DITTO Tool Integration: Basic Prototypes for Interfacing

DITTO Project Deliverable 1.2, Milestone 2.

Version 1: September 2015.

Phillip James, Faron Moller and Markus Roggenbach.

Swansea University, UK.

Abstract

The acronym DITTO stands for “Developing Integrated Tools To Optimise Rail Systems”. This report scopes how to integrate the various tools and software used within DITTO. We elaborate on models, abstractions and transformations in general; present a vision of how the various tools and software could be combined towards a useful tool chain; we elaborate on the various safety analyses developed at Swansea University; and we discuss first steps towards an interfacing with the RailML format as a starting point for our tool chain.

DITTO

www.dittorailway.uk

1 Motivation

The acronym DITTO stands for “Developing Integrated Tools to Optimise Rail Systems”. This report scopes how to integrate the various tools and software used within DITTO.

All project partners within DITTO use models of rail systems. These models are designed to serve different purposes and therefore look at these systems from different perspectives. However, what they have in common is that they aim to describe, potentially, the very same rail system. Furthermore, it is understood that computer support is necessary due to the inherent complexity of these models. One of the objectives of the Swansea project partner is to investigate how to establish an integrated tool chain for analysing and optimising a rail system through the modelling of the three partners within the DITTO consortium.

Such an endeavour needs close collaboration. How to use the various tools and software needs to be demonstrated and documented, especially the relation between the rail system and its model, how this model is encoded into an input format, what commands are used for the analysis, and how the expected result shall look like. Such information is needed in order to reproduce analyses and optimizations – as a first step towards integration into a tool chain. This report is a step in this direction and outlines our ideas for such a tool chain. However, these collaborations are in still in their beginning. Consequently, with having more and deeper knowledge, we might want to refine our ideas. Central to our envisioned tool chain is the OnTrack tool – see DITTO Deliverable 1.1: OnTrack Documentation, Version 2: June 2015.

Our report is structured as follows. In the beginning, we elaborate on models, abstractions and transformations in general. Then, we present a vision (with many thanks to John Preston for his ideas and pictures) of how the various tools and software could be combined towards a useful tool chain. Then, we elaborate on the various safety analyses developed in Swansea. Finally, we discuss first steps towards an interfacing with the RailML format as a starting point for our tool chain. Appendix A compiles an overview of the tools under discussion, Appendix B provides an overview on the DITTO project, a glossary can be found in Appendix C.

2 Models, Abstractions and Transformations in Computer Science

This project centres on the definition, manipulation and analysis of models of rail systems, for safety, reliability, and timetable efficiency. The three research groups involved have different backgrounds (computer scientists, traffic engineers, transport operations researchers) and thus different conceptual understandings of what a model

of a railway entails. We thus provide generic definitions for model as well as the related notions of abstraction and model transformation to set the scene.

Model

> (1) A miniature representation of a thing, with the several parts in due proportion; sometimes, a facsimile of the same size.
> (2) Something intended to serve, or that may serve, as a pattern of something to be made; a material representation of or embodiment of an ideal; sometimes a drawing or a plan; a description of observed behaviour, simplified by ignoring certain details.

Building models is at the core of any scientific and engineering discipline. Scientists need models to interpret their data and make predictions; and traditional engineering products such as bridges and aeroplanes are never built until models of them have been developed and studied to understand the characteristics of the product. These models may be small-scale versions of the product which are tested in wind tunnels; or they may be purely abstract models described on paper which are then analysed more formally, for instance through computer simulations.

Models come in all shapes and sizes, and are designed to capture specific aspects of the thing they represent. Consider, for example, the following two uses of simple railway models.

- If we are interested in teaching the history of the development of railway locomotives, then full scale working replicas would be fun, but probably inconvenient; small scale working replicas might do, or even non-working replicas. Meaning (1) is appropriate.
- If we are interested in developing strategies for safe shunting, then a child's train set might do. But we could also make do with a paper-and-pencil model with a sketch of the track and buttons representing the engines and rolling stock; or a computer model with a graphical interface and a simulator might even be more useful. Meaning (2) is appropriate.

Models allow systems to be understood, and their behaviour predicted, only within the scope of the model; they may give incorrect descriptions and predictions for situations outside the scope of their intended use. For example, a toy train set would not be much use if we were interested in the stresses and strains induced in real rolling stock when shunting.

Abstraction

> The act or process of leaving out of consideration one or more properties of a complex object so as to attend to others.

Abstraction is an important part of model building: identifying those features that are essential for inclusion in the model and separating out those features that can be neglected since the essential elements do not rely on their presence.

As an example of the use of abstractions, Ordnance Survey (OS) maps are used by walkers in Britain who usually want to know where they are, where they are going, and how long it will take. OS maps are to scale and include grid reference pairs allowing the user to pinpoint locations very accurately. On the other hand, the London Underground train map and A to Z street atlas have different formats from OS maps, and from each other, as they serve different purposes.

Transformation

> The process of changing the character or appearance of something in order to improve it.

In particular, model transformation refers to an automatable process for converting between different models of an entity. The different tools which may be used for analysing different aspects of a system will naturally work on different models of the system, defined by the different abstractions applied to identify the relevant entities of importance to the tool at hand. Model transformation allows conversion from one model to another.

3 Vision

The vision for the DITTO project is the integration and development of three toolsets for analysing and improving safety and capacity at both the macro and micro levels of railway design. Over the following section, we present the envisaged tool chain and the design processes it will support. We thank John Preston for providing the overall idea and also the workflow pictures.

3.1 Swansea Safety Analysis

The starting point to our proposed tool chain will begin with safety verification and analysis using the OnTrack tool from Swansea University, see Appendix A for a tool summary. Considering Figure 1, based on a provided CAD track plan and the associated control tables, a user draws a Scheme Plan using the OnTrack graphical front end (right path). These scheme plans are models formulated relative to OnTrack's DSL meta model. It is work in progress, to read a RailML representation and to transform it into a model relative to OnTrack's DSL meta model, see Section 4.2 RailML Integration for first work in this direction.

A scheme plan is then the basis for subsequent workflows that support its verification, simulation, analysis etc. Scheme plans can be translated to formal specifications in various (specification) formalisms. These transformations turn a Scheme Plan into a Formal Specification Text ready for verification.

Once a formal model has been generated, it can then be simulated or verified using the tools associated with the formal specification language that has been used for

generation. For example, ProB can be used for animating and verifying CSP||B, SPASS can be used for verifying CASL, TimedCSP Simulator can be used for simulating and visualising train runs. If safety verification happens to fail, then the process can be re-performed with an appropriately adjusted scheme plan.

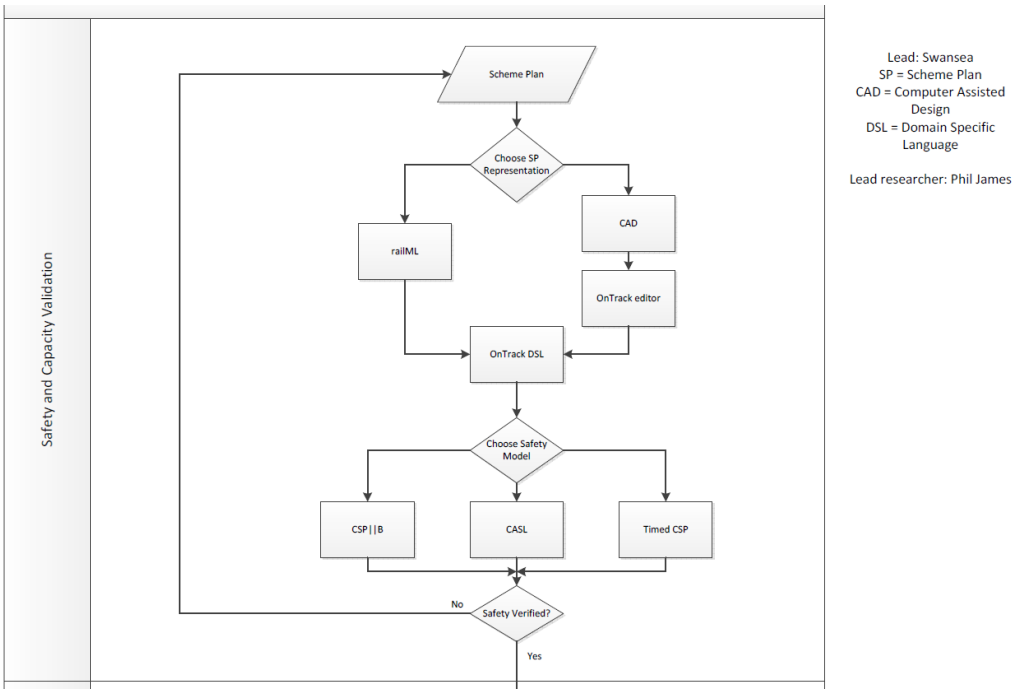


Figure 1: **Safety analysis with OnTrack.**

3.2 OCCASION Timetable Optimization

Once a scheme plan has been successfully verified the OnTrack tool will provide generation features for other models from the DITTO context. These models then can be simulated and analysed with respective tools.

OnTrack will generate Layouts for a static optimisation phase using the OCCASION toolset from Southampton, see Appendix A for a tool summary. Figure 2 then illustrates the process undertaken using the OCCASION toolset. Firstly, this toolset will perform analysis using the Capacity Utilisation Index (CUI) to determine performance implications. From generated performance scenarios, OCCASION will first carry out nodal optimisations at the meso level using stochastic optimisation. If the resulting design is seen to require further optimisation, then a suitably revised timetable is produced and the process is repeated.

From this point, OCCASION will carry out network optimisation at the macro level. Again, if the design resulting from this phase of optimisation is seen to require further

optimisation, then a suitably revised timetable is produced and the whole process is repeated.

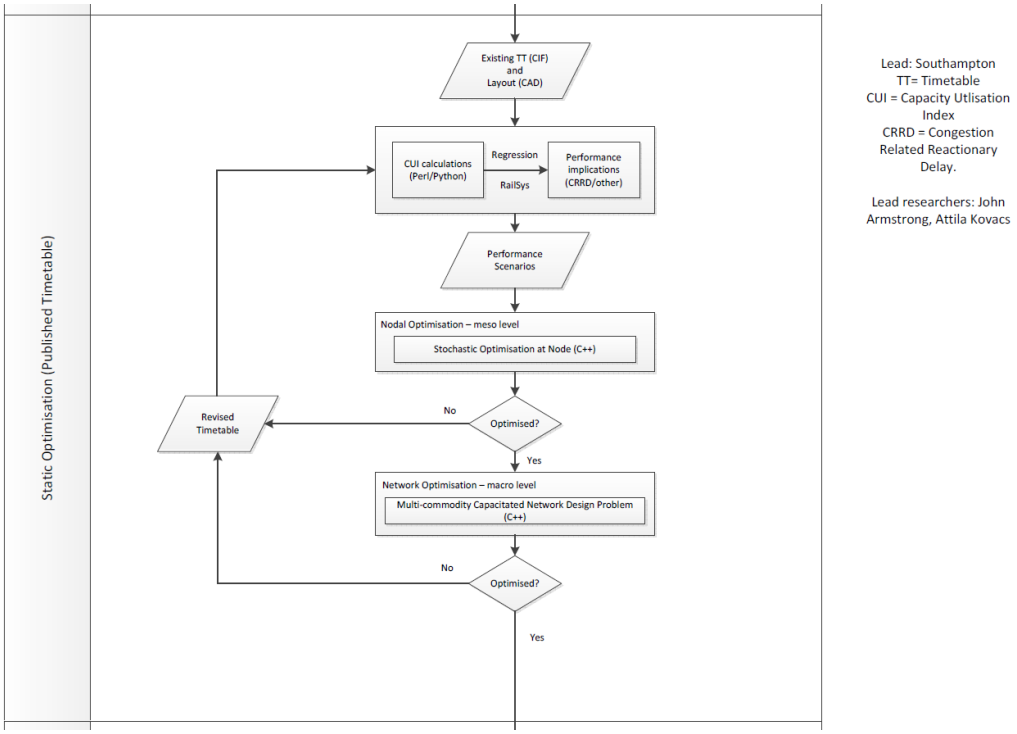


Figure 2: **The optimisation process using OCCASION.**

3.3 Tracula Timetable Optimization

Taking as input the final optimised timetable and performance scenarios output from the OCCASION toolset, the third step in the tool chain, see Figure 3, will be micro Level optimisation using the Tracula toolset from Leeds University, see Appendix A for a tool summary. Based on a dynamic simulation process devised in the Dracula tool for modelling cars, the Tracula tool will consider algorithmic adjustments to train control for dynamic rescheduling of trains. Where adjustments are required, they will initially be made algorithmically. If, when testing these adjustments with train controllers further adjustments are deemed to be required, then these are carried out manually and the whole process is repeated.

3.4 Extension with Machine Learning

The final step in the tool chain is still to be implemented as an extension to the OCCASION tool. This step is based around using a machine learning approach to perform

final adjustments to the scheduling for trains. After this machine learning process, if adjustments are still deemed to be required, then the whole dynamic rescheduling phase is undertaken once more. If no further adjustments are required, then the process ends with a optimised scheduled scheme plan.

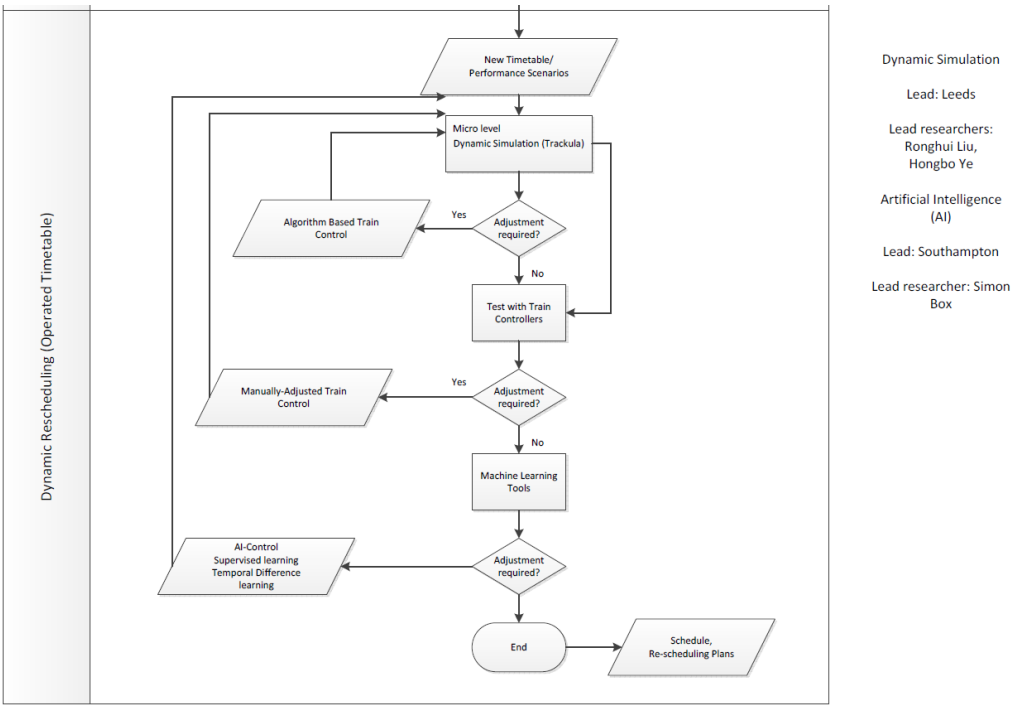


Figure 3: **The timetable optimisation process with Tracula.**

4 Reporting on the State of the Art: Safety Analysis

In order to support the tool chain presented in Section 3, a number of model transformations will need to be designed and implemented. Over the following sections, we shall report on the state of the art of each of these tools and the firsts steps that have been undertaken in designing a series of model transformations to support the presented tool chain.

4.1 Safety Analysis Results

With regards to safety, in close cooperation with Siemens Rail Automation, Chippenham, UK, Swansea currently investigates two research strands:

- safety of solid state interlockings; and
- safety of ERTMS level 2 as a whole.

It is planned that, later in the project, this portfolio is extended by ERTMS level 3 as well. Objective of our approach is to support the design of these systems by providing a safety analysis early in the design cycle, for a detailed discussion see Section 1 ``Supported Processes from Industry" in DITTO Deliverable 1.1: OnTrack Documentation, Version 2: June 2015.

Our analyses of solid state interlockings directly contributes to ERTMS Level 2:

a) ERTMS Level 2 includes interlockings as a central component on which safety relies upon.
b) Some groups, e.g., Haxthausen/Peleska, argue that safety analysis for ERTMS can be reduced to the analysis of solid state interlockings by treating ERTMS Level 2 marker boards as ``virtual signals'' They write: ``By introducing the concept of virtual signals, we have been able to handle the assignment of movement authorities in a way that is very similar to the situations where conventional signals are used.'' [1].

In the following we provide an overview on the various approaches on safety analysis in Swansea.

4.1.1 Safety Analysis for Solid State Interlockings

Towards the automatic check for safety conditions, together with Siemens Rail Automation, we have explored a number of modelling formalisms and technologies:

- The algebraic specification language CASL and the automatic theorem prover SPASS have been used in the PhD dissertation by Phillip James. His research led to technology reported in the following publication:

  P. James and M. Roggenbach: *Encapsulating Formal Methods within Domain Specific Languages: A Solution for Verifying Railway Scheme Plans.* Mathematics in Computer Science 8(1):11-38, 2014.

- The specification language CSP||B and the model checker ProB have been used in research carried out within the SafeCap project. This led to technology reported in the following publication:

  P. James, F. Moller, H. N. Nguyen, M. Roggenbach, S. Schneider, and H. Treharne: *Techniques for modelling and verifying railway interlockings.* International Journal on Software Tools for Technology Transfer 16:685-711, 2014.

  The technology published in 2014 is limited to uni-directional scheme plans only. In a number of follow-up works, we have been able to extend it to bi-directional railways. However, some results concerning abstraction are still missing. A publication compiling a comprehensive technology for bi-directional railways is in preparation. Further plans include:

- automatising of this technology in the OnTrack tool,
- adding counter-example presentation, and
- performing user studies with signalling engineers

➢ The process algebra CSP and the model checker FDR are used in the MRes project by Mike Smith. An initial comparison with safety verification based on CSP||B models shows a significant speed-up (see Figure 4). We reported on this result in:

M. Smith, H. N. Nguyen, M. Roggenbach:
Revisiting modelling and analysing railway control systems in CSP.
Pre-Proceedings of AVoCS 2015.

	2015 CSP modelling		2012 CSP||B modelling	
Rail Yard	Time Taken	States Checked	Time Taken	States Checked
Station	0.11s	5156	21.34s	2441
Single Junction	0.17s	15,336	82.75s	8645
Double Junction	0.72s	984,720	4056.66s	240,655

Figure 4: **Verification comparison.**

Again, this work is limited to uni-directional scheme plans only. Further steps will include:

- developing the technology for bi-directional scheme plans and
- automating it in the OnTrack tool.

4.1.2 Safety Analysis for ERTMS Level 2

The switch from classical railway signalling systems to ERTMS/ETCS train control poses a number of research questions for the formal methods community:

- Can safety be guaranteed?
- Can formal methods be used to confirm that such a switch improves capacity?
- Is it possible to predict capacity using formal methods?

To address such questions it is necessary to develop and analyse timed or hybrid models. ERTMS/ETCS Level 2 takes speed and braking curve of each individual train into account. These determine the train's breaking point well in advance of the end of authority that the signalling system had granted to this train. Such an approach is in contrast to classical signalling systems, which treat all trains in the same way. Therefore, they need to be designed for worst braking. Consequently, in formal safety analysis, these systems can be treated on a purely logical level, ignoring the aspect of time -- see, e.g., our above discussion of safety analysis for solid state interlockings.

An ERTMS/ETCS system consists of a controller, an interlocking, a radio block centre, track equipment, and a number of trains. While the ERTMS/ETCS standard details the interactions between trains and track equipment (e.g., in order to obtain concise train position information) and radio block centre and trains (e.g., to hand out movement authorities), the details of how controller, interlocking and radio block centre interact with each other are left to the suppliers of signalling solutions such as our industrial partner Siemens Rail Automation UK. We model and analyse this hybrid system of systems in the Real-Time Maude system, which is a multi-purpose tool with support for executable specification, simulation and verification:

P. James, A. Lawrence, M. Roggenbach, and M. Seisenberger:
Towards safety analysis of ERTMS/ETCS Level 2 in Real-Time Maude
Submitted to an International Workshop.

4.1.2.1 An ERTMS Level 2 narrative

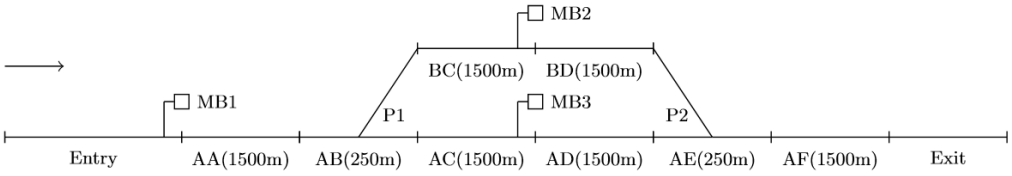
An important, first step in the development of a model is to compile a narrative summarising which aspects of the system shall be covered. In the following, we give a narrative of ERTMS Level 2.

ERTMS Level 2 extends classical railway signalling. To this end its location specific design extends the classical notion of a scheme plan by information used for the radio block centre (RBC). ERTMS safety analysis also requires train characteristics such as maximum speed, acceleration and braking curves.

Scheme Plans

A scheme plan is a well-established concept within the railway domain. Figure 5 depicts such a scheme plan for a pass-through station. It comprises of a track plan, a control table, release tables and RBC tables. The track plan provides the topological information for the station. It consists of 8 tracks (e.g., BC) each with a length, 3 marker boards (e.g., MB1), and two points (e.g., P2). A topological route is a piece of railway on which a train can travel, (typically) between two marker boards (e.g., from MB1 to MB2). The control table describes under which conditions a route can be set (It is a design decision, if a topological route appears in the control table. The routes in the table are those available for use by trains). For example, a train can only proceed on route 1A when point P1 is in normal (straight) position and tracks AA, AB and AC are clear, i.e., currently not occupied by any train. The release tables is used to implement sequential release, a technique to improve capacity. Release tables describes when a point is again free to move after being locked for a particular route. For example, when sending a train on route 1A, point P1 is free to move already, when this train has reached track AC. This allows to send another train on route 1B before the first train has reached track AD and thus completely left route 1A. Finally, the RBC tables are used for movement calculations within the RBC.

We consider open scheme plans with entry and exit tracks only. Furthermore, we assume that marker boards are placed at the end of tracks, and that the speed limit is the same for all tracks.



Interlocking Tables:

Route	Clear Tracks	Normal	Reverse
1A	AA, AB, AC	P1	
1B	AA, AB, BC		P1
2	BD, AE, AF		P2
3	AD, AE, AF	P2	

Point	Route	Release
P1	1A	AC
P1	1B	BC
P2	2	AF
P2	3	AF

RBC Tables:

Current Position	Continuation Routes
Before MB1	1A, 1B
Before MB2	2
Before MB3	3

Granted Route	EOA
1A	3249m
1B	3249m
2	6499m
3	6499m

Figure 5: **Scheme plan for a station.**

ERTMS System Architecture

Once a scheme plan has been designed, a number of control systems are implemented based around it. In the following we systematically identify the entities of ERTMS, describe their abstract behaviour and determine the abstract information flow between them in line with the design by Siemens Rail UK, see Figure 6.

The controller (manual or computerised) is responsible for controlling the flow of trains through the railway network. The controller completes this task by sending ``route request'' messages to the interlocking. These route requests are dependent upon elements such as the current timetable to be adhered to and details on congestion within the network. For simplicity, we abstract from ``route cancel'' and `acknowledgement'' messages.

The interlocking is responsible for setting and granting requested routes. Once the controller has requested a route, the interlocking will use information on current track occupation and point settings (from the track equipment) to determine if it is safe for the requested route to be set. Whether a route can be set or not is computed in a process based upon the conditions stipulated by the control table, see Figure 5. Once the interlocking has checked that all points on the route are free to move or already in the right position, it will send a ``route available'' message to the RBC. This informs the RBC that the route is free for use, however it is not yet reserved for a train. The RBC initiates the process of locking a route for a particular train by sending a ``request to

proceed'' message to the interlocking. On receiving this message, the interlocking will then ensure that, based on the control table, all tracks for the route are free and that the points are indeed locked in the required positions. Once this step is completed, the interlocking sends a ``proceed'' message to the RBC indicating that a train can use the route.

The RBC's main responsibility is to take the route information presented by the interlocking and use it to manage the movement of trains across geographic positions on the railway. To do this, the RBC and trains use the notion of a movement authority. A movement authority is an area of geographical railway that a train is permitted to move within. The furthest point along the railway to which a train is permitted to move is indicated by a point known as the end of authority (EoA) which is given to a train by the RBC. As a train moves across the railway network, it uses beacons on the track to continually calculate its position. When it is nearing its EoA, it makes a new ``movement authority request'' to the RBC indicating that it would like its movement authority to be extended. After receiving this request, the RBC will map the physical location of the train to an available continuation route that has been presented to it by the interlocking. At this point, there should be maximally one route available that matches a particular train. This is ensured by the requests from the controller and also the ability of the interlocking to deny requests for conflicting routes. This calculation is performed based on a look-up table designed as part of the RBC for a scheme plan, an example of such a table is provided in Figure 6. It will then issue a ``request to proceed'' message to the interlocking for this route. Once the RBC has received a ``proceed'' message from the interlocking, it will compute, based on the route that has been granted, a new EoA for the train. Again, this information is provided by a look-up table, see Figure 5. This new EoA is then finally sent as a ``movement authority'' message to the train.

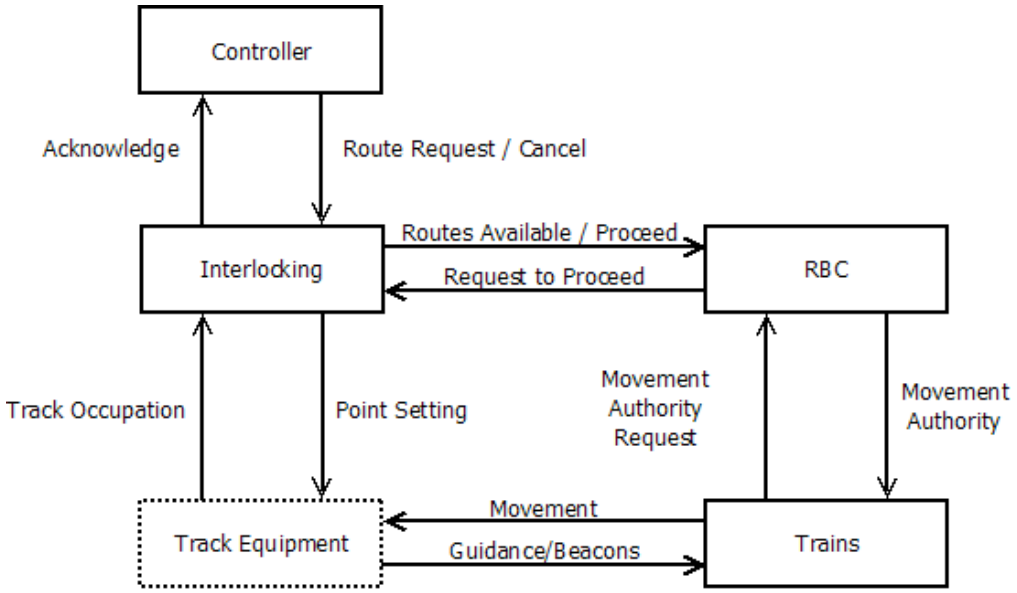


Figure 6: **Information flow in ERTMS.**

With regards to trains, their behaviour is parameterised by maximum speed, acceleration and braking curves. We make a maximum progress assumption for trains, i.e., trains are running as fast and as far as possible. Namely, if a train has a movement authority beyond its current position it will accelerate towards its maximum speed. When the maximum speed is reached, the train will continue to travel at this speed. Whilst accelerating or travelling at maximum speed the train will start braking at the last possible time in order not to overrun its EoA. Trains are guided by the track layout, respecting the positions to which the interlocking has set points. As trains move along the track, track equipment senses track occupation and reports it to the interlocking.

We assume that track equipment, (points, track circuits, beacons etc.) functions correctly and that points move instantaneously. This is justified as our verification aim is to establish correctness of the location and train specific design parameters for an ERTMS system for a single geographic region. Therefore, we refrain from modelling track equipment.

Safety Conditions

In the context of ERTMS, several high level safety conditions have been discussed such as collision freedom or derailment on a point. For the moment, we focus on collision freedom, i.e., excluding the possibility that two trains collide. In the context of classical signalling systems, this property usually is formulated logically, e.g., we verify that there are never two trains on the same track. In contrast, for ERTMS we rather consider the physical invariant: the distance (in meters) between trains never falls below a minimum threshold.

4.1.2.2 Modelling in Real Time Maude

Our modelling in Real-Time Maude comprises of:

- an analysis of the ERTMS system (see above)
- a description of its information flow (see above), and
- a concise model in Real Time Maude.

This model is astonishingly small: it consists of around only 1000 lines of code. We believe this is due to the advanced concepts, especially the object orientated features that Real Time Maude offers.

Through simulation we have demonstrated that our model exhibits a number of expected behaviours. Furthermore, by systematic error injection, we have shown that safety in ERTMS depends on all its components. This simulation and error injection give us confidence that our model is valid.

The following table presents our model checking results. As an initial configuration we always consider an empty rail yard with no trains in it. This rail yard can only be entered by two trains with different maximum speeds. The table shows verification times and the number of rewrite steps of three rail yards against a random controller and a round-

robin controller. All verification results are on a PC running Xubuntu 14.04.2 with i7 4790 @3.60Ghz and 32GB RAM.

Scheme Plan	Round Robin Controller Unbounded	Random Controller In Time 300
Junction	0.4s / 975,010 rewrites	12.25s / 24,730,270 rewrites
Pass-through Station	1.09s / 2,298,862 rewrites	393.64s / 279,958,198 rewrites
Three Platform Station	2.39s / 4,793,317 rewrites	370.73s / 301,818,278 rewrites

Figure 7: **Model checking of ERTMS.**

The table shows that unbounded model checking is successful when control is restricted, e.g., to our round-robin controller. However, when using our random controller, the state space vastly increases. Thus, we provide results for up to a given time bound of 300s. Note that this time is enough to ensure that two trains can travel completely through the Junction and Station scheme plan. This is highlighted by the model checking times for the three platform station: within 300 seconds the model checking takes less time due to the fact that trains cannot travel as far into the track plan in this time limit.

It is future work to explore further, more complex rail yards, including bi-directional ones. On the practical side we intend to extend our modelling with further controller strategies and more complex train progression behaviour. On the more theoretical side, we plan to investigate completeness and finitisation.

5 Reporting on the State of the Art: Integration

When considering tool integration, RailML [2] is a language that has been designed as a data exchange format to support interoperability within railway industry applications. In this sense, and due to its rising popularity, it makes sense to consider how our current tools can be extended to process railway scheme plans written in RailML. RailML is also used by Birmingham University in their DEDOTS toolset for the FuTRO programme. Supporting RailML as a language would also allow cross project collaboration.

As OnTrack forms the backbone of our tool chain, we propose an XML model transformation should be designed to support transformation of RailML scheme plans into OnTrack. Both OnTrack's internal domain specific language and RailML are XML based languages. Thanks to this, it is possible to employ XSLT to convert between these two representations. We shall now give a brief overview of both RailML and XSLT before discussing a translation route for OnTrack.

5.1.1 RailML

RailML or the Railway Markup Language is a common exchange format developed by the RailML consortium, consisting of a number of European railway partners. The project

started in early 2002, and has seen the release of several version of RailML with the current stable release being RailML 2.2. As with most XML based languages, RailML is composed out of several sub-languages each of which having a formal definition written in the XMLSchema language. Currently, there are five main sub-languages:

- Timetable: for the description of elements related to timetables.
- Rolling Stock: for the description of elements related to rolling stock.
- Infrastructure: for the description of physical infrastructure and topological information.
- Interlocking: for the description of interlocking data.
- Common: A general language for describing elements common to the above languages.

The specification of these languages and relevant documentation can be downloaded from www.railml.org. For illustrative purposes, Figure 8 presents a small example of RailML provided in the RailML documentation:

```
<tracks>
  <track id="T1">
    <trackTopology>
      <trackBegin id="TB1"/>
      <trackEnd id="TE1">
        <connection id="C1" ref="C2"/>
      </trackEnd>
    </trackTopology>
  </track>
  <track id="T2">
    <trackTopology>
      <trackBegin id="TB2">
        <connection id="C2" ref="C1"/>
      </trackBegin>
      <trackEnd id="TE2"/>
    </trackTopology>
  </track>
</tracks>
```

Figure 8: **Sample RailML for two connected tracks.**

The sample XML specifies two tracks connected to one another. We can see there are two track elements with identifiers T1 and T2. The beginning and end of each track is also given an identifier. Finally, we can see a connection element that links the end of track T1 to the beginning of T2.

5.1.2 XSLT

XSLT, or the Extensible Stylesheet Language: Transformations, is a language that is designed for the transforming of one XML document into another. The main idea behind the language is to structurally map an instance document of one XML language into an instance document of another XML language. The main motivation behind XSLT is the

transfer of data between computer programs or, more broadly, between different organisations.

As a motivating example for the use of XSLT, we consider a small example for translating a simple XML file into a web page written in XHTML, based on the example presented by Michael Kay [3]. Consider the following XML document:

```
<? xml version="1.0" encoding="iso-8859-1" ?>
<? xml-stylesheet type="text/xsl" href="hello.xsl" ?>
<text>Hello World</text>
```

Figure 9: **XML document encoding the text "Hello World".**

This document contains two so called processing instructions that are relevant to tools reading the file and one line of structured content. The first line, a processing instruction denoted by "<?...?>"indicates which data encoding is used within the XML file. The second line, another processing instruction, tells tools how to layout the XML when presenting the file to a user. For this, the line provides a type attribute highlighting that the layout information will be provided by a XSL (XML Stylesheet Language) file, and another attribute pointing to the concrete file "hello.xsl". Finally, the last line of the document contains a single XML element "<text>" that contains the character data "Hello World". Here the important information on how to transform this document into an XHTML document is provided by the hello.xsl file that, for example, could contain the following snippet:

```
<? xml version="1.0" encoding="iso-8859-1" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="...">

<xsl:template match="/">
        <xhtml>
        <head>
                <title>Hi!</title>
        </head>
        <body>
                <p><xsl:value-of select="text"/></p>
        </body>
        </xhtml>
</xsl:template>

</xsl:stylesheet>
```

Figure 10: **XSLT stylesheet for presenting the XML as XHTML .**

This snippet highlights one of the main features of XSLT namely a template rule. Template rules are used to specify translations for particular elements within an XML document. Given an XML document, this template rule matches root nodes of the document it is given. It then applies the translation specified inside the template rule. The translation above simple outputs a block of html (both the head and body), where in

particular, the body contains a paragraph where we insert some information from the input XML document. In this case, the <xsl:value-of select=”text”/> code selects anything that is contained within “text” elements of the input document. In our sample document this would be the text “Hello World”. Given this, the result of applying the XSLT stylesheet in Figure 10 to the XML document in Figure 9 is the XHTML document shown in Figure 11. If rendered using a web-browser, this would now display a webpage presenting the text “Hello World” to the user.

```
<xhtml>
        <head>
                <title>Hi!</title>
         </head>
         <body>
                <p>Hello  World<p>
         </body>
</xhtml>
```

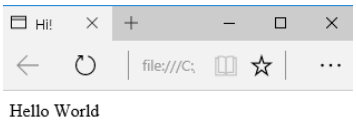


Figure 11: **Result of applying our stylesheet to the sample XML document.**

5.1.3 First Steps towards a Translation

In the context of the XSLT examples presented above, one can consider how we might create a XSLT transformation for transforming RailML documents into OnTrack’s internal XML representation. Such a translation is currently being designed for the infrastructure and common sub-languages from RailML. As a first sample of this translation, the XSLT stylesheet in Figure 12 provides a translation of the RailML document encoding two tracks given in Figure 8 into OnTrack’s internal representation.

The stylesheet in Figure 12 contains two template rules. The first rule that matches the root element of the input document is simply used to create a new “RailDiagram” instance within OnTrack’s DSL. It then uses an apply-templates call to recursively find and transform all tracks from the input RailML instance document into their representation in OnTrack’s DSL. The second template rule is where much of the work is completed. This rule constructs a new “hasUnits” element from OnTrack’s DSL to represent each track within the input document. It also sets up the required attributes for this element. For example, we can see that the attribute “name” is set to have the value of the “id” of the input track. Similarly, if there are connections at the beginning or end of a track, these connections are transformed into the “hasC1” and “hacC2” attributes within OnTrack’s DSL. More details on OnTrack’s DSL are available in [4].

```
<xsl:stylesheet version="2.0"
                xmlns:onTrack ="http://bjoernercomplete/1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
        <onTrack:RailDiagram>
              <xsl:apply-templates select="tracks/track"/>
        </onTrack:RailDiagram>
```

```
  </xsl:template>

  <xsl:template match="track">

    <xsl:element name="hasUnits">
          <xsl:attribute name="type"> onTrack:Linear</xsl:attribute>
          <xsl:attribute name="name"><xsl:value-of select="@id"/></xsl:attribute>
          <xsl:if test="trackTopology/trackBegin/connection">
                              <xsl:attribute name="hasC1">
                    <xsl:value-of select="trackTopology/trackBegin/connection/@id"/>
                    </xsl:attribute>
          </xsl:if>
          <xsl:if test="trackTopology/trackEnd/connection">
                    <xsl:attribute name="hasC2">
                    <xsl:value-of select="trackTopology/trackEnd/connection/@id"/> </xsl:attribute>
          </xsl:if>
    </xsl:element>
 </xsl:template>

</xsl:stylesheet>
```

Figure 12: **XSLT stylesheet for translating RailML track elements into OnTrack's DSL representation of track elements.**

6 Summary

This report provides a first scoping for the development of a DITTO tool chain. To this end, it discussed the notion of a model and provided a vision on tool integration. Reporting on progress, it gave a current account on safety analysis in Swansea and described how RailML could be integrated into the OnTrack tool. Appendix A compiles an overview on the tools and software within the DITTO project. Appendix B gives an account of the DITTO project aims.

7 References

[1] L. Vu, A. Haxthausen and J Peleska. Formal Modeling and Verification of Interlocking Systems Featuring Sequential Release. FTSCS. Springer 2015.

[2] RailML Homepage: http://www.railml.org/ Last accessed: September 2015.

[3] M. Kay. XSLT 2.0 and XPath 2.0 Programmer's Reference. 4th Edition. Wrox 2008.

[4] P James, F Moller and M Roggenbach. DITTO Deliverable 1.1: OnTrack Documentation, Version 2: June 2015.

Appendix A: Tool Summary

A1. Tracula: Leeds University

What does it do?

Tracula aims to model rail traffic network following a micro simulation approach. It is based on the Dracula tool developed for road traffic.

What inputs does it need?

The user provides individual vehicle characteristics (including, e.g., vehicle type and length, minimum safety headway, acceleration and deceleration, etc.), a network representation, and possibly further data.

What kind of results can one achieve with it?

Tracula aims to provide on-line animation of traffic movements, individual vehicles' space-time trajectories and statistical measures.

What computer platforms does it run on?

Windows

Is it under a licence?

University of Leeds License.

Where can one download the tool?

The Dracula tool for road traffic can be found at
http://www.its.leeds.ac.uk/software/dracula/

A2. Occasion tool: Southampton University

What does it do?

Occasion optimises rail time tables with consideration of nodal capacity constraints.

What inputs does it need?

The user provides a track plan and train service information (including number of tracks, the network of nodes and links, platforms, headways at different sections, train speeds).

What kind of results can one achieve with it?
The Occasion tool computes an optimised time table.

What computer platforms does it run on?

Windows 32-bit.

Is it under a licence?

Licence details available on request.

Where can one download the tool?

Software is available on request.

A3. OnTrack: Swansea University

What does it do?

OnTrack is a model generator for producing formal rail models represented in various specification languages from graphical representations of scheme plans.

What inputs does it need?

The user enters a scheme plan into OnTrack by drawing a track plan in the OnTrack graphical editor and providing control and release tables.

What kind of results can one achieve with it?

The OnTrack tool fully automatically translates the given input into a faithful formal model (represented as plain text) in a chosen specification language. Languages supported include CASL, CSP, TimedCSP, CSP||B.

What computer platforms does it run on?

Windows, Linux, Mac. It is possible to package OnTrack with Eclipse.

Is it under a licence?

OnTrack is open source software under the GNU General Public License v3.0.

Where can one download the tool?

`http://cs.swansea.ac.uk/~cspj/index.php/tool-development`

Appendix B: DITTO Overview

OnTrack shall serve as a common platform for tool integration between the DITTO partners Southampton, Leeds, and Swansea. Surrey will also continue to collaborate as a DITTO sub-site of Swansea. Below we give an outline of the main aims and objectives of the DITTO project.

This project will combine and build upon the work undertaken by three of the project teams in the RSSB/EPSRC Capacity at Nodes programme (Challenging Established Rules for Train Control, OCCASION and SafeCap) with a view to making a significant contribution to meeting the requirements of the Future Traffic Regulation Optimisation (FuTRO) programme and to UK rail related research more generally. It will contribute to FuTRO by establishing basic principles and proofs of concept and by developing optimisation formulations, algorithms and processes that will deliver a step change in rail system performance and help to meet future customer needs. This will be done by taking into account developments in human and automatic control on trains and in control centres (particularly related to ERTMS) and by making better use of data, particularly with respect to time and position of trains. The proposal contains four inter-related and complementary technical strands, with specific aims as follows:

1) Safety – although FuTRO currently resides in the management layer of railway operations, safety is a fundamental and overriding consideration in operations management and control. The safety strand of the proposal underpins the traffic management strands, allowing optimisation activities to proceed in the knowledge that safe operating conditions are being maintained and that theoretical capacity limits are not being exceeded. The tools developed will also have generic applications to traffic regulation.

2) Reliability – the trade-offs between the provision of additional train services, and the resultant increases in capacity utilisation, and the maintenance of service quality are an area of particular interest within the industry, and this strand of the proposal aims to quantify these trade-offs so as to develop timetables that optimise capacity utilisation without compromising service reliability.

3) Dynamic simulation – micro-level data on the status of individual trains will be combined to produce an optimal, macro-level outcome, transmitting the system-wide needs back to the microlevel, so that individual train movements can be optimised within overall system requirements.

4) Network integration – we will produce optimised timetables that can be adjusted in real time through dynamic simulation. We will examine the scope for artificial intelligence to combine our optimisation and simulation tools to produce tractable solutions to real-time traffic control.

Appendix C: Glossary

Common Acronyms

AI – *Artificial Intelligence.*
CAD – Computer Aided Design.
CRRD – Congestion Related Reactionary Delay.
CUI – Capacity Utilisation Index.
DSL – Domain Specific Language.
ECTS – European Train Control System.
EPSRC – Engineering and Physical Sciences Research Council.
EoA – End of Authority.
ERTMS – European Railway Traffic Management System.
RBC – Radio Block Centre.
RSSB – Rail Safety and Standards Board.
SP – Scheme Plan.
TT – Timetable.

Glossary

Abstraction – The act of disregarding properties of a system, when constructing a model of the system, which are irrelevant to the analysis to be carried out on the model.
B – A specification language for software systems based on sets and predicate logic.
CASL – A specification language for software systems based on predicate logic and induction. Standing for "Common Algebraic Specification Language", it was developed by a consortium to be general purpose and subsume existing specification languages.
CSP – A specification language for describing reactive systems (i.e., processes). Standing for "Communicating Sequential Processes", it is algebraic in nature with analysis techniques based on algebraic manipulations.
CSP||B – A specification language for describing systems; it combines the two languages *CSP* and *B*, with the process-based aspects catered by the features of CSP and data catered by the features of B.
DEDOTS – A toolset developed at Birmingham University; it stands for "Developing and Evaluating Dynamic Optimisation for Train Control Systems".
DITTO – The name of the project; it stands for "Developing Integrated Tools To Optimise Rail Systems".
FDR – A toolset for checking properties of formal models specified in CSP.
FuTRO – Standing for "Future Traffic Regulation Optimisation", this is the programme under which the DITTO project is funded.
HTML – A basic language for creating web pages; it stands for "HyperText Markup Language".
Model – A representation of something – specifically a system or process – which can be formally analysed.
Model checking – The (automated) means by which a desirable property of (a model of) a system is demonstrated to be fulfilled.
OCCASSION – A precursor project funded by the RSSB/EPSRC; standing for "Overcoming Capacity Constraints: A Simulation Integrated with Optimisation at Nodes", its objective was to identify and investigate innovative methods of increasing the capacity of nodes

(i.e. junctions and stations) on the railway network, without substantial investment in additional infrastructure.

OnTrack – A toolkit developed at Swansea University under the SafeCap project (2011-2013) which automates workflows for railway verification, starting with graphical scheme plans and finishing with automatically generated formal models set up for verification.

ProB – A toolset for checking properties of formal models specified in B.

RailML – Standing for "Railway Markup Language", it is a data exchange format for interoperability in railway industry applications.

Real Time Maude – A specification language for software systems which are sensitive to timing conditions which is based on equational logic.

SafeCap – A precursor project funded by the RSSB/EPSRC; standing for "Overcoming the railway capacity challenges without undermining rail network safety" its objective was to develop modelling techniques and tools for improving railway capacity while ensuring that safety standards are maintained.

SPASS – An automated theorem prover for predicate logic with equality.

TimedCSP – An extension of the CSP specification language which allows the modelling of systems which are sensitive to timing constraints.

Transformation – The process of changing between different models of the same system in order to make it amenable to different toolsets for different analyses.

Tracula – A dynamic simulation model developed by the University of Leeds, based on their car following model Dracula, which is to be used to adjust train running speeds in real time.

XHTML – Standing for "Extensible HTML", it is part of the family of XML markup languages which extends HTML.

XML – A family of markup languages for encoding documents in a format which is both human- and machine-readable.

XMLSchema – A description of a type of XML document expressed in terms of constraints on the structure and content of documents beyond the basic constraints imposed by XML itself.

XSL – A family of languages used to transform and render XML documents; it stands for "Extensible Stylesheet Language".

XSLT – A language for transforming between XML documents; it stands for "Extensible Stylesheet Language: Transformations".