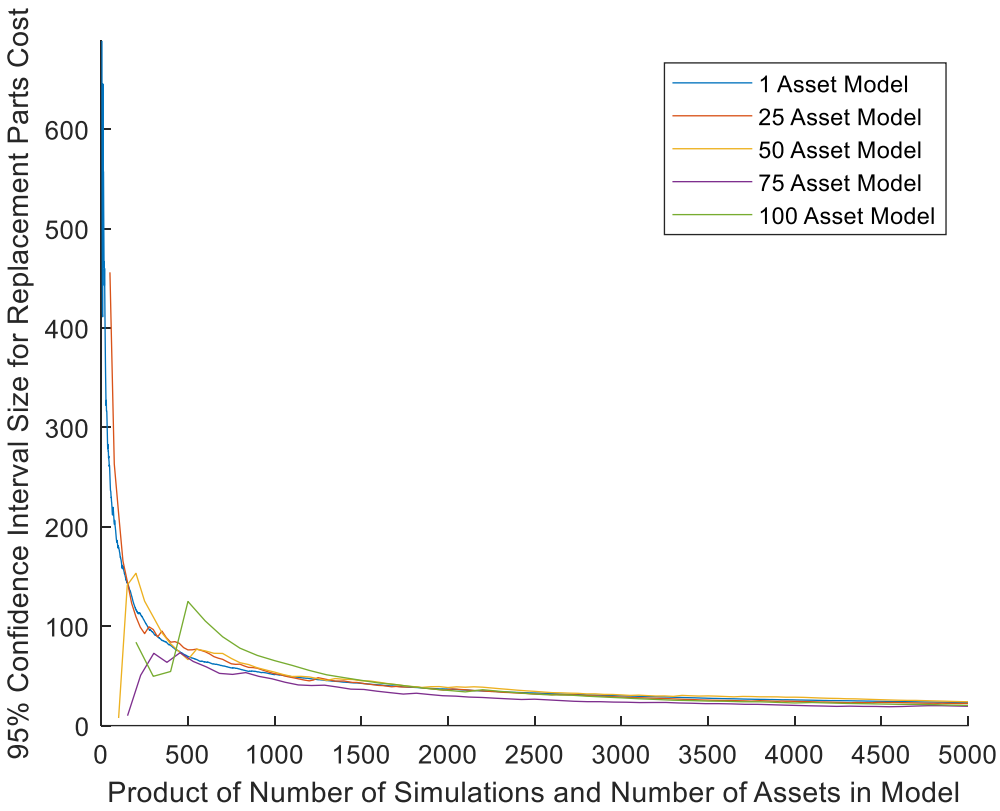

Figure 8-43 Convergence of Monte Carlo Solution of Framework 250 Crossing Assets and 1 Engineer Simulated for 50 Years.

Due to the idealised nature of this model it is not possible to say how many assets a real world model, with diverse assets, could simulate. It is not known how many simulations of a whole asset model are required for convergence. However an upper bound can be considered. Assume that any large whole system model takes the same number of simulations to converge as a single asset model, in this example 1,000 simulations. Also assume contemporary computer hardware capable of running 16 simulations simultaneously and a maximum time budget of 14 days. This equates to a 5.5 hour simulation time per sample, allowing a maximum asset number around 70,000.

8.5 Conclusions

This chapter has presented a CPN model for sharing maintenance engineers between assets, having determined that this is a more efficient methodology compared to TPNs for this purpose. It considers a specified number of engineers, whose number can be varied throughout the day to reflect shift lengths and changes. These engineering staff fulfill work orders based on their urgency, with the engineers closest to the asset requiring work being chosen to fulfil the work order. This served to demonstrate that modelling that a

deterministic scheduling system is possible within the Petri net paradigm, and that its computational costs are acceptable.

Increasing numbers of assets yielded a non-linear increase in computation cost. However this still enabled models containing large numbers of assets to be simulated. A linear increase in memory requirement was observed as the number of assets within the model increased, governed mostly by the size of the SCPN asset models. This enables the simulation of models with up to 70,000 assets, with even larger asset models possible if they require less than 1,000 simulations to converge. Further increases in assets modelled could be found by increasing the computing power available.

Many opportunities for further work are available for this resource sharing model. Chiefly, it should be expanded to better reflect work order scheduling practises. As at present the model uses a responsive exhaustive search to allocate engineers to the closest, highest priority work order. However, a heuristic approach may create a more accurate model and improve computational efficiency. At present the model focuses on simple work orders which can be carried out promptly. However an additional scheduling system to account for mobilisation time of complex tasks which require greater planning to carry out could also improve accuracy.

9 Conclusions

This chapter summarises the work conducted in this thesis. It appraises how this thesis meets the objectives outlined in the introductory chapter. It also outlines the contributions of this work to the field, and lastly discusses further work which has been enabled by the work presented in this thesis.

9.1 Thesis Objectives

Objective One: The first objective of this thesis was to review the state of asset management modelling for each major railway asset group, evaluating the state of the art of modelling for these assets. This furthers the adoption of frameworks by highlighting which assets have already achieved a suitable level of development, and which assets require development in order to form a functional library of asset models. This was achieved by way of a literature review. It was found that there was a large number of studies conducted on modelling bridge and rail track assets, producing a wide variety of asset management models using varying methodologies. It is evident that the core aspects of modelling these assets are sufficient.

A single asset management model was found for Overhead Line Equipment (OLE). This model had a comparable level of development to those presented for bridges and rail track. However, as the sole model published for this purpose there is little discourse or contrasting approaches with which to evaluate it. It is therefore unclear if OLE would benefit from additional modelling development. Similarly to OLE, only two asset management models were found for signalling. These models were comparatively simple when viewed against the bridge, track and OLE model. It is therefore quite likely that signalling asset management modelling will benefit from further development.

Earthworks were poorly represented in asset management modelling. Several basic Markov models were found for modelling degradation. Whilst further development would be beneficial, it is unlikely to be useful until the integrity of earthworks can be better assessed. Many models supporting asset management for level crossing assets were found, focusing on predicting fatal accidents. However, no asset management models were found.

No asset management models for Level Crossings were found, some risk models were found, however these were often focused on a specific aspect of risk at level crossings, such as blocking back. Owing to their mechanical complexity and adverse risk profile, Level Crossings were chosen to be the focus of objective two.

Objective Two: This objective was to develop a novel asset management model. Automatic Half Barrier (AHB) type level crossings were chosen specifically due to their technical complexity and high accident rate. The models developed for the Automatic Half Barrier level crossing comprised three separate modelling efforts. The first, a model of the mechanical and electrical operation of an automatic half barrier level crossing was developed to elucidate the effects of specific component failures and combinations of failures upon the operational state of the crossing. This was achieved using a timed Petri net with high level extensions to simulate the operation of the electronic and mechanical elements depicted in technical schematics for the crossing type.

This supported and verified the development of a fault tree for the most serious system failure modes where the protection systems failed to operate despite an approaching train. The results of the simulation model itself have not been verified. This would require using actual level crossing equipment in a laboratory setting.

The second model developed was a Coloured Petri Net (CPN) for predicting the risk of collisions between trains and road vehicles at AHB crossings. The model predicted increases in collision risk whenever either or both of the protection systems failed to function. It was also able to reproduce the non-linear relationship between collisions and traffic levels observed in previous studies.

The last model developed solely for the AHB crossing was the asset management model. This model predicts the volume of maintenance required, hazardous failure events and maintenance expenditure for an AHB crossing. The model uses the stochastic Petri net modelling methodology, and was solved using a Monte Carlo method.

The asset management model includes limitations common to the other modelling effects presented. The failure rates used in this model could be been obtained through data analysis but were assumed. The accuracy of the models predictions were not tested against real world data. It is a general limitation of the models presented in this thesis that their accuracy has not been verified through data analysis. This was a deliberate choice to enable a broader set of objectives including both asset model development and to further the field of asset management modelling frameworks.

Objective Three: The most important objective of this work, considered throughout this thesis, has been to identify the most suitable modelling methodology for asset management frameworks. Literature review found two Petri net methodologies had been successfully

applied to modelling large whole asset models, stochastic Petri nets and coloured Petri nets. The models found in literature offered no clear advantage to either methodology for either single asset management modelling or whole asset, whole lifecycle use. To explore the computational requirements of both methodologies, two logically identical models were developed and tested.

It was found that stochastic Petri nets can be simulated rapidly, but require large amounts of computer memory. Coloured Petri nets produced the opposite result, being slow to compute but requiring little memory. To find a balance between these, a third type of Petri net was developed, referred to as the Simple Coloured Petri net. Computational testing demonstrated this Petri net methodology would allow the largest amount of assets to be simulated in a single model.

The results of this computational testing have a significant limitation as the results obtained are dependent upon the software used to simulate the Petri nets. The software was authored specifically for this work, using the same programming language, compiler and simulated using the same computer hardware to make the comparisons as valid as possible. But it does not rule out that with additional software development the results might be different.

The last aspect of asset management frameworks considered in this thesis was how to model resources shared between each assets. It was found during literature review that efforts have been made to model resource allocation between multiple assets, but none were sufficient as they used a non-standard modelling methodology or simply were not documented. It was therefore sought to find a computationally efficient and otherwise effective Petri net methodology to allow resource allocation between multiple assets. Due to its limited feature set, Stochastic Petri nets were found to be incapable of modelling complex resource sharing processes. Therefore, despite its computational drawbacks, the only Petri net suitable was the Coloured Petri net due to its significantly enhanced feature set.

A model for resource sharing was developed using this methodology. A Coloured Petri net was used to allocate engineers to fulfil maintenance to work orders for assets modelled using the Simple Coloured Petri net Methodology. There was a modest increase in computational time compared to non-resource sharing configurations, though still enabled a large number of assets to be simulated with resource sharing.

9.2 Key Contributions

Below, the key contributions this work has made to the field of railway asset management modelling are briefly summarised.

- This work reviewed the current state of the art for asset management modelling of railway assets. This has highlighted assets which require additional modelling work, such as signalling. It has also highlighted earthworks as an asset which may require developments in geotechnical engineering before asset management modelling is feasible or useful.
- In some legacy railway applications such as level crossings, relay logic based control systems persist. The development of Petri nets with Boolean logic gate functionality allows the operation of these systems to be simulated. This can be used to automate testing of both normal operation, and the effects of failure upon the system.
- A risk model for predicting the volume of collisions at automatic level crossings has been developed. Road vehicles and trains are modelled using an agent based approach, allowing simple re-configuration to reflect different trains, road vehicle types and behaviours towards crossing protection systems.
- A stochastic Petri net model for assessing different asset management strategies for Automatic Half Barrier crossings was developed. The model takes inputs such as inspection and renewal frequencies and outputs predictions for key performance indicators such volumes of maintenance required and volumes of hazardous failure events.
- Prior to this work, whole system models for asset management of railway assets have been published which use Petri nets. However, there has been little discussion of the computational costs of the different types of Petri net. This work has presented the computational costs in terms of memory and computation time for three logically identical Petri net methodologies. Finding that Stochastic Petri nets benefit from a faster computation time but higher memory requirements and the opposite findings for Coloured Petri nets.
- A novel Petri net type designed to maximise simultaneous modelling of assets was developed, referred to as the Simple Colour Petri net. This is structurally identical to a Stochastic Petri net, however additional identical assets can be added to the model for a considerably lower overall memory increase compared to a Stochastic Petri net. This enables simulating a far larger number of assets within a single Petri net model using the Simple Colour Petri net compared to the Stochastic Petri net.

- This thesis presented two methods of optimising the simulation of Stochastic Petri nets. The first proposed analysing the structure of the Petri net prior to simulation to create a table mapping places whose number of tokens have changed to transition potentially affected. This reduces the number of transitions whose state must be checked each iteration of the simulation loop. The second grouped transitions together as blocks, tracking the transition with the shortest firing delay within the block. This reduces the number of transition delays which must be checked when finding the next transition to fire chronologically. These optimisations greatly reduced simulation times, particularly for large models. They will also benefit smaller models undergoing computationally intensive optimisation procedures.
- A computationally efficient methodology for modelling large whole asset, whole lifecycle models was developed. This comprised a Simple Colour Petri net to simulate individual assets, with a Coloured Petri net modelling the sharing of resources between them. This allowed up to 70,000 assets to be simulated using contemporary desktop computer hardware. Prior to this work, other authors had considered the effect of using different types of computer hardware to increase the number of assets which may be simulated in a single model, but not the modelling methodology itself.

9.3 Further Work

The work within this thesis furthered asset management modelling capability and knowledge. But in doing so has created a new frontier. The work which could be done to push this frontier further is briefly outlined and discussed below.

This thesis has presented three models for Automatic Half Barrier crossings, however with the focus of this work being to advance asset management modelling frameworks, much work remains to be completed within these models. The results of the AHB crossing system simulation and fault tree it helped develop might be tested using the hardware from an actual AHB crossing in a laboratory to determine if the results it produces are accurate. The model itself considers only the logic of the relay models, there may be electrical effects such as relay chattering which could also be explored.

The AHB crossing collision risk model was simulated using assumed parameters for driver behaviour. These parameters could be determined from field study of drivers at AHB crossings. These parameters may also be obtained from CCTV from level crossings, or driving simulator studies. Though, the method of data collection employed may skew the results.

Further, this could present an opportunity to validate the model. Having calibrated the model using data collected from a crossing or selection of crossings, the model could then be used to estimate the volume of hazardous behaviours over a further period of time at the same crossings.

Much of the development time devoted to the AHB crossing asset management model was dedicated to evaluating the computational requirements of different Petri net methodologies. This left several aspects of the model in need of further development. The model used assumed failure distributions and parameters for the components of the crossing. Analysis of failure records of actual AHB crossings can be performed to obtain find more appropriate failure distributions and parameters. Following this, the model may then be evaluated as to how accurately it can predict key performance indicators by comparing its results with historical data.

This thesis furthered the use of frameworks for railway asset management modelling, highlighting new avenues for research. It was shown that there has been insufficient development of asset management modelling of signalling assets. This would be fruitful area for research, as signalling assets are in use in large numbers on all railway networks around the world. Further, as they are fundamentally mechanical and electrical components, they are likely to exhibit well defined and predicable failure modes, simplifying model development. With greater computer control over signalling assets on the horizon with the development of ERTMS there is also scope for the novel inclusion of software failures and maintenance into the asset management modelling process.

Asset management modelling of earthworks was also found to be lacking. This was speculated to be more due to the difficulty of assessing the condition and safety earthworks, rather than a simple lack of development. Work none the less could be done to verify this, and to explore how emerging technologies could be used to elucidate the condition and state of earthworks, and how this might be integrated into an asset management model.

There are other assets whose state of the art for modelling can be considered mature, requiring little further development. Regardless of this maturity, they cannot immediately be integrated into an asset management modelling framework library. The models in literature will require modification to make them output results compatible with each other. They will require some standardisation of the Petri net structures used to further their compatibility when simulated as one cohesive model, particularly where inter-asset interactions are to be modelled.

With a library of asset modelled completed, attention should be turned to the form and function of the software need to compile, simulate and interpret the results of the models created. This software needs to take inputs such as asset types, conditions, and locations, and use them to construct a Petri net from the asset models within the library. The complexity of this software will depend in part on the diversity of the asset models within the library, for example if multiple types of Petri net methodologies are included.

The resource sharing model presented in the penultimate chapter of this thesis was intended as a basic example of what could be achieved, and to demonstrate its feasibility. It is therefore rich with aspects which could be developed further. It uses a very basic algorithm to assign resources which performs an exhaustive search. This is unlikely to reflect real world behaviour, where a heuristic approach might be of greater accuracy. There has also be little exploration of how to improve the efficiency of Colour Petri net simulation. Work in this area could focus on either the Colour Petri nets themselves, or the architecture of the software which simulates them. The consideration of service provision focuses mostly on recording the number of events which may delay a train. Integration of a model for movement and routing of trains along a network would allow the creation of much more detailed predictions of service provision including aspects such as lengths of service delays.

Lastly, it would be useful to determine what, if any, significant effects different resource sharing models have on the accuracy of the predications produced by that model. This work could be performed by reproducing some of the whole asset models in literature with a variety of different resource sharing approaches and comparing their predictions both amongst the models and real world data for the assets they model.

10 References

Abubaker, A., Baharum, A. & Alrefaei, M. H., 2015. Automatic Clustering Using Multi-objective Particle Swarm and Simulated Annealing.. *PLOS ONE,* 10(7).

Agrawal, A. K., Kawaguchi, A. & Chen, Z., 2010. Deterioration rates of typical bridge elements in New York. *Journal of Bridge Engineering,* 15(4), pp. 419-429.

ALCAD, 2020. *Vantex Range, Maintenance-free Ni-Cd batteries,* Harlow: ALCAD.

Alva-Hurtado, J. & Selig, E., 1981. Permanent strain behavior of railroad ballast. *Proceedings of the 10th International Conference on Soil Mechanics and Foundation Engineering,* pp. 543-546.

Andrews, J., 2012. A Modelling Appraoch to Railway Track Asset Management. *Journal of Rail and Rapid Transit,* 227(1), pp. 56-73.

Argent, R. et al., 2006. Comparing modelling frameworks – A workshop approach. *Environmental Modelling and Software,* 21(7), pp. 895-910.

Aruoba, S. B. & Fernandez-Villaverde, J., 2015. A comparison of programming languages in macroeconomics. *Journal of Economic Dynamics and Control,* Volume 58, pp. 265-273.

Audley, M., 2014. *Rail Track Geometry Degradation and Maintenance Decision Making, PhD Thesis.* Nottingham: University of Nottingham.

Audley, M., 2014. *Rail Track Geometry Degradation And Maintenance Decision Making, Phd Thesis.* Nottingham: University of Nottingham.

Audley, M. & Andrews, J., 2013. The effects of tamping on railway track geometry degradation. *Journal of Rail and Rapid Transit,* 227(4), pp. 376-391.

Bai, L. et al., 2015. Markov-based model for the prediction of railway track irregularities. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit,* 229(2), pp. 150-159.

British Rail, 1991. *Standard Lifting Barrier MKII for Level Crossings,* Milton Keynes: Internal Accesss Only.

British Rail, 1993. *Track Circuits - Treadle Assisted Track (Technical Drawings),* Milton Keynes: Network Rail - Internal Access Only.

British Railways Board, 1981. *Specification for Miniature Tractive Armature DC Neutral Line Relay for Railway Signalling Purposes,* Crewe: British Railways Board - Internal Access Only.

British Railways Board, 1992. *Introduction to Level Crossing Circuitry,* London: British Railways Board - Internal Access Only.

BSI Group, 2012. *BS EN 62551 Analysis techniques for dependability - Petri net modelling,* London: BSI.

BSI, 2019. *Systems and software engineering. High-level Petri nets. Concepts, definitions and graphical notation,* London: BSI.

Calvert, G., Neves, L., Andrews, J. & Hamer, M., 2020. Modelling interactions between multiple bridge deterioration mechanisms. *Engineering Structures,* Volume 221.

Cesare, M., Santamarina, C., Turkstra, C. & Vanmarcke, E., 1992. Modelling Bridge Deterioration with Markov Chains. *Journal of Transportation Engineering,* Volume 118, pp. 820-833.

Chan, F. T. S., Prakash, A., Ma, H. L. & Wong, C., 2013. A hybrid Tabu sample-sort simulated annealing approach for solving distributed scheduling problem. *International Journal of Production Research,* 51(9), pp. 2602-2619.

Chang, M.-H., Das, D., Varde, P. & Pecht, M., 2012. Light emitting diodes reliability review. *Microelectronics Reliability,* 52(5), pp. 762-782.

Cheng, Y.-H. & Yan, L.-A., 2009. A Fuzzy Petri Nets approach for railway traffic control in case of abnormality: Evidence from Taiwan railway system. *Expert Systems with Applications,* Volume 36, pp. 8040-8048.

Chiachío, J., Chiachío, M., Prescott, D. & Andrews, J., 2018. Modelling adaptive systems using plausible Petri nets. *8th International Workshop on Reliable Engineering Computing.*

Chian, W., Liu, K. F. R. & Lee, J., 2000. BRIDGE DAMAGE ASSESSMENT THROUGH FUZZY PETRI NET BASED EXPERT SYSTEM. *Journal of Computing in Civil Engineering,* 14(2), pp. 142-149.

Cinlar, E., 1975. *Introduction to Stochastic Processes.* Dover Edition ed. New Jersey: Prentice-Hall.

Cullen, W., 2001. *The Ladbroke Grove Rail Inquiry.* Norwich: HSE Books.

Davis Associates Limited, 2005. *Level Crossings: Summary of Findings and Key Human Factors Issues,* Norwich: HSE Books.

Deb, K., 2001. *Multi-Objective Optimization using Evolutionary Algorithms.* s.l.:Wiley and Sons.

Denysuik, R. et al., 2016. Multiobjective Optimization of Maintenance Scheduling: Application to Slopes and Retaining Walls. *Advances in Transportion Geotechnics 3, The 3rd Internation Conference on Transportion Geotechnics,* Volume 143, pp. 666-673.

Department For Transport, 2018. *Vehicle Speed Compliance Statistics, Great Britain: 2017,* London: Department For Transport.

Department of Transport, 1983. *Report on the Collapse of Penmanshiel Tunnel that occured on 17th March 1979,* London: Department of Transport.

Department of Transport, 1990. *Report on the Collaspse of Glanrhyd Bridge on 19th October 1987,* London: Department of Transport.

Dikanski, H., Imam, B. & Hagen-Zanker, A., 2018. Effects of uncertain asset stock data on the assessment of climate change risks: A case study of bridge scour in the UK. *Structural Safety,* 71(1), pp. 1-12.

D'souza, S., 2017. *Asset Management of Offshore Oil and Gas Installations. Phd Thesis.* Nottingham: University of Nottingham.

Estevan, A. M., 2015. *Dependability and Safety Evaluation of Railway Signalling Systems Based on Field Data.* PhD Thesis ed. Lulea: Lulea University of Technology.

Evans, A., 2011. Fatal accidents at railway level crossings in Great Britain 1946–2009. *Accident Analysis and Prevention.*

Fecarotti, C. et al., 2018. *Deliverable D6.2 Requirements and initial concept for an asset management framework,* Milton Keynes: Innovative Intelligent Rail.

Fennell, D., 1988. *Investigation into the King's Cross Underground Fire,* London: Department of Transport.

Fenton, N. & Neil, M., 2013. *Risk Assessment and Decision Analysis with Bayesian Networks.* CRC Press: s.n.

Fourment, M. & Gillings, M., 2008. A comparison of common programming languages used in bioinformatics. *BMC Bioinformatics,* Volume 9:82.

Frangopol, D. M., Kong, J. S. & Charaibeh, E. S., 2001. Reliability-based life-cycle management of highway bridges. *Journal of Computing in Civil Engineering,* 15(1), pp. 27-34.

Gallay, R., 2015. Metallized Film Capacitor Lifetime Evaluation and Failure Mode Analysis. *Proceedings of the CAS-CERN Accelerator School: Power Converters, Baden, Switzerland,* pp. 45-56.

Gao, C., Fu, C., Huang, J. & Hu, S., 2014. Failure analysis for electromagnetic relay contacts adhesion by using XES. *2014 Prognostics and System Health Management Conference (PHM-2014 Hunan).*

Ghazel, M., 2009. Using Stochastic Petri Nets for Level-Crossing Collision Risk Assessment. *IEEE Transactions on Intelligent Transportation Systems,* , 10(4), pp. 668-677.

Ghosn, M. & Frangopol, D. M., 1998. Structural reliability in bridge engineering. *Journal of Bridge Engineering,* 3(4), pp. 151-154.

Glover, F., 1990. Tabu Search: A Tutorial. *Interfaces,* 20(4), pp. 74-94.

GrippenKoven, J. & Dietsch, S., 2016. Gaze direction and driving behavior of drivers at level crossings. *Journal of Transportation Safety & Security,* 8(1), pp. 4-18.

Gunn, D. et al., 2018. Deterioration model and condition monitoring of aged railway embankment using non-invasive geophysics. *Construction and Building Materials,* Volume 170, pp. 668-678.

Han, L., Xing, K., Chen, X. & Xiong, F., 2015. A Petri net-based particle swarm optimization approach for scheduling deadlock-prone flexible manufacturing systems. *journal of intelligent manufacturing.*

Health and Safety Executive, 2002. *The track obstruction by a road vehicle and subsequent train collisions at Great Heck 28 February 2001,* Norwich: HSE Books.

Henry Williams Limited, 2020. *Electromechanical Detectors, Cautor and Forfex,* Darlington: Henry Williams.

Hewitt, D. & Green, J., 2016. Observation of electrolytic capacitor ageing behaviour for the purpose of prognostics. *42nd Annual Conference of the IEEE Industrial Electronics Society.*

Jain, V., Swarnkar, R. & Tiwari, M. K., 2003. Modelling and analysis of wafer fabrication scheduling via generalized stochastic Petri net and simulated annealing. *International Journal of Production Research,* 41(15), pp. 3501-3527.

Jensen, H. & Kristensen, L. M., 2009. *Coloured Petri Nets.* 1st ed. Berlin: Springer-Verlag Berlin Heidelberg.

Jensen, K., 1997. *Coloured Petri Nets: Basic Concepts.* Berlin: Springer.

Jiang, Y. & Sinha, K., 1989. Bridge Service Life Prediction Model Using the Markov Chain. *Transportation Research Record,* Volume 1223, pp. 24-30.

Kagra, H., 2013. Design of Experiment for factors affecting Contact Resistance in Metal to Carbon Relays. *International Journal of Science, Engineering and Technology Research,* pp. Volume 2, Issue 1, January 2013.

Kilsby, P., Remenyte-Prescott, R. & Andrews, J., 2017. A modelling approach for railway overhead line equipment asset management. *Reliability Engineering & System Safety,* Volume 168, pp. 326-337.

Kilsby, P., Remenyte-Prescott, R. & Andrews, J., 2018. A Petri Net based life cycle cost analysis approach. *ImechE Part F: Rail and Rapid Transit.*

Knoop, V. L., 2017. *Introduction to Traffic Flow Theory: An introduction with exercises.* 1st ed. Delft: Delft University of Technology.

Kung, H. T., Luccio, F. & Preparata, F. P., 1975. On Finding the Maxima of a Set of Vectors. *Journal of the ACM,* 22(4), pp. 469-476.

Kurapati, V., 1995. *A Petri Net Based Methodology for Modelling, Simulation, and Control of Flexible Manufacturing Systems - Phd Thesis.* New Jersey: New Jersey Institute of Technology.

Laskari, E. C., Parsopoulos, K. E. & Vrahatis, M. N., 2002. Particle swarm optimization for minimax problems. *Proceedings of the 2002 Congress on Evolutionary Computation.*

Latha, K. & Rajaram, R., 2009. Tabu annealing: an efficient and scalable strategy for document retrieval. *International Journal of Intelligent Information and Database Systems,* 3(3), pp. 326-337.

Le, B., 2014. *Modelling railway bridge asset management.* Nottingham: Univeristy Of Nottingham, PhD Thesis.

Le, B. & Andrews, J., 2013. Modelling railway birdge asset management. *Journal of Rail and Rapid Transit,* 227(6), pp. 644-656.

Le, B. & Andrews, J., 2015. Petri Net Modelling of Bridge Asset Management using Maintenance Related State Conditions. *Structure and Infrastructure Engineering.*

Le, B. & Andrews, J., 2016. Modelling railway bridge degradation based on historical maintenance data. *Safety and Reliability,* 35(2), pp. 32-55.

Le, B. & Andrews, J., 2016. Petri net modelling of bridge asset management using maintenance-related state conditions. *Maintenance, Management, Life-Cycle Design and Performance,* 12(6), pp. 730-751.

Le, B., Andrews, J. & Fecarotti, C., 2017. A Petri net model for railway bridge maintenance. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability,* 231(3), pp. 306-323.

Lee, J. & Lee, J., 2009. Conversion of Ladder Logic Diagram to Petri Net using Module Synthesis Technique. *International Journal of Modelling and Simulation,* 29(1).

Liang, D., Liu, S. & Tao, Z., 2012. Based on Petri Nets and Hybrid Genetic-Tabu Search Approach to Scheduling Optimization for Dual-Resource Constrained Job Shop. *2nd International Conference on Electronic & Mechanical Engineering and Information Technology.*

Li, D., Wang, L. & Wang, M., 1999. Genetic algorithm and tabu search: A hybrid strategy. *IFAC Proceedings Volumes,* 32(2).

Litherland, J., 2019. *Whole System Approaches to Railway Asset Management, PhD thesis,* Nottingham: University of Nottingham.

London Underground Limited, 1999. *Environmental Performance Report,* London: London Transport.

Madanat, S., 2000. Optimal Inspection and Maintenance Policies for Infrastructure Systems Under Measurement and Prediction Uncertainty. *TRB Transportation Research Circular,* Issue 498.

Mahboob, Q., 2014. *A Bayesian Network Methodology for Railway Risk, Safety and decision support ,* Dresden: Technische Univeritat Dresden.

Maroni, A. et al., 2020. Using Bayesian networks for the assessment of underwater scour for road and railway bridges. *Structural Health Monitoring,* 0(0), pp. 1-15.

Mašović, S. & Stošić, S., 2015. Application of Semi Markov decision process in bridge management. *IABSE Geneva Conference 2015.*

Matsumoto, M. & Nishimura, T., 1998. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation,* 8(1), pp. 3-30.

McGinnity, B., Fitch, T. & Rankin, W., 1998. A Systematic and Cost-Effective Approach to Inspecting, Prioritising and Upgrading London Underground's Earth Structures. *Instituion of Civil Engineers,* Volume Proceedings of the Seminar "Value of Geotechnics in Construction", pp. 309-332.

Medjoudj, M. & Yim, P., 2007. Extraction of Critical Scenarios in a Railway Level Crossing Control Systems. *International Journal of Computers, Communications & Control ,* Volume 2 (3).

Melchers, R. E., 1999. *Structural Reliability Analysis and Prediction.* s.l.:John Wiley.

Morcous, M., 2006. Performance Prediction of Bridge Deck Systems Using Markov Chains. *Journal of Performance of Constructed Facilities,* Volume 20, pp. 146-155.

Morgan, M. G., Henrion, M. & Small, M., 1990. *Uncertainty: A guide to Dealling with Uncertainty in Quantitative Risk and Policy Analysis.* 1st ed. Cambridge: Cambridge University Press.

Naybour, S., Andrews, J. & Chiachio-Ruano, M., 2019. Efficient Risk Based Optimization of Large System Models using a Reduced Petri Net Methodology. *Proceedings of the 29th European Safety and Reliability Conference (ESREL 2019),* pp. 826-834.

Network Rail, 2004. *AHBC Typical Circuits - Power Supply Circuits,* Milton Keynes: Network Rail - Internal Access Only.

Network Rail, 2004. *Handbook for the Maintenance and Inspection of Level Crossings,* Milton Keynes: Network Rail - Internal Access Only.

Network Rail, 2008. *LED Light Unit X41000,* Milton Keynes: Network Rail - Internal Access Only.

Network Rail, 2011. *Signalling Design: Module X10 - Level Crossings: Automatic Half Barriers,* Milton Keynes: Network Rail - Internal Access Only.

Network Rail, 2012. *Drainage Asset Policy,* London: Network Rail - Internal Access Only.

Network Rail, 2012. *Electrical Power Asset Policy,* Milton Keynes: Network Rail - Internal Access Only.

Network Rail, 2012. *Signalling Asset Policy,* Milton Keynes: Network Rail - Internal Access Only.

Network Rail, 2014. *CP5 Earthworks Asset Policy,* London: Network Rail - Internal Access Only.

Network Rail, 2014. *Structures Asset Policy,* London: Network Rail - Internal Access Only.

Network Rail, 2016. *Bleasby Automatic Half Barrier Crossing Annual Review.* Milton Keynes ed. s.l.:Network Rail - Internal Access Only.

Ng, S.-K. & Moses, F., 1998. Bridge Deterioration Modelling using Semi-Markov Theory. *Structure Safety and Reliability,* Volume 1-3, pp. 113-120.

Noortwijk, J. & Klatter, H., 2004. The Use of Lifetime Distriubtions in Bridge Maintenance and Replacement Modelling. *Computers and Structures,* Volume 82, pp. 1091-1099.

Norris, J., 1997. *Markov Chains.* Cambridge: Cambridge University Press.

Office of Rail and Road, 2017. *Freight Rail Usage 2017-2018 Q1 Stastical Release,* London: Office of Rail and Road.

Office of Rail and Road, 2017. *Rail infrastructure, assets and environmental 2016-17 Annual Statistical Release,* London: Office of Rail and Road.

Office of Rail and Road, 2018. *Passenger Rail Usage 2017-2018 Q4 Statistical Release,* London: Office of Rail and Road.

Office of Rail and Road, 2018. *UK Rail Industry Financial Information,* London: Office of Rail and Road.

Office of Rail Regulation, 2003. *Train Derailment At Potters Bar 10 May 2002,* London: Office of Rail Regulation.

Office of Rail Regulation, 2006. *Train Derailment at Hatfield: A Final Report by the Independent Investigation Board,* London: HSE Books.

Omar, M., Salam, R. A., Rashid, N. A. & Abdullah, R., 2004. Multiple sequence alignment using genetic algorithm and simulated annealing. *Proceedings. 2004 International Conference on Information and Communication Technologies: From Theory to Applications, 2004..*

Palacios, J. J. et al., 2015. Genetic tabu search for the fuzzy flexible job shop problem. *Computers & Operations Research,* Volume 54, pp. 74-89.

Patra, A. P. & Kumar, U., 2009. Availability analysis of railway track circuits. *IMechE Part F: J. Rail and Rapid Transit,* 224(1), pp. 169-177.

Pecht, M. G. & Chang, M. H., 2012. Failure Mechanisms and Reliability Issues in LEDs. *Solid State Lighting Technology and Application Series,* Volume 1, pp. 43-110.

Pham, D. T. & Karaboga, D., 2000. *Intelligent Optimisation Techniques.* London: Springer.

Pinheiro, M. et al., 2015. SQI - A Quality Assessment Index for Rock Slopes. *3rd International Conference on Transportation Geotechnics.*

Podofillini, L., Zio, E. & Vatn, J., 2006. Risk-informed optimisation of railway tracks inspection and maintenance procedures. *Reliability Engineering and System Safety,* 91(1), pp. 20-35.

Pour, S. M., 2017. *Towards Signalling Maintenance Scheduling for European Railway Traffic Management System,* Denmark: Technical University of Denmark.

Power, C. et al., 2016. Development of an Evidence-based Geotechnical Asset Management Policy for Network Rail, Great Britain. *Procedia Engineering,* Procedia Engineering Procedia Engineering, 143(143), pp. 726-733.

Prescott, D. & Andrews, J., 2013. Investigating railway track asset management using a Markov analysis. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit,* 229(4).

Quezada, J. et al., 2014. Formal design methodology for transforimg ladder diagrams to Petri nets. *Internations Journal of Advanced Manufacturing Technology,* 72(1-4).

Quiroga, L. & Schnieder, E., 2011. A Simulation Approach to the Optimization of Railway Infrastructure Maintenance Strategies. *International Journal of Performability Engineering,* 7(6), pp. 545-554.

RAIB, 2010. *Derailment near Gillingham tunnel, Dorset 28 November 2009,* Derby: Rail Accident Investigation Branch, Department for Transport.

RAIB, 2010. *Derailment of a freight train near Stewarton, Ayrshire,* s.l.: Rail Accident Investigation Branch, Department for Transport.

RAIB, 2010. *Failure of Bridge RDG1 48 (River Crane) between Whitton and Feltham 14 November 2009,* Derby: Rail Accident Investigation Branch, Department for Transport.

RAIB, 2012. *Near miss at Ufton level crossing,* London: Office for Road and Rail.

RAIB, 2013. *Near-miss at Butterwood level crossing, North Lincolnshire.* Derby: RAIB.

RAIB, 2014. *Near-miss at Butterswood level crossing, North Lincolnshire,* Derby: Rail Accident Investigation Branch.

RAIB, 2018. *Collision at Stainforth Road level crossing,* Derby: Rail Accident Investigation Branch.

Railtrack, 1999. *Network Management Statement,* London: Railtrack PLC.

Rama, D. & Andrews, J., 2015. A holistic approach to railway infrastructure asset management. *International Journal of Performability Engineering,* Volume 11 (2), pp. 107-120.

Rayward-Smith, V., Osman, I. H., Reeves, C. & Smith, G., 1996. *Modern Heuristic Search Methods.* Chicester: Wiley.

Robelin, C.-A., 2003. *Facility-Level and System-Level Stochastic Optimization of Bridge Maintenance and Replacement Decisions Using History-Dependent Models, PhD Thesis..* Berkeley: University of California.

Roberts, C., Marquez, F. P. G. & Tobias, A. M., 2010. A pragmatic approach to the condition montioring of hydraulic level crossing barriers. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit,* 224(1), pp. 605-610.

Rodríguez-Díaz, F., García-Martínez, C. & Lozano, M., 2010. A GA-based multiple simulated annealing. *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona, Spain.*

Rodriguez, D., Zimmermann, A. & Silva, M., 2004. Two heuristics for the improvement of a two-phase optimization method for manufacturing systems. *2004 IEEE International Conference on Systems, Man and Cybernetics.*

Rothery, H. C., 1879. *Circumstances Attending the Fall of a Portion of the Tay Bridge on the 28th December 1879.,* London: George Edward Eyre and William Spottiswoode.

RSSB, 2002. *Guidance on the Preparation of Risk Assessments Within Railway Safety Cases,* London: Railway Safety.

RSSB, 2010. *Development of the All Level Crossing Risk Model: A History, 1993-2010,* London: RSSB.

RSSB, 2011. Investigating the economics of the 3rd rail DC system compared to other electrification systems, T950.

RSSB, 2017. *Annual Safety Performance Report 2016/2017,* London: RSSB.

RSSB, 2018. *Annual Safety Performance Report,* London: RSSB.

Rubinstein, R. Y. & Kroese, D. P., 2017. *Simulation and the Monte Carlo Method.* New Jersey: Wiley.

Ruktanonchai, N., Ruktanonchai, C., Floyd, J. & Tatem, A., 2018. Using Google Location History data to quantify fine-scale human mobility. *International Journal of Health Geographics,* 17(28).

Sachan, S. & Donchak, H., 2020. Generalized Stochastic Petri-Net Algorithm with Fuzzy Parameters to Evaluate Infrastructure Asset Management Policy. *IEEE International Conference on Fuzzy Systems.*

Saffarzadeh, V. M., Jafarzadeh, P. & Mazloom, M., 2010. A Hybrid Approach Using Particle Swarm Optimization and Simulated Annealing for N-queen Problem. *World Academy of Science, Engineering and Technology, International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering,* 4(7), pp. 947-951.

Santo-Reyes, J. & Beard, A. N., 2016. Inquiry into the fire on Heavy Goods Vehicle shuttle 7539 on 18 November 1996. *Journal of Rail and Rapid Transit,* 231(8), pp. 850-876.

Schere, W. & Glagola, D., 1994. Markovian Models for Bridge Maintenance Management. *Journal of Transportation Engineering,* Volume 120, pp. 37-51.

Shafahi, Y. & Hakhamaneshi, R., 2009. Application of a Maintenance Management Model for Iranian Railways Based on the Markov Chain and Probabilistic Dynamic Programming. *Scientia Iranica,* 16(1).

Shang, H., Berenguer, C. & Andrews, J., 2017. Delayed maintenance modelling considering speed restriction for a railway section. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability,* 231(4), pp. 411-428.

Sharma, S. et al., 2018. Data-driven optimization of railway maintenance for track geometry. *Transportation Research Part C,* Issue 90, pp. 34-58.

Sianipar, P. & Adams, T., 1997. Fault-Tree Model of Bridge Element Deterioration Due to Interaction. *ASCE Journal of Infrastructure Systems,* 3(3), pp. 103-110.

Sobanjo, J., 2009. State transition probabilities in bridge deterioration based on Weibull sojourn times. *Structure and Infrastructure Engineering,* 7(10), pp. 747-764.

Stott, P. F., 1987. *Automatic Open Level Crossings, A Reivew of Safety,* London: Her Majesty's Stationery Office.

Thomopoulos, N. T., 2012. *Essentials of Monte Carlo Simulation: Statistical Methods for Building Simulation Models.* 1st ed. New York: Springer.

Tinoco, J., Snaches, S. & Miranda, T., 2015. Application of Markov chains in the prediction of rock slopes degradation. *Proceedings of the XVI ECSMGE Geotechnical Engineering for Infrastructure and Development.*

Turner, S., 2008. *Review of Network Rail's All Level Crossing Risk Model (ALCRM) ,* Buxton: Office of Rail Regulation.

Twiss, E., 1996. *Relay Ladder Logic and Petri Nets for Discrete Event Control Design: A Comparative Study - PhD Thesis.* New Jersey: New Jersey Institue of Technology.

Uff, J., 2000. *The Southall Rail Accident Inquiry Report.* Sudbury: HSE Books.

Voß, S. & Fink, A., 2012. Hybridizing reactive tabu search with simulated annealing. *LION'12 Proceedings of the 6th international conference on Learning and Intelligent Optimization.*

Westinghouse Signals, 1999. *AHBC Typical Circuits - Track Control Circuits - X00410,* Milton Keynes: Network Rail - Internal Access Only.

Westinghouse Signals, 1999. *AHBC Typical Circuits X00030,* s.l.: Internal Access only.

Xiaobing, Y. et al., 2018. Reliability Evaluation Methodology of Complex Systems Based on Dynamic Object-Oriented Bayesian Networks. *IEEE Access,* Volume 6, pp. 11289-11300.

Yang, S. L., Frangopol, D. M. & Neves, L. C., 2005. Optimum maintenance strategy for deteriorating bridge structures based on lifetime functions. *Engineering Structures,* 28(2), pp. 196-206.

Yang, Y. N., Pam, H. J. & Kumaraswamy, M. M., 2009. Framework Development of Performance Prediction Models for Concrete Bridges. *Journal of Transportation Engineering ASCE,* 135(8), pp. 545-554.

Yianni, P., Dovil, R., Neves, L. & Andrews, J., 2016. A Petri-Net-Based Modelling Approach to Railway Bridge Asset Management. *Structure and Infrastructure Engineering,* Volume 13 (2), pp. 287-297.

Yuping, Z., Shouwei, J. & Chunli, Z., 2005. A very fast simulated re-annealing algorithm for the leather nesting problem. *The International Journal of Advanced Manufacturing Technology,* Volume 25, pp. 1113-1118.

Zaharah, I. S., Yue, W. L. & Somenahalli, S., 2011. An Assessment of Heavy Vehicle Safety at Level Crossing Using Petri Nets: South Australia Case Studies. *Journal of the Eastern Asia Society for Transportation Studies,* Volume 9.

Zhao, J., chan, A. & Burrow, M., 2009. A genetic-algorithm-based approach for scheduling the renewal of railway track components. *Journal of Rail and Rapid Transit,* 223(6).

11. Appendix

11.1 Appendix A – Equations of Motion

The appendix to chapter 5 deriving the equations of motion used within the model.

Assuming a constant rate of deceleration the velocity-time graph is as below.

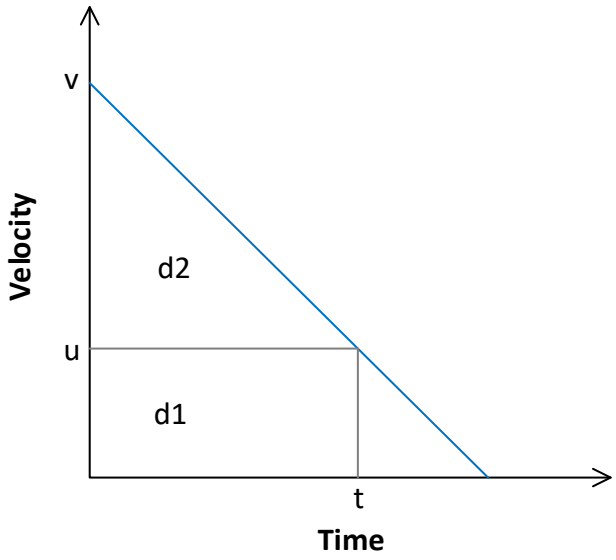

Figure 11-1 Velocity-Time Graph

From this, equation (1) is derived.

$$d_t = d_1 + d_2 = \frac{(u+v)t}{2} \qquad 11\text{-}1$$

A second equation is derived from the relationship between acceleration and time.

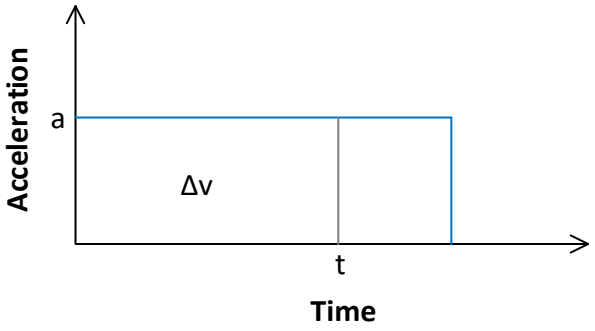

Figure 11-2 Acceleration-Time Graph.

From which equation (2)

$$v - u = at \quad (11\text{-}2)$$

To determine acceleration as a product of initial velocity and distance travelled, the final velocity is set to zero, u = 0. Combining equations (1) and (2) then yields a third equation for the rate of deceleration for an initial speed and total stopping distance.

$$a = \frac{v^2}{2d_t} \quad (11\text{-}3)$$

Now an equation will be derived by combining equations (1) and (2) where the final velocity may take a non-zero value:

$$d = \frac{(2v - at)t}{2} \quad (11\text{-}4)$$

Re-arranging

$$at^2 - 2vt + 2d = 0 \quad (11\text{-}5)$$

Solving for t and selecting the positive root only

$$t = \frac{v + \sqrt{v^2 - 2ad}}{a}, \text{ Where } a = \frac{v^2}{2d_t} \quad (11\text{-}6)$$

Using equation (6) the time required for a vehicle to cover a known distance whilst decelerating can be determined.

11.2 Appendix B – Model Parameters

The appendix to chapter 6, detailing parameters used within the asset management models.

Failure Distributions and Parameters

Failure Modes – Control System Relays	**Distribution and Parameters**
High Resistance Relay Contact	Weibull: Scale Factor 12000, Shape Factor 1.5
Welding Relay Contact	Exponential: Mean 3650000
25% Capacitor Degradation	Exponential: Mean 5000
Failure Modes – Track Circuits	
High Resistance Relay Contact	Weibull: Scale Factor 15000, Shape Factor 1.5
Welding Relay Contact	Exponential: Mean 3650000
Failure Mode - Treadles	Distribution and Parameters

Open Circuit Failure	Weibull: Scale Factor 7000, Shape Factor 1.5
Closed Circuit Failure	Weibull: Scale Factor 7000, Shape Factor 1.5
Failure Modes – Lights	
Bulb Failure – Open Circuit	Weibull: Scale Factor 8000, Shape Factor 1.5
Bulb Failure – Closed Circuit	Weibull: Scale Factor 12000, Shape Factor 2
Fixtures and fittings - damage	Exponential: Mean 15000
Failure Modes – Barriers	
Damage to Structure	Weibull: Scale Factor 18000, Shape Factor 1.5
Main Bearing Failure	Weibull: Scale Factor 17000, Shape Factor 1.5
Hydraulic Ram Failure	Weibull: Scale Factor 15000, Shape Factor 1.5
Motor Pump Unit Failure	Weibull: Scale Factor 9000, Shape Factor 1.5
Solenoid Valve Stuck Open Failure	Weibull: Scale Factor 7000, Shape Factor 1.5
Solenoid Valve Stuck Closed Failure	Weibull: Scale Factor 70000, Shape Factor 1.5
Proving Contact Failure	Weibull: Scale Factor 4000, Shape Factor 1.5
Failure Modes – Cabin	
Cabin Degradation	Weibull: Scale Factor 4000, Shape Factor 1.5
Failure Modes – Power Supply	
Power Supply	Weibull: Scale Factor 12000, Shape Factor 1.5
Back-Up Battery	Weibull: Scale Factor 12000, Shape Factor 1.5
Mains Power Cut	Exponential: Mean 365

Mains Power Cut Resolution	Exponential: Mean 0.04
Spurious Trip	Exponential: Mean 365
Failure Modes – Road Surface	
Road Surface Block Deterioration	Weibull: Scale Factor 10000, Shape Factor 1.5

Repair costs and times

Barriers	**Cost**	**Repair Time**
Barrier Support	5000	180minutes
Proving Contacts	400	40minutes
Main Bearing	300	120minutes
Hydraulic Ram	500	90minutes
Motor-Pump Unit	1000	120minutes
Solenoid Valve	150	90minutes
Track Circuit Repair		
Track Circuit Relay	600	60minutes
Treadle	1000	60minutes
Road Traffic Lights	**Cost**	**Repair Time**
LED Bulb	100	30minutes
Fixtures and Fittings	700	120minutes
Flasher	500	60minutes
Power Supply	**Cost**	**Repair Time**
Power Supply	1000	60minutes
Batteries	1000	20minutes
Cabin	**Cost**	**Repair Time**
Cabin repair	3000	8hrs
Control System	**Cost**	**Repair Time**
Relay Replacement	800	30minutes
Road Surface	**Cost**	**Repair Time**
Road Surface Block Replacement	2500	60minutes

Interventions

Intervention	Distribution and Parameters
Opportunistic Inspection	Exponential: Mean = 28
Scheduled Inspection	Lognormal: S = 0.697, M = 1.367, Mean = 1.367, SD = 0.697
Routine Maintenance	Lognormal: S = 1.006, M = 2.133, Mean = 14, SD = 7
Urgent Maintenance	Lognormal: S = 0.145, M = -1.908, Mean = 0.15, SD = 0.08
Opportunistic Renewal	Exponential: Mean = 2000
Scheduled Renewals	Lognormal: S = 1.613, M = 3.304, Mean = 100, SD = 50

11.3 Appendix C – Regular Petri Net Design Tools

Several tools to design Petri Nets were created in MS Visio as part of this work, they are available for download at the web address at the end of this appendix. These tools are particularly effective when designing large Petri Nets. A stencil allows common Petri Net elements to be dragged and dropped onto a Visio sheet, connecting using Visio's snap fit system. A macro can then export the Petri Net as drawn in its entirety, this is the most valuable of all the tools as transcribing a Petri Net from a drawing into a machine readable format for simulation can takes days of tedious and error prone work. Lastly, a macro can import the results of a simulation into the Visio Petri Net drawing to show how the state of the Petri Net places change as the simulation progresses. This can highlight faults within the logic of the Petri Net and is invaluable for debugging a Petri Net.

Note, use of these tools requires the professional edition of MS Visio.

Visio Stencil

The stencil comprises standard Petri Net elements which work with both the export and debugging tools. The elements within this stencil are shown below.

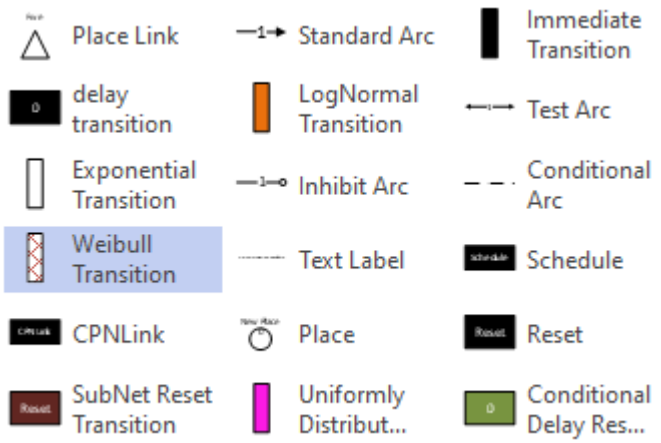

Figure 11-3 Visio Stencil

Shapes from the stencil are simply dragged and dropped onto the Visio sheet. When dropped, many shapes will ask for values to be input.

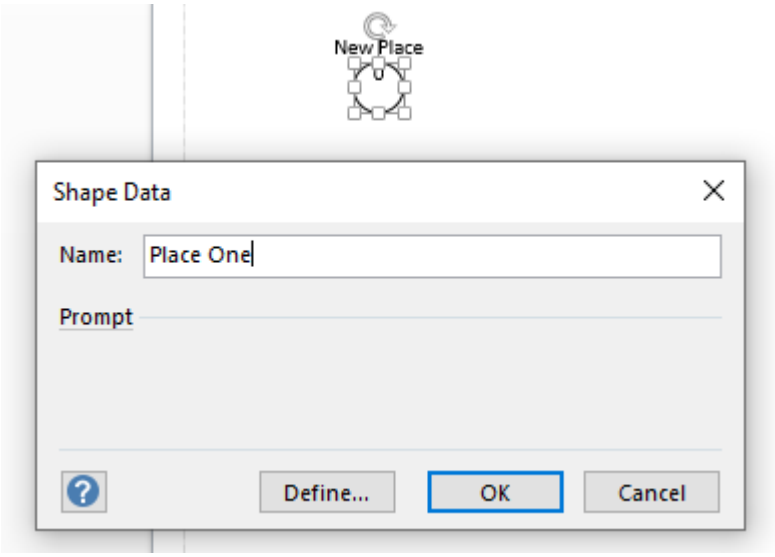

Figure 11-4 Shape Data Input Dialogue Box

Opening the shape data window allows additional values to input and changed.

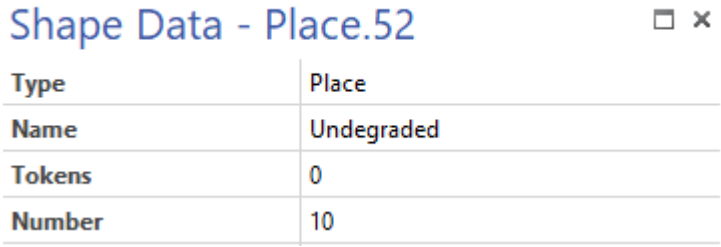

Figure 11-5 Shape Data Window

Each element has a ‘type’, this informs later macro-based tools what type the Petri Net element is. Each element also has a number to identify each element. These should not be altered. Elements are numbered automatically when the Petri Net is exported.

Adding other Visio shapes may cause errors using the tools described later, the ‘Text Label’ shape within the stencil allows text to be overlain on the Petri Net without causing errors. If other Visio shapes outside the stencil are added to the sheet they should be given the shape data field ‘Type’, and this field set to ‘Label’ to prevent errors.

Export Tool

The export tool reads the Petri Net drawn using the stencil described above and writes its structure to an excel file, where it may be imported in simulation software. This tool uses a macro written in VBA. To use the tool, first ensure the PetriNetData.xlsx sheet is within the same folder as the Visio file, or the export will fail. press Alt-F11 to bring up the macro menu. Select and run the macro named ‘ExtractMapping’, for small Petri Nets the macro should complete instantly, however for Petri Nets with thousands of elements there may be substantial delay.

The data is formatted within the sheet as follows.

Detail: This sheet contains three values. The value in Cell A1 shows the number of places within the Petri Net, Cell A2 shows the number of transitions, and Cell A3 shows the number of inhibit arcs.

Place: Each row corresponds to the Places numbered on the Visio sheet. The value indicates the starting number of tokens within the Place.

PlaTra: This is short for Place to Transition Mapping. It details the arcs which enable transitions. Rows on this spreadsheet correspond to a numbered Transition on the Visio sheet. For each arc enabling a transition there are two values. The first details the Place number the arc originates at, the second its multiplicity. Where there are multiple arcs enabling a transition, these are listed in the following Cells.

TraPla: This is short for Transition to Place mapping. It is very similar to PlaTra described above, but details arcs which originate at transitions and terminate at places. Note, any Test Arcs will feature in both PlaTra and TraPla.

Transition: This sheet details transition parameters, where row numbers correspond to each transition. The first Cell values indicates the type of transition using a numerical value,

subsequent Cells contain transition parameters. For delay transitions, '1' indicates their type, with the following number their delay. Exponentially distributed stochastic transitions are numbered as '4', with the following number their delay. Further details are within the macro itself.

Inhibit: This lists each Inhibit Arc, each row number corresponds to the number of the arc. The first number per row is the Transition the Arc Inhibits, the second number is the originating Place, the last number is the number of tokens needed to inhibit the Transition.

Debugging Tool

The debugging tool imports data obtained by simulating the Petri Net. It allows each timestep of the simulation to be stepped through to confirm if a Petri Net is functioning as expected, and to locate errors. As each time step advances, places which have gained tokens are highlighted in green, and those which have lost tokens are highlighted in red.

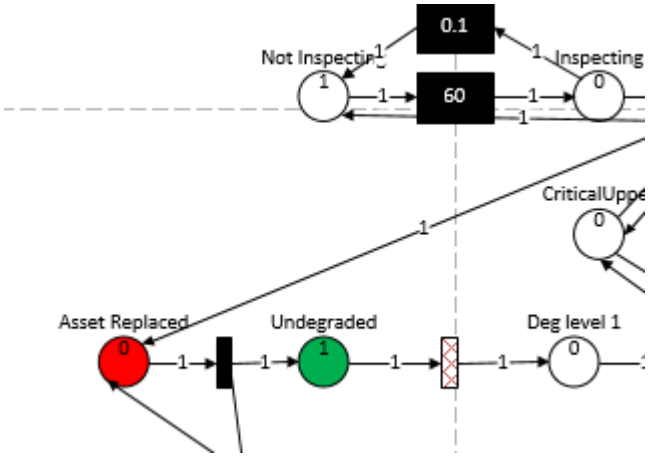

Figure 11-6 Example of Place Highlighting When Using the Debugging Tool.

To use this tool, place the 'Time Box' object from the stencil onto the Visio sheet containing the Petri Net. This box contains data fields which store the current time step which is vital for using the macros. Running the 'StepThroughData' macro advances the timestep, the 'ResetPlaces' macro restores the Petri Net to its original state. Specific time steps may be accessed by changing the 'TimeLogElement' data field within the Time Box and then running 'StepThroughData'.

These macros expect three files. The first file is called 'TimeLogDetails.xlsx', and contains only the number of timesteps which the Petri Net was simulated for. The second is called 'TimeRecord.xlsx' and contains a list of the times for each timestep. The last file is called

‘PlaceRecord.xlsx. Each row of this spreadsheet represents the state of each place at that timestep, with sequential timesteps matching up with the times listed in ‘TimeRecord.xlsx’.

These tools and example files can be downloaded from:

https://github.com/PetriNetTools/VisioTools