The SafeCap toolset for improving railway capacity while ensuring its safety

Alexei Iliasov, Newcastle University
Alexander Romanovsky, Newcastle University

Abstract—**The on-going RSSB/EPSRC UK SafeCap project develops modelling techniques and tools for improving railway capacity while ensuring that safety standards are maintained. This paper reports recent SafeCap results on designing a Domain Specific Language (DSL), a verification infrastructure and the approaches to estimating and improving capacity.**

Keywords—**signalling safety, railway capacity, formal modelling, railway junction.**

I. Introduction

Ensuring and demonstrating railway system dependability is crucial for the way our society operates. Formal methods have been successfully used in developing various railway control systems. The best-known examples include the use of the B Method [6] for designing various metro and suburban lines, and airport shuttles all over the world [9,2]. The formal methods here are used to trace the requirements to system models and to ensure and demonstrate the system safety. Our work builds on this work.

Our aim is to develop methods and tools that allow railway engineers to improve the node and junction capacity while guaranteeing operational safety. By capacity we understanding the capability of a given railway layout and associated signalling rules to provide a certain quality of services determined by specific traffic patterns.

To achieve capacity improvement we offer engineers to model various changes of the layouts and signalling rules in the vicinity of the junctions or stations and evaluate the effect of the changes on the overall capacity. The safety of the original and modified models are checked by automated tools via a translation of railway topology and signalling rules into formal verification conditions.

A common domain language for the description of railway schemes and signalling rules was necessitated by the desire to apply and compare the differing modelling techniques used in the project to a common set of problems as defined by the industrial partner of the project, Invensys. The domain language is meant to satisfy the requirements of a railway engineer, who prefers to see a detailed layout presented in a way as close as possible to the established practice, and a researcher, who needs to work with a relatively small and precisely defined set of concepts. It is, perhaps, impossible to completely meet such conflicting requirements in a single language but we see it as one of the project goals to make a solid effort in this direction.

An effort at an improvement requires changing railway topology and signalling rules. Any such change must be analysed from the position of operational safety: absence of collisions and derailments. A complex project may require hundreds or thousands of changes (such large figures are achievable with mechanised transformation patterns) making an efficient and scalable tool support paramount. The aim of automated verification predicates a certain degree of formality in the definition of language concepts.

The techniques described in the paper form the foundation of an open modelling environment called the SafeCap Platform [4]. The environment offers a intuitive modelling interface that may be used by a railway engineer, not trained in any manner in the use of such tools, to enter and update railways schemas, access, 'by a press of a button', a range of verification tools and try finds ways to improve junction capacity.

The rest of the paper is organized as follows. Section II describes the SafeCap domain specific language. The SafeCap approach to verification of safety properties is explained in Section III. Sections IV and V discuss capacity criteria and ways to improve junction capacity. Finally, Section IV briefly introduces a tooling platform supporting the SafeCap method.

II. Domain Specific Language

In SafeCap we study in isolation complex stations and junctions of a railway network. The focus is on the analysis and improvement of the throughput of a given layout in the context of service pattern defined by the wider context of the railway network. In geographical terms, a layout could span over an area of several miles.

SafeCap offers a fairly compact core DSL. This is due to the limited scope of our study - railway junctions rather than complete networks - and also the desire to have an open-ended language where differing secondary elements may be defined for specific problem classes. The domain language defines the foundational concepts for the modelling of railway capacity. It attempts to capture, at a suitable level of abstraction, track topology, route and path definitions and signalling rules. By design, the language is extensible: one can dynamically define further attributes for all the predefined language elements and these are automatically picked up by the SafeCap Platform editor at no extra implementation effort.

The basic element of a SafeCap schema is the definition of railway topology.

A *track* is a piece of physical railway track; it is characterised by length, measured in meters and a *height map* - a detailed profile of track gradient used for capacity assessment. Track is not directed - a train may travel over track in either direction.

A *node* is a fictitious device gluing tracks into a continuous layout. Each track is associated with two nodes which it connects.

This work is sponsored by RSSB and EPSRC UK.

Nodes and tracks of a schema define an undirected graph known as the *topology graph of a schema*. Such a graph must not contain self-loops (tracks that start and end on the same node) and must satisfy certain restriction on the number of tracks connected to any given node (i.e., the *degree* of a node). A valid topology contains nodes of degrees 1 to 4. There can be no isolated nodes (degree 0) and nodes with degree higher than 4[1]. The following are the valid node types: a *boundary node* (degree 1) that marks the boundary between the considered layout and the rest of a railway network; a normal node (degree 2) used to connect two pieces of track; a *point node* (degree 3) and a *diamond crossing* (degree 4).

On the diagram in Fig. 2[2], node positions are identified by letters; boundary nodes are highlighted by pale green (or grey) squares. Boundary nodes may be further qualified into *sink* (minus sign) and *source* (plus sign) nodes to define nodes where trains only appear or disappear. On the diagram in the figure, the number under a track defines the track length in meters

To have several trains travelling over the same physical topology it is required to have an additional layer of structuring. This layer assists in the formulation of train movement principles that ensure safety. The following are the logical structuring concepts of a schema.

A *point* is a machine that moves loose ends of rails to alter the path of a train. In logical terms, a point identifies a sub-graph of the overall layout available at any given moment.

An *ambit*[3] is a connected sub-graph of the topology graph equipped with a *train detection circuit*. An ambit is able report whether there is a train anywhere on ambit tracks but not the number of trains, if there is more than one, exact positions or the occupation status of individual tracks. We do not consider the possibility of detection circuit failure.

There are two important cases of ambits - those that include points and those that do not. The latter are called *sections* and are made of linear track and diamond crossings. The former are known as *junctions* and, in addition to linear tracks and diamond crossings, contain at least one point. From the modelling viewpoint, sections are *static* entities - they do not change their configuration over time.

A *route* is a continuous connection between two nodes of a layout. In *route-based signalling*, a route is also the unit of track reservation - a process that ensures the availability of some free track in front of a train. Formally, a route is a path sub-graph of a topology graph. Overlapping routes (sharing one or more ambits) are permitted.

A *line* is a sequence of routes; such a sequence must form a path through a layout starting and ending on boundary nodes.

A *rule* is a logical condition attached to a route or a point of a schema. The formation of a rule may follow the traditional approach of specifying clear and occupied ambits, point states, signals aspects and etc. In SafeCap, we also offer the option of writing a rule as an arbitrary first-order logic formulae expressed over schema elements.

Note that we do not discuss signals - we regard them as optional track-side equipment indicating route states. We use, however, signal pictograms to denote route boundaries on a schema diagram. This is merely a notational convenience.

The diagram in Fig. 1 illustrates the relationship between the major concepts of the SafeCap DSL.

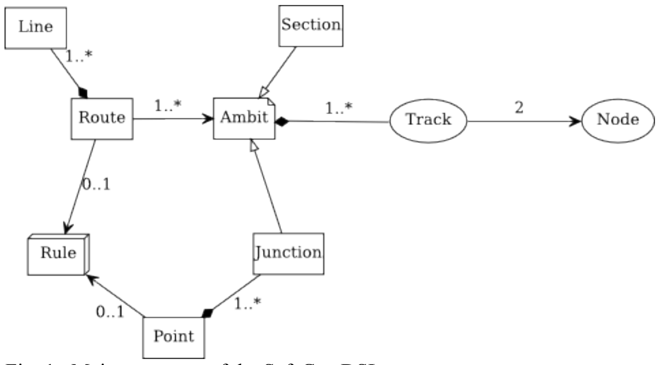


Fig. 1. Major concepts of the SafeCap DSL.

On the diagram the arrows signify: containment (diamond), inheritance (triangle) and aggregation (arrowhead). Circled are elements of physical topology. Rules may be defined in a traditional tabular form or as logical constraints in a notation specific to the verification back-end.

The SafeCap DSL is a *formal* language: a schema is interpreted as a hybrid transition model - a model mixing continuous and discrete behaviours. The discrete part is employed to derive static verification conditions (that is, theorems) and, as a supplementary technique, serve in the discovery of transition traces leading to the violation of safety conditions. The continuous part refines the discrete part with the notions of train acceleration/deceleration, point switching, driver reaction times and so on.

Static verification conditions take the form of logical constraints over the elements of a schema. They express requirements to schema topology, formation of routes, placement of speed limits (a square box with a number in Fig. 2 diagram), and signalling rules of routes and points. If all such conditions are discharged, it is guaranteed, for all possible rail traffic that the schema is free from potential collisions and derailments.

There are 32 such conditions and cumulatively they form the *theory of the SafeCap schema*s. A schema is valid if it is proven to be a model in this theory.

III. Safety Verification

We briefly discuss how the two central safety conditions of railways are addressed in the SafeCap DSL.

A schema must be free from collisions. A collision happens when two trains occupy the same part of a track. In terms of our DSL, we can only speak about potential collisions due to simultaneous occupation of the same ambit by two trains.

1 We do not, at this time, consider non-standard points and crossings.

2 This diagram is generated by the SafeCap Platform.

3 We have opted for a new way to refer to a train detection circuit to avoid confusion with inconsistently used existing terminology.

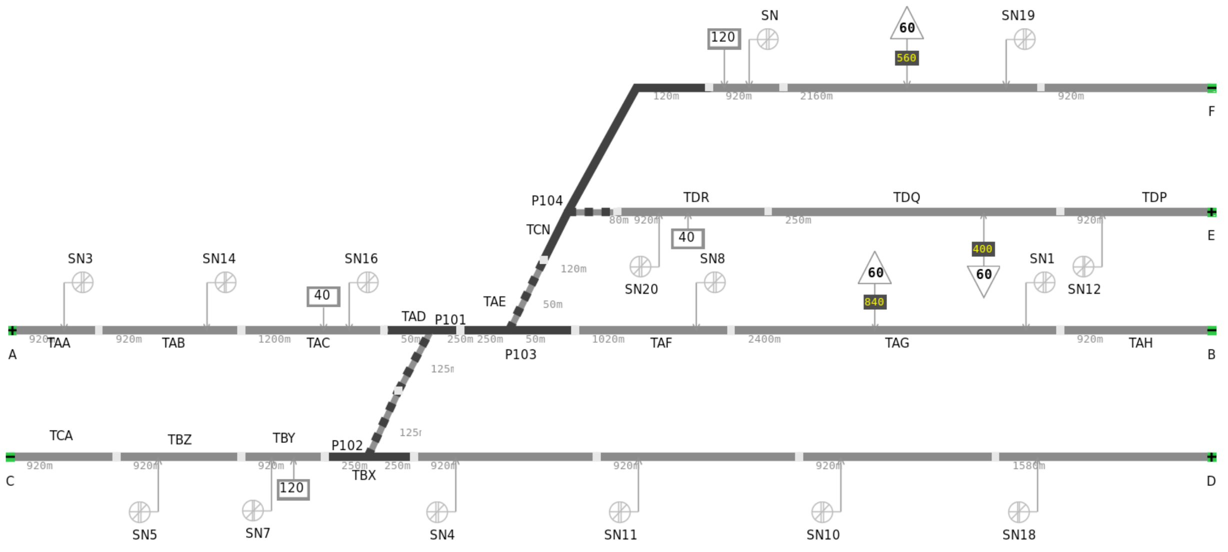


Fig. 2. Double lead junction scenario.

One cannot show the absence of collisions without explaining how a train moves around a railway layout, i.e., the driving rules. If a train (its driver and on-board equipment) ignore route state indication there is really nothing to prevent train collisions. Hence, the absence of collisions is not a property of a schema per se but that of a combination of a schema and train driving rules (which may including assistant technologies such as TPWS and ATP). Of course, it is essential to establish that a schema is safe under some sensible driving rules. We have validated the DSL semantics through an encoding of an ATP-assisted driving in Event-B model and proving that DSL topology and signalling constraints imply the absence of collisions.

A schema must be free from derailments. A derailment may happen when a train moves over a point changing its configuration. There is an explicit rule attached to each point requiring that a point may be moved only when all the routes containing the point are not occupied. The condition is sufficient to ensure that none of track points are occupied when a point moves, provided that trains are driven in compliance with route states.

The primary mechanism for verifying schema safety is constraint solving. An automated tool mechanically derives verification conditions from the definition of a schema and translates them into an input notation of an SMT-LIB compliant SMT solver. By the standards of SMT solvers, our models are relatively small so a result is reported nearly instantaneously.

One downside of applying constrain solving is the absence of any useful feedback to indicate the source of a problem should an error be discovered. To compensate for this, whenever an SMT solver detects a problem, the tool also runs a model checker. Unlike solver that operates on derived, a model checker directly explores the state space of a discrete transition system that gives semantics to SafeCap schemas. It is thus able to report a sequence of step (discrete train movements, point switching, route state changes, etc.) that leads to a collision or derailment. In most cases, such a sequence may be visually replayed to help a user to debug the schema.

Below is an excerpt from the B Method specification of two-aspect signalling of the double lead junction scenario.

```
MACHINE safety
SETS
 POINT;
 SIGNAL;
 TRAIN;
 AMBIT={TBZ1, TBW, TAA, TBV, TCM, TDR, TCL, TCK, TAD, TCN, TAE, TAC, TAB,
TBX, TBY, TAG, TBZ, TAA1, TAE2, TDQ, TDQ1, TBW1, TBU1};
 ASPECT={RED,GREEN};
 LINE={A_F, A_B, D_C, E_C};
 ROUTE={A_N3, N3_N14, N14_N16, N16_N, N_N19, N19_F, N16_N8, N8_N1, N1_B,
D_N18, N18_N10, N10_N11, N11_N4, N4_N7, N7_N5, N5_C, E_N12, N12_N6,
N6_N20, N20_N7}
...
PROPERTIES
   /* @axm4 */ LineAll = {
 A_F |-> {A_N3, N3_N14, N14_N16, N16_N, N_N19, N19_F } ,
 A_B |-> {A_N3, N3_N14, N14_N16, N16_N8, N8_N1, N1_B } ,
 D_C |-> {D_N18, N18_N10, N10_N11, N11_N4, N4_N7, N7_N5, N5_C } ,
 E_C |-> {E_N12, N12_N6, N6_N20, N20_N7, N7_N5, N5_C }     }
...
OPERATIONS
 move_hd(tt) =
  PRE
     /* @grd1 */ tt : dom(hd)
   & /* @grd2 */ hd(tt) /= RouteLast(hdr(tt))
  THEN
      hd := hd <+ {tt |-> RouteNext(hdr(tt))(hd(tt))}     ||
      occ := occ <+ {RouteNext(hdr(tt))(hd(tt)) |-> tt}
  END;
  move_hd_collide(tt) =
   PRE
     /* @grd1 */ tt : dom(hd)
   & /* @grd2 */ hd(tt) /= RouteLast(hdr(tt))
   & /* @grd3 */ RouteNext(hdr(tt))(hd(tt)) : dom(occ)
  THEN
      collision := TRUE
  END;
...
```

Fig. 3. The B Model used for model-checking safety properties.

IV. Capacity Assessment

Depending upon objectives, the capacity assessment may be done as a calculation of a single value according to one of predefined formulae or by running a detailed simulation of train movements. We use a range of capacity metrics and are actively looking for new ones.

Theoretical line capacity. This criterion assesses the capability of a line (a path through a junction) to support a certain amount of traffic irrespective of signalling constraints and safety requirements. The following formula gives theoretical line capacity in trains per second:

$$TC_1(l) = \left(\sum_{r \in routes(l)} \sum_{a \in ambits(r)} \frac{len(a)}{sl(a)}\right)^{-1}$$

where *len(s)* and *sl(s)* are the length, in meters, and the maximum allowed speed, in meters per second, of ambit *a*. The figure produced by *TC(l)* is always much higher than attainable in practice and is generally not useful even as a first approximation. It is sensitive only to track length and static speed limits. For the scenario in Fig. 2 we have TC1(AB) = 0.014 and TC1(AE) = 0.015 which gives 50 (AB) and 55 (AE) trains per hour.

An alternative calculation considers the kind of traffic travelling through a junction and assesses its capability to traverse junction lines. It calculates line capacity assuming the worst mixture of traffic – interleaving of fastest and slowest trains.

$$TC_2(l) = \left(max_{i,j} \frac{min(V_i^{max}, V_j^{max})}{2} \left(\frac{1}{d_i} - \frac{1}{d_j}\right)\right)^{-1}$$

where *i* and *j* are some two dissimilar train kinds travelling over line *l;* V_i^{max} and V_j^{max} are the train maximum speeds*;* d_i and d_j are the maximum decelerations of trains i and j. The calculation for the double lead junction gives TC2(AB) = TC2(AE) = 0.009 that correspond to approximately one train every two minutes.

Critical section. One obvious weakness of the theoretical capacity method is its inability to capture the notion of shared or intersecting track and hence the interference of traffic on crossing lines that leads to decreased capacity. It is thus tempting to localise the source of interference - identify the so called *critical section* of a network - and try to measure its contribution to capacity loss. One approach is to calculate the maximum time a train on a given line occupies the critical section. The shorter such time is, the less is the interference of a chosen line with other intersecting lines. Taking a weighted sum of occupation times of the crossing lines gives a figure that might be expected to correlate with the capacity lost due to line interference. Weights are selected to reflect relative traffic frequency on the lines.

The main challenge consists in identifying the location and the boundaries of a critical section. Differing critical sections would not only give different absolute figures of capacity but may also demonstrate different sensitivity to alterations in topology and signalling. It also matters whether the occupation time may is estimated following the crude theoretical capacity approach or is based on a more refined technique taking into the account signalling rules, point switching and train inertia.

For the example in Fig. 2, a sensible choice of critical section is the pair of ambits TAD and TAE including points P101 and P103. Using the traffic service pattern from '*Satisfaction of schedule'* method below*,* we can estimate the maximum occupation times for lines AB and EC to be 34 and 22 seconds. Note that nominally faster line AB has a higher occupation time. Although an AB train takes the normal branches of P101 and P103, it happens that one AB train has to accelerate from a complete stop to cover the length of the critical section.

Wasted track capacity. One could take a differing viewpoint and try to assess how efficiently the existing signalling makes use of the available track. Assuming a certain traffic pattern, one measures the minimum amount of free track observed during a the traffic scenario. The principle idea is that line interference and inefficient signalling tend to increase train headways. Reducing interference and improving signalling would often result in tighter headways, i.e., more track being occupied by trains. This criterion is unbalanced: it sharply favours slower train speed that automatically give shorter headways. For the same reason, it is sensitive to ambit sizes and the number of signalling aspects. At the moment, our tool does not measure wasted track capacity.

Cumulative travelled distance. With this criterion one measures the total distance travelled by all the trains that have entered the junction. The measurement is done for a set period of time and starts after a certain delay in an attempt mitigate the effect of the initial absence of traffic. The guiding intuition here is that a more efficient layout and signalling favour a balance between higher average speed (more distance covered by an individual train) and less wasted capacity (more trains concurrently running on a schema track). The cumulative travelled distance is computed as follows

$$CTD = \sum_{i \in TR} \int_{t_0}^{t_1} v_i(t)\, dt$$

where $t_0 - t_1$ is observation windows, *TR* is the set of trains and v_i(t) is speed of train *i* at time *t*. Alternatively, one can take $t_0=0$ and $t_1=t_{max}$ and consider the overall average speed $CDT/(t_1 - t_0)$. The figure calculated by the expression does not have any particular meaning and is used to rank differing versions of the same schema.

Satisfaction of schedule. If there is a detailed specification of a desired traffic pattern through a junction then one may take the ability to satisfy the pattern as a (binary) measure of capacity. A service pattern defines train kinds and the times trains appear at boundary nodes of a schema during some extent of time known as pattern duration. It is checked whether all the trains detailed by the pattern enter and leave the schema within the duration of the pattern.

The following is a sample definition of service pattern. The first part is specification trains and some of their characteristics, pertinent to train performance.

Name	**Speed, max**	**Acceleration, max**	**Deceleration, max**	**Length**
EC	40	0.5	0.4	240
FL	22	0.1	0.08	300

The second part specifies the pattern duration and the times trains appear on the boundary nodes of the lines of a schema.

Duration		15:00
Line	Train	Time
A_B	EC	0:00
A_B	EC	2:00
E_C	FL	0:00
E_C	FL	3:00

In this pattern, two fast trains (EC) suffer interference from two slow trains (FL). A simulation shows that the scenario show in Fig. 2 does not satisfy the schedule. In fact it takes more than 20 minutes for the last train to leave the junction.

The SafeCap Platform checks a schedule satisfaction by running a timed simulation of the hybrid model. A static check of the discrete model part would have already ensured that the model is safe - a simulation is only used to accurately assess the capacity of a schema.

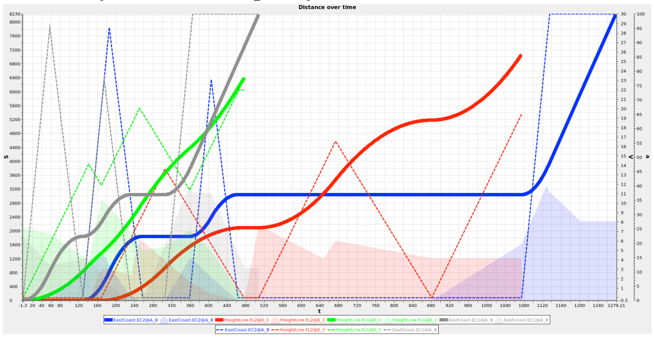


Fig. 4. A timed simulation of the double lead junction scenario.

The plot in Fig. 4 gives a history of train movement along the schema (solid lines depict cumulative distance, x-axis is time, dotted lines show speed and filled areas near the bottom are the current movement authority of a train). It shows that the second of EC trains (blue line) is blocked for a considerable time. This schema is equipped with two aspect signalling.

V. Capacity Improvement

SafeCap offers two instruments for capacity improvement: a library of user-defined patterns transforming topology graphs and signalling; and an automated search procedure that attempts to find a list of changes that lead to a desired capacity improvement. In this paper we only discuss the former.

Patterns are written in the Epsilon transformation language - a specialised notation for writing model to model transformations. A pattern is also an executable program: its instantiation is automatic.

Pattern 1. One drastic measure to improve capacity could be a change in topology avoiding the intersection of lines A_B and E_C. For this, we instantiate in succession two patterns: the first one merges points P101 and P103 to create a diamond crossing D101-103. The second pattern replaces the diamond crossing with a bridge. The bridge carries the E_C traffic avoiding any interference with the A_B traffic. As a side effect, the previously available connection between boundary nodes A and F has now disappeared (see Fig. 2).

As expected, this change does improve the schema capacity although different estimation methods report differing results (cf. Fig. 4 and Fig. 6). Theoretical measures show no change in capacity apart from those accounted for by change in track length.

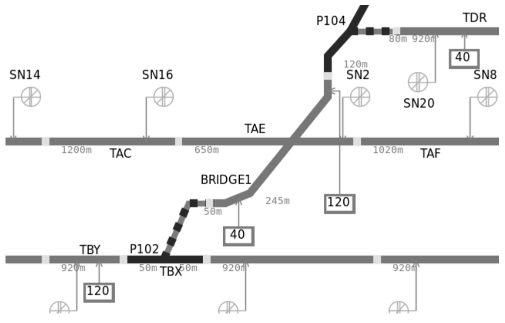


Fig. 5. Modified double lead junction scenario.

The critical section method demonstrates a sharp decrease in critical section occupation time: it is zero for line EC and much lower for AB as all the trains now go through the section at a higher speed.

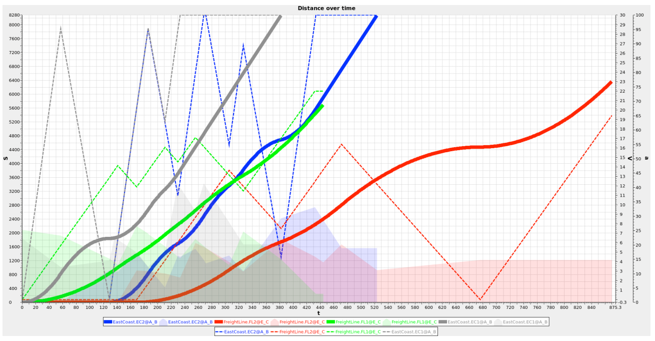


Fig. 6. Double lead junction scenario with bridge.

Wasted capacity method reports very slight improvement – just under 3%. Travelled distance method gives a sensible 32% improvement. We have found this technique to be the most stable indicator of capacity increase.

Pattern 2. This pattern automatically upgrades all signalling rules to 4 aspect signalling while route boundaries remain unchanged. Somewhat surprisingly, this pattern delivers almost as much capacity improvement (according to the travelled distance metric) as the first pattern (although the bridge with 4 aspect signalling is still much better).

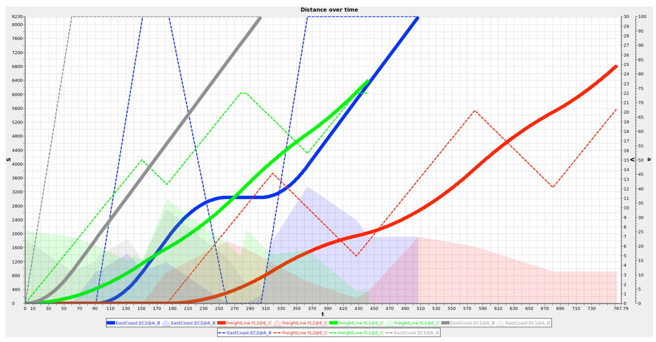


Fig. 7. Double lead junction scenario with 4 aspect signalling.

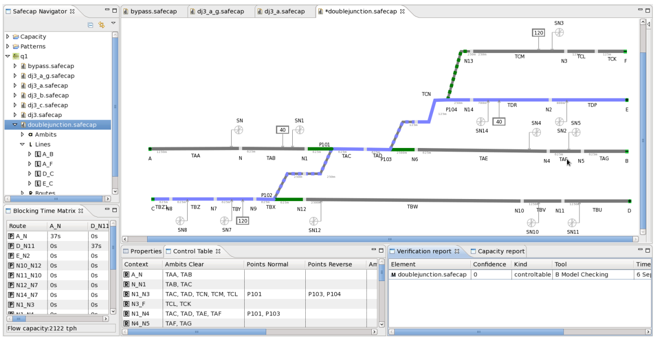
This little example attempts to illustrate that it is often worthwhile to explore different directions of capacity improvement and then separately assess their practicality. The following table summarises the capacity assessment exercise for the two schema changes.

Method	**Bridge**	**Signalling upgrade**
Theoretical, 1	-2%	0%
Theoretical, 2	0%	0%

Critical section	230%	-170%
Wasted capacity	?	?
Travelled distance	32%	24%
Schedule	unsatisfied → satisfied	unsatisfied → satisfied

VI. SafeCap Toolset

The SafeCap DSL is the central component of our modelling framework - the SafeCap Platform [4]. The purpose of the Platform is to enable railway engineers to analyse complex junctions by experimenting with signalling rules, signalling principles, track topology, safety limits (e.g., speed limits for points and crossings) while receiving an on-line feedback from automated verification and analysis tools. The overall motivation in the development of the Platform is to offer a range of techniques for assessing and improving the ability of layouts and signalling rules to efficiently accommodate railway traffic.



Blocking Time Matrix

Route	A_N	D_N11
A_N	37s	0s
D_N11	0s	37s
E_N2	0s	0s
N10_N12	0s	0s
N11_N10	0s	0s
N12_N7	0s	0s
N14_N7	0s	0s
N1_N3	0s	0s

Flow capacity:2122 tph

Properties | Control Table

Context	Ambits Clear	Points Normal	Points Reverse	Am
A_N	TAA, TAB			
N_N1	TAB, TAC			
N1_N3	TAC, TAD, TCN, TCM, TCL	P101	P103, P104	
N3_F	TCL, TCK			
N1_N4	TAC, TAD, TAE, TAF	P101, P103		
N4_N5	TAF, TAG			

Verification report | Capacity report

Element	Confidence	Kind	Tool	Time
doublejunction.safecap	0	controltable	B Model Checking	6 Se

Fig. 8. The Safecap Platform screenshot.

As an implementation platform, we have decided to use the Eclipse Integrated Environment and Eclipse Modelling framework (EMF) to realise our DSL. One important consideration was the ability to benefit from the extensive EMF ecosystem which offers a tool-kit for model manipulation and the construction of graphical and textual editing tools. A screen shot of the running platform is given in Fig. 8.

VII. Conclusion

This work is conducted as part of a SafeCap research project (safecap.cs.ncl.ac.uk, 2011-2013) sponsored by RSSB and EPSRC UK. The partners of the project are Newcastle University (UK) - Coordinator, Swansea University (UK), Invensys Rail and AIST (Japan). The overall aim of the project is to develop modelling techniques and tools for improving railway capacity and ensuring safety standards. To achieve this aim the project team is working on developing proof-based reasoning about time in state-based models, providing an open tool support for verifying timed systems, developing an intuitive graphical domain-specific language for the railway domain with a tailored tool support and on identifying and validating design patterns for improving railway capacity by altering route design, track layout, signalling principles and driving rules.

One decision we have made early in the project was to develop a railway domain specific language (DSL) to allow the engineers to operate in terms of their domain and to hide the formal verification from them as much as possible. The definition of this DSL is in the core of our method. We have developed a compact and, in our opinion, simple domain language for describing railway topologies and signalling rules. The language is strictly defined making it possible to benefit from widely available verification tools. In the scope of this paper we were not able to discuss train movement rules that constitute an important part of the domain language semantics.

Our plans are to further develop the Platform, the SafeCap DSL and the methods of analysing and improving capacity. Route-based signalling discussed in this paper is only one the several semantics that we plan to support in the Platform. An interesting challenge, from the viewpoint of capacity assessment, is modelling the dynamic definition of ambit boundaries that adapt to train progress and speed.

The railway domain has always been one of the areas in which formal methods are successfully deployed and used in a substantial way. For example, in France RATP (a major rail operator) with a considerable experience in formal methods, looks favourably on using formal methods to conform to the development standards that they require. From the mid 90s, in France, RATP, the main rail operator with considerable experience of formal methods, has been approving various developments that use the B method as meeting the development standards RATP require [2]. There are now several tool development companies supporting the use of the B [6] and Event-B [3] methods in the railway domain: ClearSy, Systerel, and Formal Mind. In addition to the B method model-checking has been successfully used by various railway companies working together with universities (see, for example, [7,8]).

Acknowledgement

This research is supported by the EPSRC/RSSB UK SafeCap project. We are grateful to Simon Chadwick, Markus Roggenbach and Dominic Taylor for their feedback on the earlier versions of this work.

References

[1] SafeCap Project website, 2012, available at safecap.cs.ncl.ac.uk.
[2] D. Essame and D. Doll , “B in Large-Scale Projects: The Canarsie Line CBTC Experience.”, ser. Lecture Notes in Computer Science, J. Julliand and O. Kouchnarenko, Eds., vol. 4355. Springer, 2007.
[3] J.-R. Abrial, Modelling in Event-B. Cambridge University Press, 2010.
[4] SafeCap Platform website, 2012, available at sf.net/projects/safecap.
[5] D. Bjørner, “Formal Software Techniques in Railway Systems.” 2000.
[6] J.-R. Abrial, The B-Book. Cambridge University Press, 1996.
[7] M. Leuschel, J. Falampin, F. Fritz, and D. Plagge, “Automated property verification for large scale B models with ProB.” Formal Asp. Comput., vol. 23, no. 6, pp. 683–709, 2011.
[8] A. Ferrari, G. Magnani, D. Grasso, and A. Fantechi, “Model checking interlocking control tables.” in FORMS/FORMAT, E. Schnieder and G. Tarnai, Eds. Springer, 2010, pp. 107–115.
[9] P. Behm, P. Benoit, A. Faivre, and J.-M. Meynadier, “Meteor: A Successful Application of B in a Large Project”. FM ’99. Springer-Verlag, 1999, pp. 369–387.
[10] A. Iliasov and A. Romanovsky, “SafeCap domain language for reasoning about safety and capacity.” CS-TR-1352, Newcastle University, 2012.