RTCSIM: an innovative, extendable computation engine for timetable validation

T. Polzin & R. Gooßmann
HaCon GmbH, Hanover, Germany

Abstract

In the beginning of 2005, HaCon started an initiative to redesign the computation engine for running times and mapping of train service dynamics as provided within the *TPS Timetable Planning System,* which is the core production system for railway operational planning in Denmark. In the light of increasing demands on timetable validation technology, this activity aimed at providing a future-safe solution for the integration of such technology into operational planning applications.

Meanwhile, the first versions of the newly designed module *RTCSIM* have been integrated within the actual software version of *TPS*. The system is used for integrated production planning over different time horizons including STP (Short Term Planning) at *Banedanmark*, *DSB* and *DSB S-train* as well as *Trafikstyrelsen*, the Danish regulation and certification body. It was implemented within the years 1999–2002 and has evolved steadily until today, facing imminent changes in business processes and system interfaces.

TPS features a high resolution level of infrastructure data including tracks and routes as well as detailed security system information laid out in a principle track plan. Train services are mapped into the model using detailed occupation and release information for the infrastructure elements following the UIC 406 leaflet. It includes conflict detection and resolution algorithms being applied to the overall capacity plan maintained by the system, featuring train services, temporary speed restrictions, track blockages or other withdrawals of network capacity. Using the modular architecture of the new computation engine will unleash several innovative potentials. This especially facilitates the future use of extensive functionality for assessment and optimisation of railway network capacity within *TPS*. With that, HaCon now continues with a long tradition of application development such as the well known *UX-SIMU* software package as supplied until 1999.

This paper gives an overview about the underlying concept and mechanisms of the new computation engine *RTCSIM* as well as an assessment of impacts on future timetable validation techniques.
Keywords: timetable planning, operational planning, timetable validation, microscopic infrastructure, simulation, capacity assessment.

1 Introduction

Recent assessments about using stochastic, synchronous timetable simulation as validation method within the annual timetable development process made clear that availability of complete and effective plans is a key issue (see e.g., Watson [8] and Weits [9]). Consequently, these validation methods must feature a high measure of integration into operational planning systems being used for management of the accurate and up-to-date "master plan". Otherwise, loss of acceptance and thus, loss of timetable quality would be the result, especially when it comes to short term planning (STP) where the amount of daily service variance is high.

This paper outlines the process of implementation of the *RTCSIM* computation engine for timetable validation purposes, its underlying concepts and impacts on future timetable validation techniques.

The paper is structured as follows: In Section 2, we refer to the general business background of timetable planning in order to depict the need for a fast and reliable validation technique as part of the daily operational planning. Section 3 describes the implementation process of *RTCSIM*, starting with an overview of the goals related to this development initiative. Section 4 includes a more detailed depiction of software concepts and architecture including integration options of *RTCSIM*. In Section 5, we discuss the current status of the project followed by an introduction to two major innovative concepts of the *RTCSIM* module. Section 6 provides our brief conclusions.

2 Background

While pursuing their aim to deliver timetables of good quality, planners have to face several business process challenges, such as:

- The need for fast decisions on TOC (Train Operating Company) requests;
- Existing timetable regularity requirements and commercial impacts (e.g., performance guarantees);
- Increasing network capacity usage especially in station areas;
- The need for short term changes;
- The planning and operational rules to be obeyed;
- The train service request compliance; and
- Assumptions and existing knowledge about reality;

Dealing with these issues means dealing with plans that make use of limited resources and at the same time, ensure that these plans are feasible.

More practically, such plans feature not only realistic timetables but also the ability to overcome potential obstacles in operation as fast as possible. This ability is also referred to as the *timetable quality.* Providing high quality timetables while fulfilling the business process requirements as mentioned above makes it obvious that there is an immanent need for a fast and reliable decision support for immediately assessing the quality of changed or amended timetables as part of the daily planning tasks.

The ***TPS Timetable Planning System*** is the core production system for railway operational planning in Denmark, being in production under the name "*STRAX*" since 2002 (see also [2] for details). It makes use of a high resolution geography information including railway infrastructure on signal berth level and different signalling systems with their intrinsic properties. Train services are mapped into the model using detailed occupation and release information for the infrastructure elements following the UIC 406 leaflet.

The system includes an integrated kernel module for running time calculation and animation of train services of the maintained timetables which is based on the well-known *UX-SIMU* tool as supplied by HaCon until 1999. This tool and it's synchronous timetable simulation was used at several railway customers including the German railways through-out the 1990's for evaluation of railway infrastructure, timetables and operation.

2.1 RTCSIM

RTCSIM is designed as a separate computational module for timetable validation. It provides the following services:

- Runtime Calculation (RTC): The computation of one train running separately, using detailed information about trains and tracks; includes station track adjustment, trackway search, and the calculation of signalling system dependant occupation and conflict information.
- Synchronous Simulation (SIM): In this context, simulation means the representation of interacting train runs on a given infrastructure at a given time interval. Typically it is run under perturbed conditions (e.g., stochastically introduced delays or defects), such that signalling systems' safety rules, dispatching rules (e.g., prioritisation, unscheduled overtaking, re-routing), and functionality to avoid unrealistic situations (in particular, deadlocks) are applied. *RTCSIM* provides simulations in a special form of *Simulation Event Protocols (SEP)*, which will be described in Section 4.4 together with basic analytical facilities that can be used by the visualisation front end.
- Additional Services: E.g., for basic queries of driving dynamics, such as for the "maximum permissible load" of a train run.

3 Implementation process

3.1 Goals of the *RTCSIM* development initiative

The main goals for the development have been the following:

- *Compatibility with the existing UX-SIMU kernel module*
  As the existing *UX-SIMU* based kernel module has a profound empirical basis and the *TPS* customers maintain a huge amount of data within their *TPS* systems, it has been the aim to provide compatibility to the existing kernel module concerning the calculation results as well as regarding the usability of available *TPS* data.

- *Extensibility*
  In particular, for supporting new signalling systems (see Section 4.3) and evolving dispatching functionality (see Section 5.1).
- *Maintainability*
  As *RTCSIM* will be used in a commercial product with long lifetime, it is a high priority goal to have modules that are well structured and comprehensible in the beginning and also maintain their structure after changes and additions.
- *Quality Assurance*
  Being a part of the timetable production system at DSB and Banedanmark, the requirements for quality assurance (fault tolerance, quality of results, stability of product version releases) needed to be fulfilled.

3.2 Setting up the project

The *RTCSIM* project has been started in the beginning of 2005. The project schedule included milestones for delivery of different sub-modules in different stages as well as intensive testing and benchmarking activities. Emerging from a long history of providing planning systems and simulation tools for the railway industry, the in-house knowledge about relevant needs and methods formed the base for the set-up of requirements. Moreover, recent results especially from the fields of operational railway science and software engineering were incorporated into the specification documents used for the *RTCSIM* development. The development process was characterised by tight contact to railway planners, scientists and users of simulation models throughout the complete realisation period.

3.3 Involved methods and railway science

The process used an in depth analysis of the old *UX-SIMU* rooted computation in *TPS*. We evaluated the operational railway science background and re-thought the mathematical formulas for the driving dynamics. Additional input from conferences and journals was taken into account.

4 Software design concepts

In this section, we describe the main software design concepts that have been applied to achieve the goals as mentioned in Section 3.1.

4.1 Unification

The *UX-SIMU* based kernel module had multiple routines for similar things, e.g., the computation of one train on its own and of multiple trains at once used different concepts for representing the security system aspects. The new *RTCSIM* module has just one routine for one thing. For example, it adopts the paradigm "a runtime calculation is the simulation of one train running on its own". This leads to significantly improved maintainability and quality assurance.

4.2 Quality assurance

For *RTCSIM* we use different automatic testing procedures that are performed and checked every night.

- Unit Tests: Test for the basic functional units, tested on their own, typically on restricted test data, immediately detecting if a new development breaks already implemented functionality. This encourages developers to refactor parts of the program (see [1]).
- Integration Tests: Tests for the integrated functionality, comparing just the over all results.
- Regression Tests: Computer programs tend to reproduce bugs that have been fixed. Hence, test cases that showed the previously buggy behaviour have some indication to represent critical cases. Therefore, we implemented a system that checks these possibly critical cases.
- Performance Tests: Automatic performance tests give an immediate feedback on the performance implication of program changes.

Additionally, we use a report on code coverage showing which parts of the code are not covered by any test as well as a memory analysis report based on *valgrind* [6].

4.3 Modularization

In the programming of the existing UX-SIMU based computation engine the aspects dispatching, driving dynamics, and security were interwoven. In *RTCSIM* these aspects are handled by separate modules.

This improves

- extensibility (because it is easier to replace one aspect of functionality by something else);
- maintainability (because the code has a clearer structure); and
- overall system quality (because the separate modules can be tested on their own more rigorously [3]).

4.3.1 Example: driving dynamics encapsulation

As an example, we outline the successful encapsulation of driving dynamics aspects. Here, the major invention is the development of the module interfaces. For flow of information into the driving dynamics module we use standardised *speed restrictions*, each of them containing information on when the speed restriction is transmitted to the train, and on which region it applies. For the flow of information back from the driving dynamics we use *driving triggers*. These triggers can be thought of virtual balises that perform some action as the train passes them (e.g., blocking some part of the infrastructure).

4.4 Introduction of Simulation Event Protocols (*SEP*)

Another major modularization step is the invention of Simulation Event Protocols (*SEP*). These protocols cover all information that is necessary to represent a given point in time as part of the simulation time frame. Improvements through *SEP*s are:

- Quality Assurance: *SEP* is a storable, hence manageable document with long term reproducibility, independent of forthcoming *RTCSIM* versions. It includes quality assurance information such as, e.g., creator, creation timestamp, source project etc.;

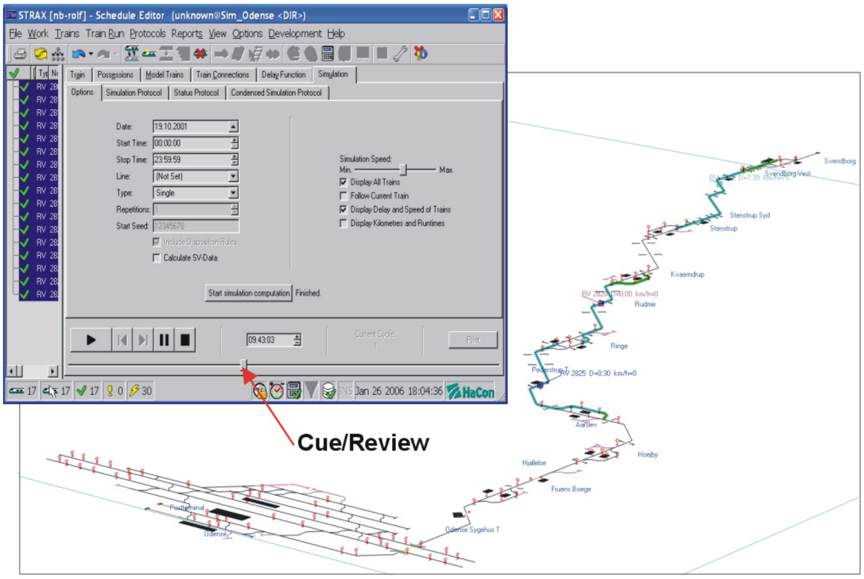
STRAX [nb-rolf] - Schedule Editor (unknown@Sim_Odense <DIR>)
File Work Trains Train Run Protocols Reports View Options Development Help
Train Possessions Model Trains Train Connections Delay Function Simulation
Options Simulation Protocol Status Protocol Condensed Simulation Protocol
Date: 19.10.2001
Start Time: 00:00:00
Stop Time: 23:59:59
Line: (Not Set)
Type: Single
Repetitions: 1
Start Seed: 12345678
Include Disposition Rules
Calculate SV-Data
Simulation Speed: Min. Max.
Display All Trains
Follow Current Train
Display Delay and Speed of Trains
Display Kilometres and Runtimes
Start simulation computation
Finished.
09:43:03
Current Cycle
Print
Jan 26 2006 18:04:36
HaCon
Cue/Review
Svendborg
Svendborg Vest
Stenstrup Syd
Stenstrup
Kvaerndrup
Rudme
Ringe
RV 2825 D=0:30 km/h=0
Aarslev
Hoejby
Hjallelse
Fruens Boege
Odense Sygehus T
Odense

Figure 1: RTCSIM Simulation with reviewing feature.

- Improved Analysis: Critical situations can be examined over and over, using a tape recorder like cue/review functionality (going backwards in time); see also figure 1. A critical situation may be recorded as movie to show it even without a *TPS* application in the background;
- Keyfigure Tracking: Not only train animation can be seen repetitively but also the changes of absolute or accumulated keyfigures such as number of hindrances, sum of delays, capacity consumption etc.;
- Communication: *SEP*s can be sent easily via e-mail;
- More effective utilisation of computational power:
  - No need to re-compute existing computations;
  - Task scheduling, i.e., distribution of simulation jobs on multiple processors (scalability);
- Software Design Aspects:
  - Thoroughgoing separation of GUI-visualisation and simulation;
  - *SEP* files are available in binary, structured text or XML format;

4.5 Integration options

As the *RTCSIM* software is designed to be a separate module with a well defined interface on binary/ XML basis, there are several integration possibilities.

- *RTCSIM* can be used inside the *TPS* GUI or as command line interpreter in foreground or background mode. Each integration makes use of a *TPS-RTCSIM*-interface module that connects *RTCSIM* with *TPS* data.
- Just replacing this interface module, *RTCSIM* could be connected to other applications' data or work inside another application, if the data model of the other application maps somehow to *RTCSIM*'s needs for computation.

5 Current state and perspectives

Meanwhile, the main goals as mentioned in Section 3.1 have been reached and the first versions of the newly designed module *RTCSIM* has been integrated within the actual software version of *TPS*.

Concerning the computation speed, *RTCSIM* outperforms it's predecessor module by a factor of 2 to 3. The clean architecture of *RTCSIM* offers many potentials for optimisation. Realising these potentials is a currently ongoing process, therefore it is not sensible to give quantified performance tables here.
In the following, we introduce two major innovative concepts of RTCSIM.

5.1 The concept of control laws

As the purpose of stochastic simulation is to deal with perturbed operations condition, the near realistic handling of these perturbations (i.e., by dispatching) is an important goal for simulation routines. The dispatching must be powerful, so that it can handle the situation appropriately, but also not better than reality, because otherwise scenarios that have even been simulated successfully could be impossible to handle in real life situation. From practical experience and reports in the literature [7] with simulation algorithms we derive another major requirement: *The dispatching decisions as being applied by the program must be presented to the user in a reasonable and understandable way.*

The basic approach is that there is a larger number of concrete Control Laws that are applied for concrete dispatching decisions, but they are not generated by hand, but by *Meta Control Laws* that specify the desired dispatching policies (like prioritisation) on a higher level. Making use of technically challenging algorithms, *Meta Control Laws* produce tables of Control Laws that can be reviewed, checked, corrected, or incremented manually. This significantly improves the comprehensibility of results compared to a "black box" dispatching algorithm.

5.1.1 Control laws

A *Control Law* in our context is a *decision rule which describes an operational constraint or behaviour for prevention or treatment of operational conflicts.*

It has the following basic structure:
If (OpCondition[Filter]) Then (OpAction[Filter]), where "OpCondition" describes an operational scenario involving one or more train services. "OpAction" is either "Insert (e.g., register a facultative train service)", "Delete (cancellation of a train service)" or "Change (e.g., waiting time)" plan elements. OpCondition and OpAction may both be related to filters, restricting their validity in

- geography (e.g., region, line, station, operational point);
- time (e.g., operating days, daytime interval); and
- operation (e.g., train category, train operating company, delay, passenger occupancy degree).

If no time is specified for the OpAction filter, the OpAction is valid as long as the OpCondition is matched. This is especially useful when formulating rules for deadlock detection and prevention as suggested, e.g., in [4] and [5]. OpCondition and OpAction may also include threshold values and parameters as well as logic terms "and", "or" and "not". Another option is the use of projected track occupation information in order to derive forthcoming (knock-on) delays and conflicts, i.e., conditions such as "If (... and +delay at next stop > m minutes...)".

Control Law example:
If (a regional passenger train A is delayed more than n minutes between station X and Y and a IC train B running behind it is hindered by A in off-peak hours) Then (A shall have an extra stop in track t at station Y in order to let B pass).

In a similar way, all other typical train prioritisation rules can be formulated.

5.1.2 Meta Control Laws (*MCL*)

Since OpConditions and OpActions may be formulated using different levels of abstraction, it is useful to distinguish between Control Laws that can be directly applied to existing data entities and more general rules, which need entity instantiation and extension and by that, producing itself a set of concrete Control Laws. Such general rules we call *Meta Control Laws (MCL).*

Meta Control Law example:
If (delayed regional passenger traffic causes hindrances to long distance traffic in off-peak hours) Then (overtaking should take place).

It is intended to make the use of Meta Control Laws as flexible as possible since this minimises the amount of manual work and the high abstraction level of *MCL*s ensures long term consistency disregarding detailed changes of plan, infrastructure or other planning data over different timetable periods.

Since by definition, the scope of *MCL* validity is not limited it is evident to use this new concept for official access and regulation rules as well as for complete network strategies. Moreover, such rules and strategies are easy to change in order to derive performance keyfigures for different scenarios.

5.1.3 Open API for dispatching plug-in

Another approach for powerful, predictable, comprehensible dispatching decisions is the supply of a C++ program interface (API) for dispatching: Our concept includes the provision of information services (based on the current simulation situation and the planned schedule) to an external software plug-in that decides on dispatching questions like unscheduled overtaking. Alternatively, a user-editable script language (LUA, Python) can be used for this purpose, enabling rapid configuration of dispatching tasks.

5.2 The restarting mechanism

As the simulation event protocol (*SEP*) contains all information to display a simulation at a certain point in time, it also contains the information to *restart* a simulation at a certain point in time, possibly with modified parameters. This enables us to look at possible applications:

5.2.1 Interactive refinement of dispatching

It is easier to tune the dispatching algorithm, if one can look at a certain situation, where something went wrong. The user could manipulate the Control Laws and restart the simulation at this situation. This cycle can be repeated quickly until the desired behaviour is achieved.

5.2.2 Automatic refinement of dispatching

In case of deadlocks or other undesired dispatching effects, the simulation could use backtracking to find a feasible solution.

5.2.3 Real time dispatching

If there is a possibility to receive (partial) information about the current train positions, these information can be merged into the simulation at this point in time. Together with the possibility to adjust dispatching decisions interactively (see Section 5.2.1) this would lead to a powerful support tool for real time dispatching.

6 Conclusions

We described the background, the main goals and some of the underlying concepts for successfully building *RTCSIM* as an extendable computational engine for timetable validation. We believe that the implementation of these concepts was very successful, leading to a future-safe basis for further development, for which we gave a short overview.

One of the most important innovations of *RTCSIM* lies in the ability of applying time reversion in simulation visualisation and re-entry into retrospective simulation events (backtracking). This ability can be used for pre-emptive dispatching rules taking prospective delay or conflict situations into account.

Our approach includes the provision of a Control Law plug-in for *RTCSIM,* which represents an open interface for external experts for defining and testing dispatching procedures on their own.

We introduced Meta Control Laws (*MCL*) as a major concept for mapping of high level rules for regulation and network strategy. *MCL*s are supposed to serve as a major planning instrument of forthcoming simulation applications.

The XML based *RTCSIM* communication interface facilitates easy integration in other operational planning system environments.

References

[1] Beck, K., “Test Driven Development”, *Addison-Wesley*, 2002, Boston.

[2] Kaas, A. H. & Gooßmann, R., “Implementation of the Timetable Planning System STRAX/TPS in Denmark”, *COMPRAIL 2004*, Dresden.

[3] Lakos, J., “Large Scale C++ Software Design”, *Addison-Wesley*, July 1996, Boston.

[4] Pachl, J., “Safe disposition and scheduling in railway operation”, *Signal + Draht* (92) 5/200, p. 38-41.

[5] Pachl, J., “Steuerungslogik für Zuglenkanlagen zum Einsatz unter stochastischen Betriebsbedinungen”, *Institut für Verkehr, Eisenbahnwesen und Verkehrssicherung der TU Braunschweig*, Heft 49, 1994.

[6] *Valgrind*: www.valgrind.org.

[7] C.R. Warninghoff, C. Ferchland, “Nutzung von Simulationen zur Unterstützung der betrieblichen Infrastrukturplanung”, *ETR* no. 53 (2004), vol 7/8, p. 490-498.

[8] Watson, R., “Using stochastic simulation to predict timetable performance – status and developments in the UK”, *IAROR 2005*, Delft.

[9] Weits, E.A.G., Zinsmeister, E.R., “On the validity of simulation models of railway operations, a plea for standardised models”, *IAROR 2005*, Delft.