



AUTOMAIN

Augmented Usage of Track by Optimisation of Maintenance, Allocation and Inspection of Railway Networks

Project Reference: 265722
Research Area: SST.2010.5.2-1

Planning and scheduling tool for optimised maintenance

Deliverable Reference No: 5.1

Due Date of Deliverable: 31/01/2013
Actual Submission Date: 01/02/2013

Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	X
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document Summary Sheet

This document presents the scheduling and planning engine implemented in Task 5.1 of AUTOMAIN. The engine relies on the cooperation of three combinatorial optimization problems that addresses (1) long-term planning, (2) maintenance resource routing and task splitting into track possessions, (3) scheduling of track possessions and train path rescheduling. The constraints and objective functions are presented for each problem as well as the algorithmic approach to solve these models. The data interface to formalize the input is defined and a Man-Machine Interface is proposed to display the results.

Additional information	
Subproject / workpackage:	WP5
Authors:	Christoph LACKHOVE (DLR) Lisandro QUIROGA (TU-Braunschweig) François RAMOND (SNCF) Raja REBAÏ (EURODECISION) Renaud RIVOALLAN (EURODECISION) Lynda ROUSSEAU (EURODECISION) Francis SOURD (SNCF)
Document date:	31 Jan 2013
Status:	Final, Approved by consortium

History			
V0.1	14 Nov 12	Initialisation	Based on the following working documents - State of the art report - Problem definition document - Problem solution document
V0.2	23 Nov 12	First version after WP5 meeting	Structure of the document
V0.3	7 Dec 12	F. Sourd	State of the art
V0.4	14 Dec 12	L. Quiroga F. Ramond C. Lackhove	Additions: Description of LTP and TWG MMI
V0.5	21 Dec 12	Alpha	Sent to WP5 members
V0.6	27 Dec 12	F. Sourd	Correction of the previous version Published on the web site
V0.7	28 Dec 12	Beta	Contents completed Sent to WP5 members
V0.8	7 Jan 13	Revision of TWIP version	Sent to EURODECISION
V1.0	10 Jan 13	F. Sourd	Sent to the Project Management Team
V1.1	17 Jan 13	L. Quiroga	Addition : TWG tests
V1.2	24 Jan 13	C. Roberts	Proof-reading
V1.3	29 Jan 13	F. Ramond	Link with KPIs
V1.4	31 Jan 13	F. Sourd	Comments from DB.

Table of Contents

Introduction	2
Background	2
WP5 requirements and objectives	2
State of the art	4
General overview	7
Methodology	7
Architecture of the tool	7
Cooperative optimisation problems and algorithms	9
Long-term planning	9
First step: Building an initial plan based on an approximated model	11
Second step: Validating the plan	11
Third step: Optimising the plan	12
Time-window generation	12
Large problem decomposition	13
Maintenance machine routing	14
Time-window insertion	15
Input	15
Output	16
Constraints	16
Objectives	17
TWIP algorithm	17
Interfaces	17
User interface	18
Background	18
Description	18
Page 1: Overview-condition of network	19
Page 2: Detail-condition of section	20
Page 3: Choice of task and subtask	20
Page 4: Result of optimisation	21
Page 5: Conflicts	22
Page 6: Evaluation of result	24
Data interface	24
Conclusion	25
Tool implementation and preliminary tests	25
Future works	26
Appendix 1: References	27
Scientific literature	27
We sites and software	28
Reference documents	29
AUTOMAIN working documents	29
Appendix 2: Reference data	30
Network	30
Train Grid	31
Maintenance data	33
Appendix 3: Data exchange between modules	35
Data exchange between LTP and TWG	35

Data exchange between TWG and TWIP	36
Appendix 4: LTP sample run log	38
Appendix 5: TWG experimental results	40
Appendix 6: TWIP experimental results	43
Experimental conditions & results	43
Other experiments	44
Appendix 7: Train path description : XML sample	45
Appendix 7: XSD Schema	48

Executive Summary

This report describes the results of the work done in Task T5.1 within Work Package 5 (WP5) of AUTOMAIN. The overall goal of WP5 is to develop operations research methods and tools for autonomous maintenance planning and scheduling.

The problem must be seen in two complementary ways. Planning aims at an optimised assignment, over several months or years, of the maintenance operations required over the complete network so that the maintenance resources are well used in order to limit track possessions and optimise their moves between the work sites. Scheduling focuses on a problem with a shorter time-horizon in which most of the operating constraints must be modelled. In other words, planning is macroscopic whereas scheduling is microscopic.

However, scheduling must be based on what has been planned earlier and planning cannot ignore operational constraints. Therefore, an integrated (or systemic) model in which planning and scheduling constraints are modelled is the most promising way to derive the best maintenance schedules. Since such a model is large and complex, classical optimisation algorithms are inefficient. We then propose a cooperative approach in which three modules communicate and cooperate in order to build the schedule.

LTP (long-term planning) is in charge of scheduling all the maintenance operations, making sure that the individual planning of all machines is feasible and that general capacity constraints on the network are satisfied. TWG (time-window generation) is in charge of routing maintenance machines and splitting operations into track possessions (for example an operation can be divided into five time periods of 3 hours over 5 days). TWIP (time-window insertion problem) is in charge of solving localised assignments problem with accuracy and taking into account technical constraints (i.e. optimise the planning of a maintenance task along with its corresponding maintenance train path).

In order to be run, these problems require a lot of numerical values (durations, distances, speeds, costs...) An XML-based data interface is introduced in order to practically define the input for the problems. A proposal for a graphical user interface that displays the plans and schedules computed by the algorithms concludes the presentation.



Acronyms

DoW	Description of Work, a document last changed on 8 November 2011 and sent in during the application phase as Annex I to the consortium's proposal.
CR - FR	Capability requirement - Functional requirement
GUI	Graphical User Interface
KPI	Key Performance Indicator
IT	Information Technology
LTP	Long-term planning
MMI	Man-Machine Interface
OR	Operations Research
PDD	Problem Definition Document
POC	Proof of concept
PSD	Problem Solution Document
TWG	Time-Window Generation
TWIP	Time Window Insertion Problem
WP	Work Package

Glossary

Maintenance operation	Work load that must be executed over a short period in order to improve the status of the railways infrastructure.
Maintenance task	Elementary task that has a given duration (shorter than 24 hours). The duration can be a fixed number or a function that depends on parameters. The task typically requires a set of resources to be executed and it is submitted to a set of constraints (e.g. two tasks cannot be processed in the same time, a task must be executed after the completion of another task, a task must be preceded by a setup...) In most models, a task cannot be divided into subtasks.
Mode	Sequence of tasks that aims to realise a maintenance operation.
Planning	In this report, planning means scheduling maintenance operation over a long-term horizon of 1 to 5 years.
Time-window	A time-window represents the infrastructure capacity between two points of the network over a time-period. In this work, time-window corresponds to either a train-path of a maintenance machine or a track possession for maintenance works.



Introduction

Background

The core objective of the AUTOMAIN project is to improve the efficiency of track maintenance to increase the availability of the network for freight traffic (a full description of the project and its objectives is contained in the introduction to Work Package 1, Deliverable 1.1). The proposed time horizon for widespread implementation of the proposed changes is in a number of stages, the first of which is 2026 (i.e. fourteen years from now).

The overall goal of WP5 is to develop operations research (OR) methods and tools for autonomous maintenance planning and scheduling. WP5 **does not address** the important problem of computing the target dates to execute maintenance operations. This problem is also often referred to as a *planning problem*, but is based on degradation models that are deeply different from OR scheduling algorithms. We refer the reader interested in this topic to the deliverable of the recently completed European project INNOTRACK (2010). In our work, these target dates are supposed to be computed and given in the input of our algorithm. Hence, planning is to be understood as “long-term scheduling”, the goal of which is to derive feasible and optimised schedules with respect to operational constraints.

WP5 has two milestones, namely MS7 and MS8, related to the delivery of the tool. The first milestone, which corresponds to this Delivery D5.1, means that the prototype is ready to be tested on real data. It will be tested on scenarios built on the inputs from WP3 and WP4, which is the goal of WP5 of the second milestone (MS8). After this milestone, the results will be analysed in order to produce D5.2 and D1.3.

The present work relies on preceding tasks in the AUTOMAIN project. The requirements of the tool are derived from the work done in WP1 and especially the answers to the questionnaires (D1.1) which lead to Capability Requirements (CR) and Functional Requirements (FR). Section 4 of D1.2 lists the most relevant scheduling problems that must be addressed for track maintenance. The three problems addressed in this report are directly derived from this previous work. Finally, let us mention that the conclusion of D2.2, devoted to the lean analysis of the track maintenance process, points out that efficient scheduling is one of the main key issues for efficient track maintenance.

WP5 requirements and objectives

D1.2 refines the objectives of WP5 by specifying the following capability requirements:

- package maintenance work
- reference all track condition data/faults against a network map
- access/egress work sites rapidly
- carry out reactive maintenance rapidly

- achieve very low frequencies for planned maintenance
- optimise scheduled activities automatically

A complementary document indicates that the quantitative objective of the optimised and automated planning and scheduling tool is to decrease track possession time by 8% (Luiten, 2012).

Given a maintenance plan, computing the associated overall possession time is not as straightforward as it might look. Indeed, a simple sum of the durations of all possessions for all operations does not take into account the space aspect. For instance, if two operations are being performed at the same moment at the same place, or close to each other, with the same duration, we could as well consider them as a single operation : this would divide the computed possession time by a factor 2 (see Figure 1). Hence, depending on how operations are grouped and where they are located, the assessment of the possession time must differ.

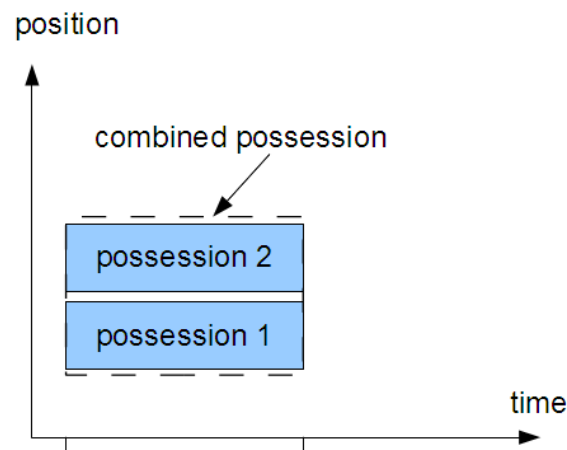


Figure 1 - Possession time computation (1)

One way to deal with this aspect is to ponder the possession time by the distance covered. In such an approach, the capacity used by maintenance operations would be represented as a surface on the time-distance diagram. Here again, this might be insufficient to represent reality. For instance, if we have the situation represented in Figure 2, no train path can fit between the possessions. So, from the network usability perspective, these three possessions are equivalent to the combined possession drawn with a dashed line, whose surface is significantly higher than the sum of individual possession surfaces.

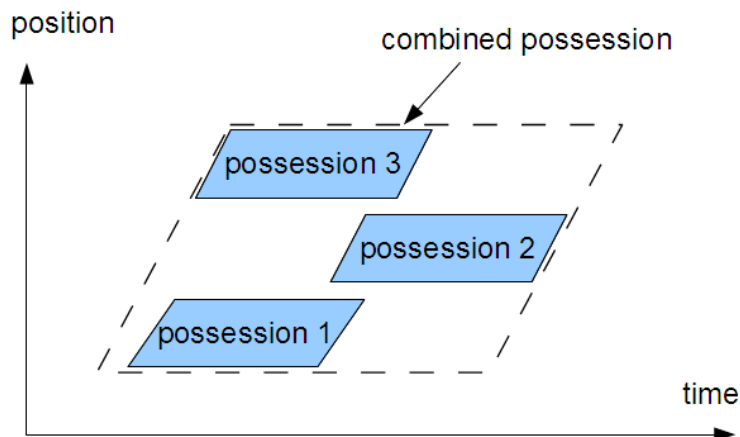


Figure 2 - Possession time computation (2)

In this work, we choose to compute the possession time as the sum of durations of all possessions which are located on distinct track segments and on distinct time intervals: this assumes that during one given possession the whole associated track segment cannot be used for commercial traffic. Although this assumption might be quite conservative, it takes into account the space aspect and the computation remains simple.

Nevertheless, we believe that this “absolute” way of computing possession time does not reflect the real capacity consumption, this is why we also introduce the impact on commercial traffic as an optimisation criterion, beside this computation of the possession times. A parallel can be drawn here with Leaflet 406 produced by UIC (2004) which tackles the evaluation of the capacity of a line, and points out that this evaluation cannot be done in an absolute manner, independently from the train paths.

State of the art

Several commercial tools devoted to supporting track maintenance scheduling are currently available (TILOS ⁷, SOG 5, Bentley Optram, Mermec RAMSIS, RTS Solutions). They are intended to facilitate the planning process by means of a graphical interface and modelling methods such as time-space or Gantt diagrams. Even if some intelligent calculation methods are incorporated in some modules of these tools, they do not implement facilities for a complete and **automated** planning and scheduling.

In order to achieve such automated planning and scheduling, research efforts are necessary in order to produce algorithms able to derive schedules that satisfy the numerous and complex constraints related to safety, maintenance processes and work regulation. Moreover, the computed schedule must be efficient (if not optimal) in terms of total cost. The best algorithms to solve planning and scheduling problems are in the field of Operations Research (OR) and Artificial Intelligence (AI). Over the last 60 years, many techniques such as mathematical programming (and especially Mixed Integer Programming) or

⁷ Web sites of the software cited in this section are given in Appendix 1.



metaheuristics (simulated annealing, taboo search, genetic algorithms...) have been proposed by the research community and applied to a large number of fields (transport, telecommunication, production management, finance...) A good overview of these algorithms and their application can be found in Leung (2004). Several commercial software libraries such as IBM ILOG CPLEX, Gurobi, Xpress or LocalSolver are available to help researchers and practitioners solve their optimisation problems. There are also many free software libraries (see for example COIN-OR).

As in production management, the relevant scientific literature on maintenance management (see e.g. Márquez, 2007) distinguishes between:

- long-term capacity planning problems that aim at fixing the day or the week when each maintenance operation must be done in order to optimise the railways' network availability and the use of maintenance machines (and other assets), and;
- short-term scheduling problems that aim at deriving a feasible schedule for all the resources impacted by the maintenance operations (timetable of the commercial trains, maintenance machines, human resources).



Figure 3: Main publications in the literature

Country	Reference	Objective function	Optimisation approach	Software engine	Calculation time	Planning horizon
Great Britain	Grimes (1995)	Profit maximisation	Genetic algorithms	Dedicated algorithm	-	-
Australia	Lake and Ferreira (2002)	Cost and disruption minimisation	Simulated. Annealing, local search, multip. LS, tabu search	Dedicated algorithm	Solution quality proportional to the calculation time	1 week
Korea	Oh et al (2006)	Minimisation of the maintenance effort under quality constraints, cost minimisation	Linear Integer programming	CPLEX	Optimal solution in about 6 hours, acceptable solutions in 1 hour	5 years
Japan	Oyama and Miwa (2006)	Maximisation of the total quality improvement	Linear integer programming	XPRESS-MP	Optimal solution in about 2 hours, acceptable solution within 30 seconds	1 year
Netherlands	Budai and Dekker (2004)	Minimisation of costs caused by service disruptions	Min to max, max to min, CEPFS	CPLEX	Depending on the scenario and heuristic, between 3 minutes and 4 hours	2 years
USA	Peng et al. (2011)	Minimisation of travelling costs	Linear integer programming	CPLEX	Convergence after 2 hours, network size not mentioned	1 year
Portugal	Vale et al. (2011)	Minimisation of costs under quality constraints	Linear integer programming	CPLEX	-	3 months
Germany	Quiroga and Schnieder (2011)	Maximisation of the total quality improvement under quality requirements, cost minimisation	Heuristic, own development	Dedicated algorithm	Under 1 second	1 year
USA	Peng and Ouyang. (2012)	Minimisation of travelling costs	Neighborhood search and MIP	CPLEX	Convergence after 2 hours, network size not mentioned	1 year



Figure 3 lists the main publications related to track maintenance planning and scheduling. We will not comment on them here since further details about these articles and software can be found in the AUTOMAIN WP5 working paper devoted to the state-of-the-art². We can, however, point out that all the articles address either long-term planning or short-term scheduling, although none considers the cooperation between these two classes of problems. From our experience, such cooperation is very important because long-term planning is typically based on models that ignore many constraints related to the traffic of commercial trains (and especially freight traffic). As a consequence, a long-term planning that seems to be good with respect to the objective functions may be difficult or expensive to apply (or even infeasible) when we consider all the traffic constraints.

Therefore we based our work on a global model in which both long and short-term scheduling can be addressed. This cooperative model is the main novelty presented in D5.1, even if each of the three models is new.

General overview

Methodology

CHIC-2 methodology is used in AUTOMAIN WP5 for the management of the optimisation project. This methodology was developed during the Esprit project 22165, completed in June 1999.

The CHIC-2 methodology is specifically dedicated to optimisation and industrial projects that address the resolution of large size combinatorial optimisation problems. This methodology does not replace the existing methodologies for IT project management. Its purpose is rather to supplement highlighting specific features appearing in projects involving the implementation of optimisation techniques.

The methodology is materialised by two working documents. First, the Problem Definition Document (PDD) is a semi-formal description of the problems of interest. It was shared with all of the WP5 team and with the leaders of the other WPs in order to converge to the most relevant models and to define the requirements for the graphical user interface (GUI). Second, the Problem Solution Document (PSD) is a formal and technical text to explicit the details of the models and algorithms. This document is shared between the WP5 members that developed the algorithms. Both documents are “living” in order to integrate any new model feature that could appear with new data sets or with the analysis of a case study.

Architecture of the tool

The state-of-the-art and the interviews with operational managers show that two types of scheduling problems appear. First, the long-term planning considers all the main maintenance tasks over a national or a regional network and schedules them over the next

² This document will be updated up to the end of the project. Section 6 of [deliverable D1.1](#) of project ACEM also gives a recent overview of the literature on the subject



year (or even over the next three years). Conversely, when a track possession for a maintenance operation is considered, additional schedules must be provided to adapt the timetable of commercial trains. Some trains must be cancelled and maintenance train paths must be inserted in order to move the maintenance machines that are required for the maintenance operation.

In order to derive an efficient global schedule, these two problems cannot be independently solved. If this were the case, the long term planning approach could derive some solution that appear to be good using the model but that prove to be infeasible when train paths are considered. Typically, the time to move a maintenance machine from one maintenance work site to another may be underestimated. Conversely, we can imagine building an “integrated model”, that is a model that contains all the decision variables and all the constraints (including traffic constraints) over a time horizon of several years and over the national (or even international) railway network. Clearly, due to the computational complexity of scheduling problems (see e.g. Leung, 2004), such a model would be too large to be solved in a reasonable time. Albrecht (2009) successfully developed models and resolution methods to solve integrated problems (i.e. building simultaneously trains timetables and track maintenance plans) but the instances considered are much smaller than the ones we address in WP5, which makes the approach unusable in our case. As a consequence, other models and algorithms must be designed. Therefore we propose a cooperative approach where three solvers work on a single shared data model. Figure 4 shows the hierarchy between these solvers and their access to the reference data.

- LTP (long-term planning) is in charge of scheduling all the maintenance operations, making sure that the individual planning of all machines is feasible and that general capacity constraints on the network are satisfied.
- TWG (time-window generation) is in charge of routing maintenance machines and splitting operations into track possessions (for example an operation can be divided into five time periods of 3 hours over 5 days).
- TWIP (time-window insertion problem) is in charge of solving localised assignments problem with accuracy and taking into account technical constraints (i.e. optimise the planning of a maintenance operation along with its corresponding maintenance train path).

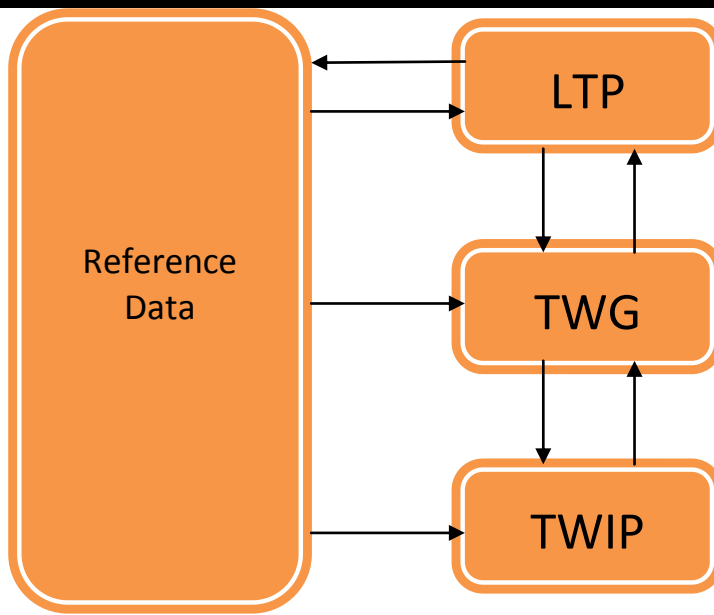


Figure 4: Collaboration between the three optimisation problems

The concept of time-window used in both TWG and TWIP is introduced as a generic term to refer to either a train-path of a maintenance machine or a track possession for maintenance works. Hence it represents the infrastructure capacity between two points of the network over a time-period. More details are given in the section related to TWIP.

LTP can call TWG in order to complete a long-term planning by routing the maintenance machines. TWG can also call TWIP to compute the optimised start and completion time of a maintenance task. The end-user that would like to re-optimize a part of the complete schedule, can call TWG or TWIP but the result must be checked by LTP, which must check and validate the impact of this re-optimisation on the complete schedule. Indeed, LTP is the only module that can modify the reference data. In order to improve the algorithmic cooperation between the modules and the decision support for the end-user, TWIP and TWP are designed to return several good solutions. Conversely, LTP can store only one validated solution in the reference data.

Cooperative optimisation problems and algorithms

Long-term planning

The LTP problem aims at proposing long-term planning solutions by computing, for each operation to be performed, a machine, a start date and an end date. It also needs to ensure that these macroscopic planning solutions respect constraints at more microscopic levels, which are local constraints over 24h periods, explicitly handled by the other modules, TWG and TWIP.

Hence, the output of LTP is an assignment of dates to each maintenance operation (start date & end date) along with a plan (Gantt diagram) of the use of available maintenance



machines over the time horizon. The individual maintenance machine plan must be feasible in terms of maintenance operations (the machines should be able to perform the operations between their assigned start and end dates) and in terms of routing (the machines should have sufficient time to be moved over the railroad network between two successive maintenance operations). Macroscopic capacity constraints, if any, can also be handled by LTP. Such constraints might include alternate paths which cannot be used simultaneously to perform maintenance operations or a limit on the simultaneous possessions over a single line or corridor.

The objectives to be optimised by LTP are a combination of objectives related to:

1. possession time (minimise total possession time required to perform all operations),;
2. impact on the commercial trains (if any, minimise delays and cancelations);
3. maintenance dates (perform maintenance at a date as close as possible to its given target);
4. machine use (use machines efficiently, especially minimise unnecessary travel times between operations).

Out of these four objectives, objective 1 is the only one strictly derived from the global AUTOMAIN objective, which is to reduce possession time. As mentioned earlier, we believe that a model with objective 1 only would lead to unrealistic solutions being produced by the tool. Indeed it might lead to planning solutions having a heavy impact on the traffic (train delays and/or cancellations), be very inefficient from the maintenance perspective and require unnecessary travels of the machines. The role of objectives 2 to 4 is precisely to avoid this kind of poor solutions being proposed by the algorithms.

One of the greatest challenges for LTP is the lack of fixed data concerning the duration of both operations and routings of the machine. Indeed, all these durations are only valid for a certain configuration associated with the assigned machine, the potential beginning dates of the operation, the simultaneity of other operations, the expected commercial traffic, etc. Every time one of these parameters changes, the durations must be recomputed through calls to TWG and, in turn, TWIP, since only these latter modules are precise enough to provide reliable information on the durations.

Traditional optimisation approaches are based on the assumption that all required data are available as soon as the algorithms start. In the case of LTP, due to the reasons stated above, this assumption does not hold. Moreover, the computation effort induced by a call to TWG (and subsequently to TWIP) requires significant computation time. Consequently, the number of calls to TWG must be minimised and the cases where TWG is called must be chosen carefully among all possibilities.

Given this computation time constraint, we adopt a specific three-step approach to solve the LTP problem. The idea behind this decomposition is to design in a short computation time a first good solution by implicitly approximating microscopic constraints. After this first “draft” solution, TWG and TWIP are called either confirming the choices that were estimated (by refining the details of the schedule) or rejecting some of the assumptions



which were made during the estimation. Once this is done for all operations, planning changes are tried iteratively to improve the solution until the allocated computation time is over.

First step: Building an initial plan based on an approximated model

The goal of this first step is to produce a general view of a reasonably good solution within a very short computation time (say, at most 1 minute out of 1 hour for the whole computation). To achieve this goal, the microscopic constraints handled by TWG and TWIP are not explicitly taken into account.

The approach is based on a classical priority list algorithm where, basically, the most critical operations and those with the earliest due dates are the first to be inserted in the plan. Simultaneously, the required resource which is the most able to perform an operation around its target date is assigned to the operation. This process is repeated until all the operations are scheduled, in other words until all the operations are given a beginning date and a machine.

One of the most important features of this step is that the TWG and TWIP modules are never called: only basic rules are used to estimate the duration of operations and machine routing throughout the network between two consecutive operations. For instance, the duration of operations is estimated based on the distance to cover, the operation type and an approximation of the distance covered every day for such an operation type. In a similar way, the routing times are approximated using the distance as the crow flies between two successive operations, at an average machine travel speed. Using only estimators has two major consequences:

1. The computation effort is limited to macroscopic considerations, hence the algorithms can be very fast, and
2. The resulting plan is only based on estimations, and can be somewhat unfeasible as the duration of operations can be over-estimated (in which case the machines could be used more efficiently), or under-estimated (in which case there is not enough time to complete the work before moving to the next operation in the plan).

Despite these approximations, this initial plan gives a good overview of what a realistic solution could be, and hence can help focus the calls to TWG and TWIP on realistic configurations during the second step.

Second step: Validating the plan

The algorithm starts with the initial estimate plan computed by step 1. For each operation o of the plan, a call to TWG is made reflecting the situation in the estimate plan: same beginning date, same machine. If TWG returns a solution in which the actual machine routing plus the duration of the operation is shorter than the one which had been estimated, we validate the assignment of operation o to the machine and the beginning date by updating the reference data. If, on the contrary, the actual duration is longer, this can lead to an inconsistency with the performance of the next operation (say o') scheduled on the machine in the estimate plan.



Therefore, a choice has to be made: either operation o is rescheduled somewhere else in the planning (other date and / or other machine), or operation o' is rescheduled. Other choices could also be made but this option is not considered here. We decide to reschedule either o or o' based on their estimated duration: we pick the shortest one with the heuristic assumption that this will have the smallest impact on the rest of the planning. To reschedule the operation, we use the same approach as in step one, using estimators in a permanent attempt to save computation time.

The algorithm proceeds until all operations are validated (along with their operational details) by solutions returned by TWG. We ensure that every time an operation is validated in the plan, all constraints are satisfied, i.e. the solution is feasible. If some constraint is not respected, another assignment (either its resource or its beginning time, or both) has to be found. Once again, we use the approach described in step 1 to deal with such cases. As an output of this second step, we obtain a plan where all maintenance operations are decomposed into workdays by TWG, thus providing their real duration, are inserted in the train grid by TWIP, and where all routings of machines are also confirmed by solutions returned by TWG.

Third step: Optimising the plan

Finally, the third step consists of optimising the solution obtained as a result of the previous step. At this stage, the solution is feasible, in the sense that all decisions are consistent with both macroscopic and microscopic constraints. The objective of this last step of LTP is then to improve the existing plan until a time limit is reached.

The improvements are achieved by a local search algorithm where the neighbour solutions are obtained by swapping operations, either adjacent operations on the same machine, or in a more general way between any two operations of the planning. The selection method is descent; only improving neighbours are considered. If the remaining computation time is sufficient, more sophisticated selection methods could be used, such as simulated annealing or taboo search.

Time-window generation

As shown in the description of the LTP module, it deals with long term problems (up to several years) and large sections of the railway network (up to a whole national network). On the other hand, the TWG module deals with short term problems (up to 24 hours) and smaller parts of the railway network.

Furthermore, neither LTP nor TWIP deal with the creation of feasible routes for maintenance resources from one maintenance operation (or even sub-task) to the next one. This is a crucial task, since movements from one point of the network to another can take several hours, even days, and this is time during which the maintenance machines are not being used for maintenance itself, but are just travelling.



On the light of these remarks, it becomes evident that an intermediary module which, on the one hand, decomposes large problems into a set of smaller problems and, on the other hand, finds short routes between and within operations, is necessary. The time-window-generation (TWG) module deals with both these tasks. From the functional point of view, the purpose of TWG is thus twofold.

Both these tasks are carried out under consideration of the constraints passed by LTP, i.e. initial machine location, machine availability, machine and operation types with their associated speeds; as well as the railway network topology characteristics, such as maximal speeds, allowed travelling directions, allowed switch and crossings moving directions, etc.

Next we describe the chosen approaches.

Large problem decomposition

TWG creates different modes for the accomplishment of an operation. This is achieved by dividing an operation into different numbers of shorter operations – we will call them *maintenance tasks* – that have a maximal duration of 24 hours³.

The decomposition procedure of an operation into a mode with n tasks comprises the following steps:

1. The operation is divided into n tasks.
2. For each task, the machine routes are calculated as described in the paragraph below (*Maintenance machine routing*)
3. By calling TWIP, the feasibility and cost of each task is assessed.
4. By summing the cost of its tasks, the total cost of the mode is calculated.

Since TWG does not know what the optimal number of tasks is, this is found out by means of the following procedure:

1. For each operation, this procedure is repeated for $n= 1, 2$ and 3 .
2. If the mode with three tasks ($n=3$) is the one with minimal cost, then the mode with $n=4$ is also calculated.
3. If the cost with $n=4$ is less than with $n=3$, then the mode with $n=5$ is calculated and so on until the cost begins to rise. However, the maximal number of tasks is restricted by the time frame allowed for the operation, which is specified by LTP.

By doing this, the optimal number of tasks is assessed.

Under some circumstances, LTP could prefer not to choose the optimal mode for the realisation of an operation, but may choose another one. For example, suppose that the total cost of the mode with $n=4$ is 98.4 and the total cost of the mode with $n=3$ is 102.2. The mode with $n=4$ is cheaper, but the mode with $n=3$ has the advantage of being one day shorter (remember that each task lasts 24 hours). Thus, LTP could prefer to pick to schedule

³ According to the available data to build a model, these maintenance tasks could be either a macroscopic task that corresponds to the required track possession or a list of shorter operations. In the current state of the code and the test, these tasks are macroscopic.



the mode with $n=3$ in order to save one day. For this reason, TWG does not only return the best mode found, but the 3 best ones. Then LTP decides which one among them will finally be scheduled.

Maintenance machine routing

TWG is responsible for finding the most suitable routes for the maintenance machine to perform the operation in each of the created modes. As an example, consider an operation with two tasks T and T' . The complete machine route would then consist of the following paths:

1. From the initial machine location to the initial location of T
2. From the initial location of T to the final location of T
3. From the final location of T to the next depot (the location where the machine is stored before T' starts)
4. From the depot to the initial location of T'
5. From the initial location of T' to the final location of T'
6. From final location T' to depot

The routing problem is also quite complex, since the time needed to travel along each track is dependent on a series of factors:

- travelling direction: most railway lines have two tracks, each of them having a preset travelling direction, for which the signalling is arranged. The allowed travelling speed in this preset direction can be up to 320 km/h for high speed lines and 160 km/h for regional lines. Travelling in the opposite direction is only possible at very low speeds, e.g. 40 km/h.
- machine maximal speed: the actual speed the machine will move at is limited not only by the allowed track speed but also by the machine speed, which is in general lower. For example, a typical maximal speed of a tamping machine is 80 km/h.
- track maximal speed: see *travelling direction*.
- timetable. Maintenance machines have, in general, the lowest priority, so they have to let any other passenger or freight train pass. Thus, it could be more convenient to take a longer route with lower traffic density than the shortest one. This could in turn depend on the time of day: for example, one route could be the most convenient at night, when there is barely any passenger train traffic, but another one could be more convenient during the day.
- switches and crossings: in standard graphs, (for example road networks), for a vehicle to be able to pass from one edge to another, it suffices that they share a common node. On the contrary, in railway networks, a train cannot always directly pass from one track to another, even if they share a common switch or crossing. In fact, at a switch there is one incoming track and two outgoing tracks, at a crossing two incoming and two outgoing tracks. It is only possible to pass from an outgoing to an incoming track and vice-versa, but it is physically impossible to pass from an incoming to the other incoming track as well as passing from an outgoing to the other outgoing track directly. However, and as long as the machine is able to travel in both directions - which in general holds - it is possible to pass from an incoming to another incoming track via an outgoing track, i.e.



incoming track 1 → outgoing track → incoming track 2

Of course, this manoeuvre takes some considerable time, which must be computed to the route. So the approach used by TWG is not to forbid passing between incoming or outgoing tracks, but to penalise such movements with an additional manoeuvring time.

The standard approach for finding the shortest paths within a graph is Dijkstra's algorithm. However, it assumes that the weights (i.e. the cost of traversing) of the graph edges are constant in time, which in this case clearly does not hold. For this reason, TWG uses the variant described by Dean (2004) that is compatible with dynamic edge weights. The central idea of this variant is that the weight of each edge is calculated individually, each time a route including it is considered, and this weight will be dependent on the precedent route edge (in order to add manoeuvring if necessary), the travelling direction, and the time of day at which the machine would be expected to traverse the track.

Time-window insertion

The time window insertion problem (TWIP) is a short-term scheduling problem, aiming to insert one or several inspection or maintenance paths on a given day in between the commercial train paths. The temporal horizon of the problem is typically 24 hours (even less than 24 hours if the time period of the day when the operations must be placed is known).

The maximal computation time allowed to solve the problem is one hour but in general it is possible to have a good solution during the first few minutes of calculation. As discussed above, instances of TWIP must also be solved in a sub-routine in order to solve TWG and TWIP. Clearly, this can be done only if the computation time is no more than a couple of seconds.

Input

Basically, TWIP is based on shared common data (that is "Reference Data") with some additional data provided by TWG. In particular, these reference data contain a list of commercial trains with their route and timetable. In the current version of our model, the commercial route is fixed and cannot be changed by the solver⁴. In contrast, the timetable can be modified (that is the train can be delayed or even cancelled).

Additionally, TWG can provide one or several maintenance tasks (whose duration is at most 24 hours). As there are different ways to achieve this maintenance task, TWG may propose several "modes" (defined in the section about TWG). TWIP will choose the best one when optimising.

Finally, a maintenance machine may be required to process a task. TWG has defined several possible paths, and the optimisation will choose the best one.

The starting times of the time windows (for either the train paths or the maintenance work) are not given; they are the main decision variables to be computed by the algorithm. Conversely, all the time-windows have both a start point *S* in the network and an end point

⁴ An operational version of the scheduler should clearly allow the possibility of rerouting some commercial trains.



E . Points S and E must be on the same track segment or at least on the same line, meaning that the path from S to E in the network must be obvious following the track. For both maintenance tasks and maintenance train paths, a security time is attached specifying the occupation time of the track during the travel (or maintenance task). Train paths can be considered as time-windows with a specific security time depending on their category (see Figure 5). Moreover, train paths usually go through several track segments. Hence, the train path is a list of elementary train paths (represented as simple quadrilateral time-windows in the time-distance graph) that are successively linked by *no-wait* precedence constraints. For example, in Figure 5, the train path between S and E' is made of two elementary time-windows. The second one must begin when the first one ends.

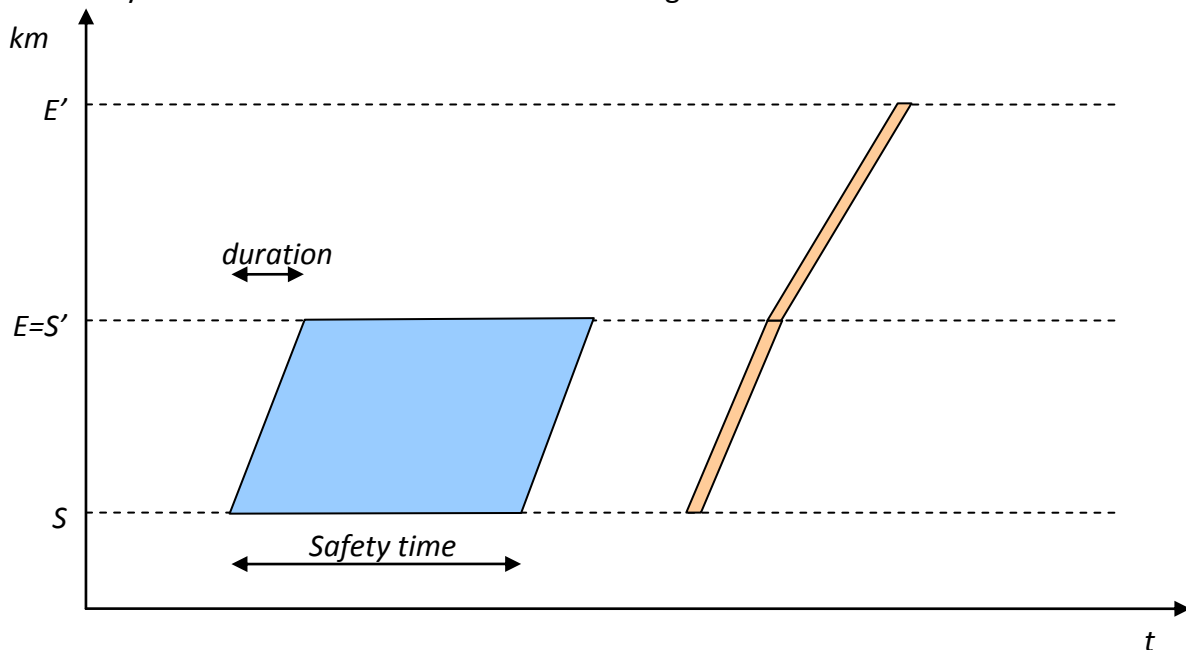


Figure 5 - A possession time window and a train path in a space-time graph

Output

The algorithm computes the schedule for the maintenance machines and the start and completion times of the maintenance tasks that have been selected. Since the so-scheduled maintenance can disturb the commercial trains, the output contains a list of cancelled and delayed trains.

Constraints

There are three main classes of constraints:

- For commercial trains
 - The time deviation between the originally scheduled time of a train and its rescheduled time must remain below a given value which depends on the type of the train. Otherwise, the train must be cancelled.
 - The speed of a train is bounded by both a minimal and a maximal speed. Both values depend on the type of train.
 - Trains must slow down if there is a maintenance task on an adjacent track.
- For maintenance machines



- Each movement of a maintenance machine has an earliest start time and a latest completion time.
 - The speed of the maintenance train is limited. This value depends on the type of train maintenance.
- For maintenance tasks
 - The task must be performed between a given earliest start time and a given latest completion time.
 - Some tasks must be done before others (precedence constraint).
 - Some tasks must be performed using a maintenance machine (resource constraint).
 - Some tasks must be performed simultaneously.

The definition of TWIP also adds constraints of non-conflict between different elements (maintenance tasks, commercial trains or movements of maintenance machines) that occur on the tracks. There must always be a safety time between two of their time-windows when they are at the same place. This time depends on the type of the time-windows.

Objectives

The goal of TWIP is to automatically insert all the maintenance tasks between the commercial trains already set in the schedule. Several elements are taken into account in the objective function. The first one is to minimise the penalties occurring for each delayed or cancelled commercial train. The second one is to minimise the duration (time span) of the maintenance tasks. The last one is to minimise the cost associated with the selected mode.

TWIP algorithm

Basically TWIP is modeled as a MIP (Mixed integer programming). Since the TWIP problem is not linear, we had to linearise it using standard methods. The comprehensive description of the mathematical model is related in described in a technical document (Eurodecision, 2012). The solver IBM ILOG CPLEX Optimizer (version 12) is used to solve the MIP.

Interfaces

The mathematical models and the algorithms introduced in the previous section are not enough by themselves to improve the schedules in practice. They must interact with the decision maker through an efficient and clear data interface and with the real world so that the abstract parameters of the models (edges and nodes of the graphs, speeds, task durations and so on) can be replaced with real data.

This section first proposes a Man-Machine Interface (MMI) that integrates the optimisation functionality but also displays the main features of a computed solution. This MMI is proposed as a prototype and it is out of the scope of the AUTOMAIN project to implement it. However, we will implement a simpler user interface derived from this proposal in order to analyse the results proposed by our algorithms.



User interface

Background

The planning and scheduling tool receives condition data (diagnosis / prognosis) about the track and track elements (e.g. switches, point machines) based on the results of WP3. Timetables as well as available maintenance machines and resources are also given as an input. The user plans all activities based on the condition prognosis. After the planning is finished, the automated scheduling tool optimises the maintenance activities and creates a scenario(s). The user can evaluate the results and the conflicts, return to the planning phase to change the input for the scheduling tool, or choose one of the created scenarios.

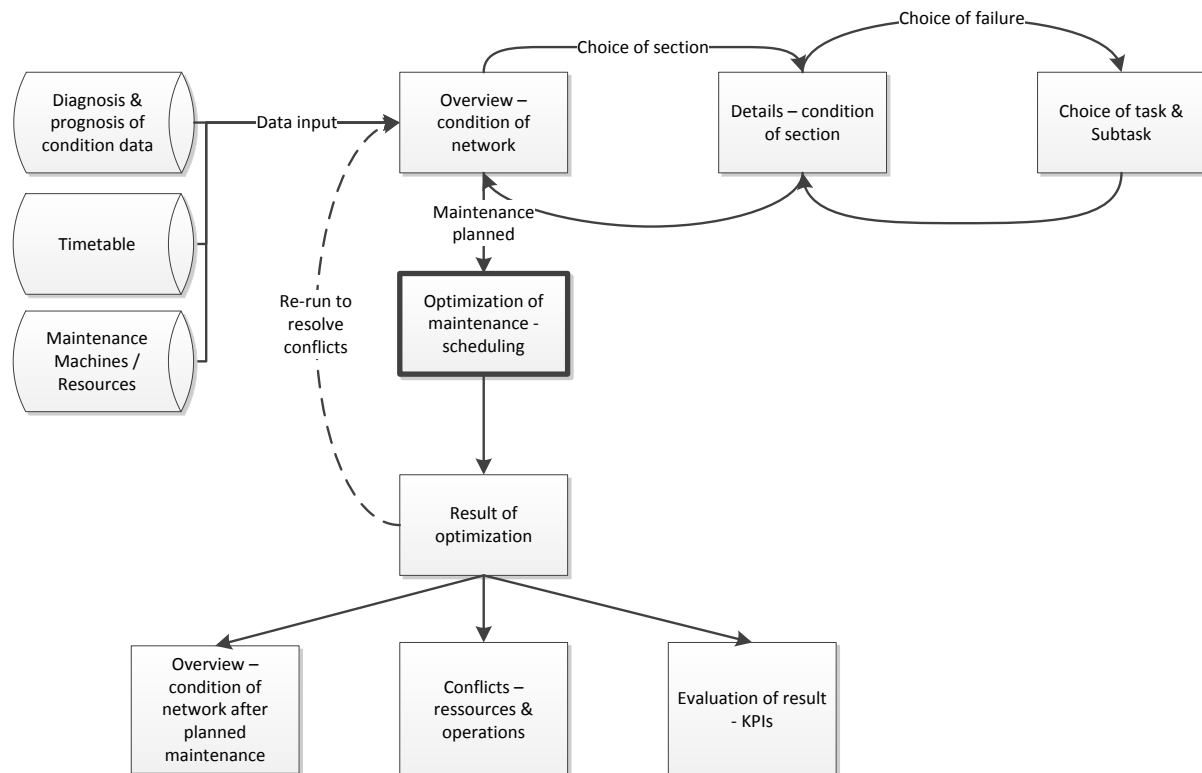


Figure 6 – Overview on the functionality of the user interface

Description

Users/ audience:

The MMI will be used by infrastructure managers who are currently planning and scheduling the maintenance manually. Thus, the users are very familiar with the processes.

Structure:

The information is displayed in a way which makes it easy to read without causing any confusion. This is provided by: labelling (title at the top of each page), information displayed as 'feedback/current state' text-box on the right side and 'navigation/help' field at the bottom of each page.

It is important that the interfaces provide clear feedback about any actions taken by the user. This includes feedback not only to acknowledge the action that the user has carried out with the interface but also to the consequences of any action. The system allows the user to navigate the site quickly.



Features and Functionality:

The characteristics of the site are described in terms of the dialogue requirements contained in principles of the ergonomics standard ISO 9241 (suitability for the task, self-descriptiveness, conformity with user expectations, suitability for learning).

Description in Detail

The MMI is presented as a screen-based interactive prototype. It offers the possibility of interaction - for example, by clicking the mouse on the representation of one of the page's controls, which accordingly will change the state of screen-based representation.

Page 1: Overview-condition of network

Structure:

The central area is a draggable map that can be moved with the left mouse click in all directions. "Navigation/help" area at the bottom of the page provides clear instructions about all actions that the user can take. The title "Overview-condition of network" simplifies the navigation and orientation. "Feedback /current state" on the right side shows the information about areas needed to be done (yellow and red areas).

Features and functionality

Three main colours are used to show the state of the certain track section ("red" means "need to be done urgently", "yellow" means "need to be done", "green"- "nothing should be done").

The track sections can be selected with the left mouse click to get more details. (Feedback/current state). The menu buttons on the right side complete the functionalities of the page: with the "Start optimisation" button optimisation of maintenance scheduling can be started. The results are shown on the page 4 "Result of optimisation".

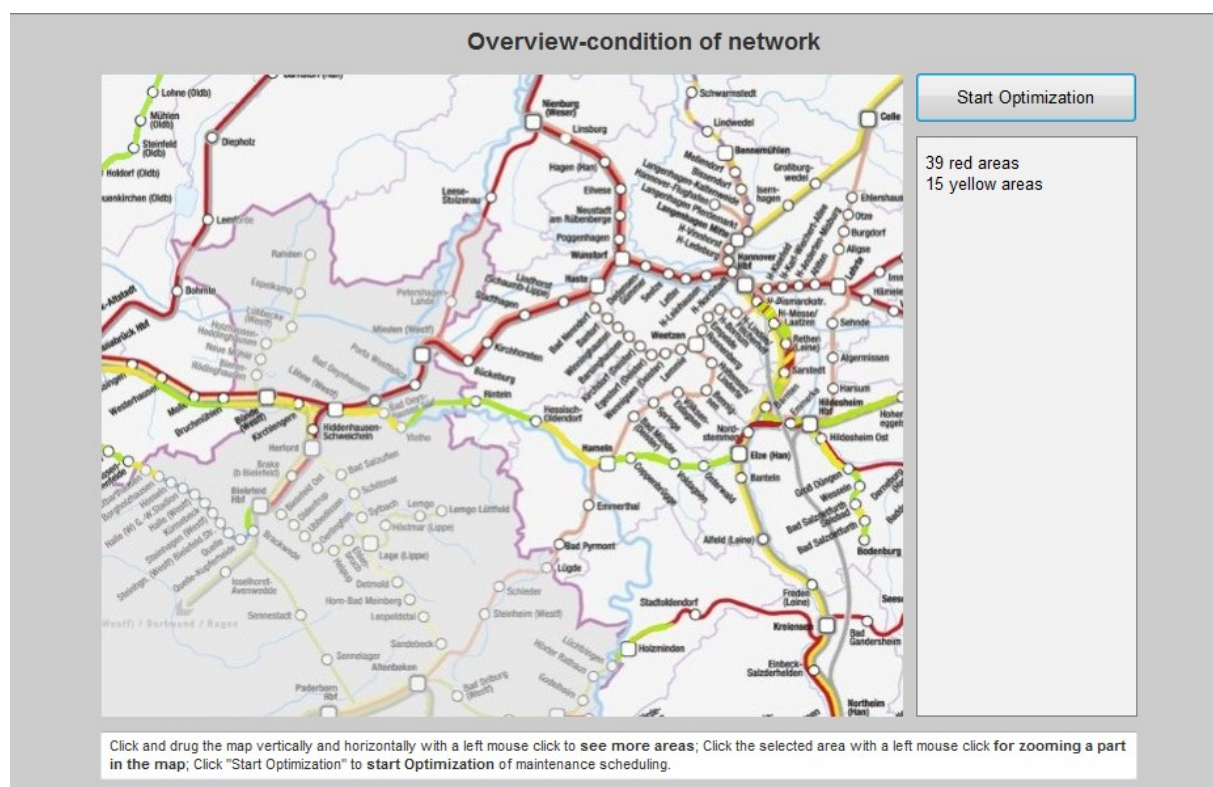


Figure 7 – Overview of the condition of the network



Page 2: Detail-condition of section

Structure:

The central area is a detailed map. “Navigation/help” area at the bottom of the page provides clear instructions about all actions that the user can take. The title “Detail-condition of section” simplifies the navigation and orientation. “Feedback/current state” on the right side shows the information about areas that need to be worked with (yellow and red areas).

Features and functionality

Three main colours are used to show the state of a certain track section (“red” means “need to be done urgently”, “yellow” means “need to be done”, “green”- “nothing should be done”).

The detailed track sections can be selected with the left mouse click to get more information about red and yellow sections. (Feedback/current state). The menu buttons on the right side complete the functionalities of the page: “back” takes users to the previous page “Overview-condition of network”, “Choice of operations” take users to page “Choice of task and subtask”.

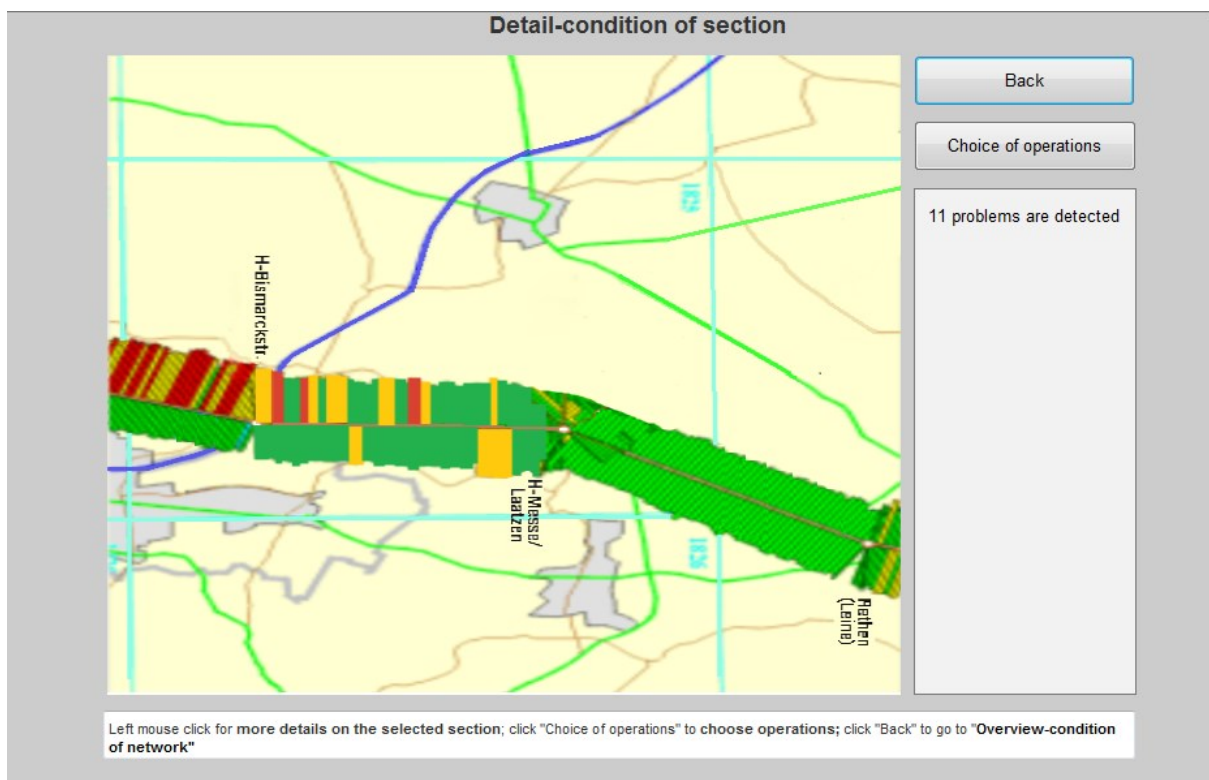


Figure 8 – Detailed view of the condition of track segments

Page 3: Choice of task and subtask

Structure:

“Description” (of the problem) area is to be found under the title. The central area is divided into six parts: six buttons represent solutions for the selected problem in terms of operation, time, machine, resource and order. Each button has a Submenu (on click). “Navigation/help” at the bottom of the page provides clear instructions to all actions the



user can take. The title “Choice of task and subtask” makes the navigation and orientation easier. “Feedback/current state” on the right side shows how many areas need to be done.

Features and functionality

The following submenus can be chosen: “Operation”, “Time”, “Machine”, “Resource” and “Order”. Multiple selections are also possible. The “Feedback/current state” textbox shows the selections. The “Back” button takes the user back to page 2 and deletes the selection. The “Done” button confirms the selection and takes the user to page 2. The selected detailed track section will be marked with grey lines to show that this section is already done. “New choice” can be used to start a new selection.

Figure 9 – Assignment of operations to a track segment

Page 4: Result of optimisation

Structure:

The central field shows a Gantt diagram for each operation. The “Navigation/help” area at the bottom of the page provides clear instructions to all actions that the user can take. The title “Result of optimisation” simplifies the navigation and orientation. “Feedback/current state” on the right side shows how many conflicts have been detected.

Features and functionality

“Back” take the user back to the page 1. The “Evaluation of results” button opens page 6 “Evaluation of results”. The “Conflicts” button leads to page 5.

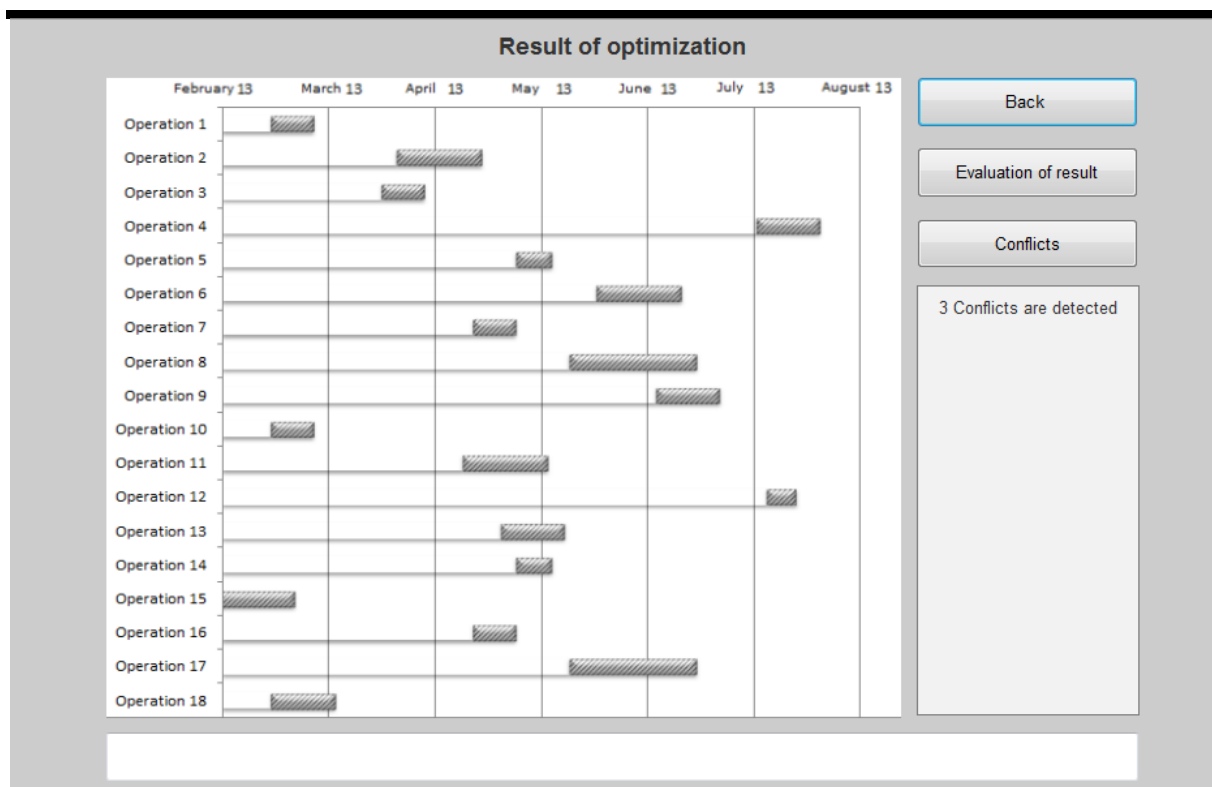


Figure 10 – Overview of the results of the optimisation

Page 5: Conflicts

Structure:

The central area is a diagram that shows the conflict areas. The “Navigation/help” area at the bottom of the page provides clear instructions about all actions that the user can take. The title “Conflicts” simplifies the navigation and orientation. The “Feedback/current state” section on the right side shows how many conflicts are detected. Buttons: “Resource Conflicts”, “Time Conflicts”, “Machine Conflicts”, “Order Conflicts” and “Capacity Conflicts” can be clicked for further information.

Features and functionality

Clicking on the button “Resource Conflicts” will display all resource conflicts, clicking on the button “Time Conflicts” will display all time conflicts. Clicking on the button “Machine Conflicts” will display all machine conflicts. Clicking on the button “Order Conflicts” will display all order conflicts. Clicking on the button “Capacity Conflicts” will display all capacity conflicts. The “Back” button will display the optimisation results (page 4).

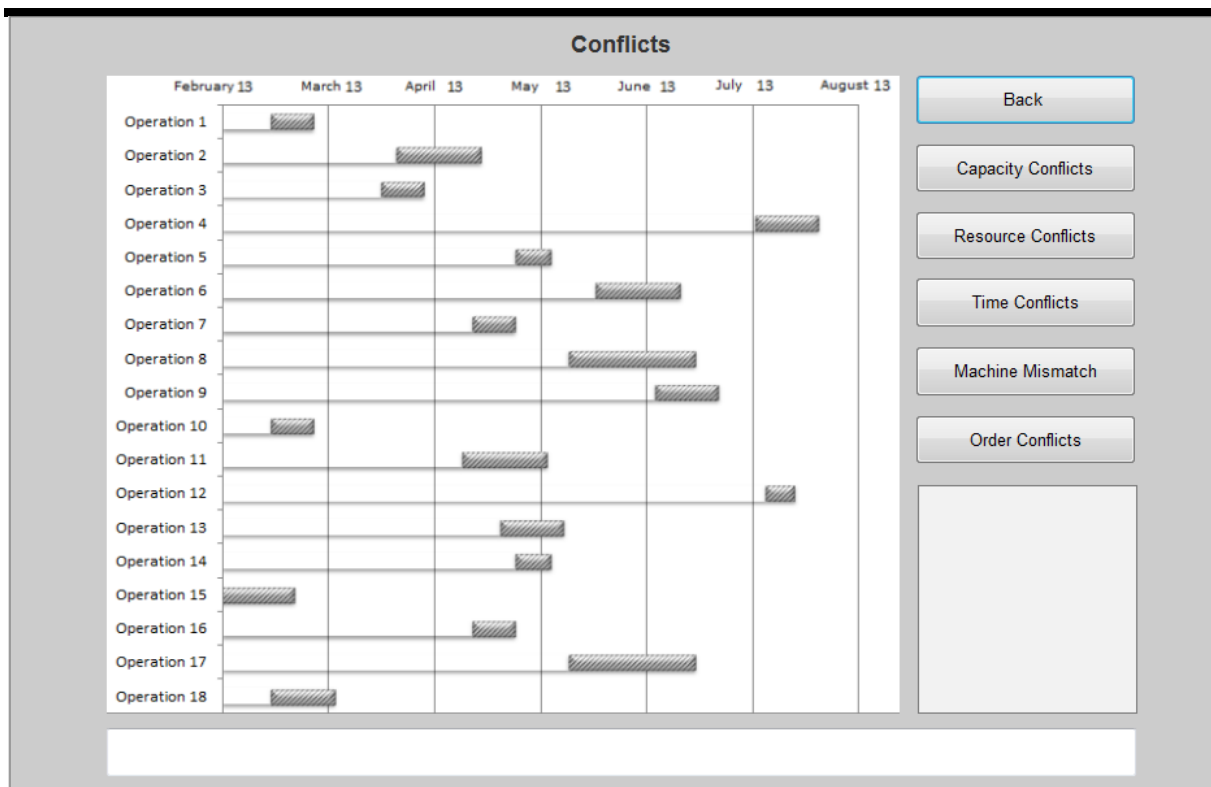
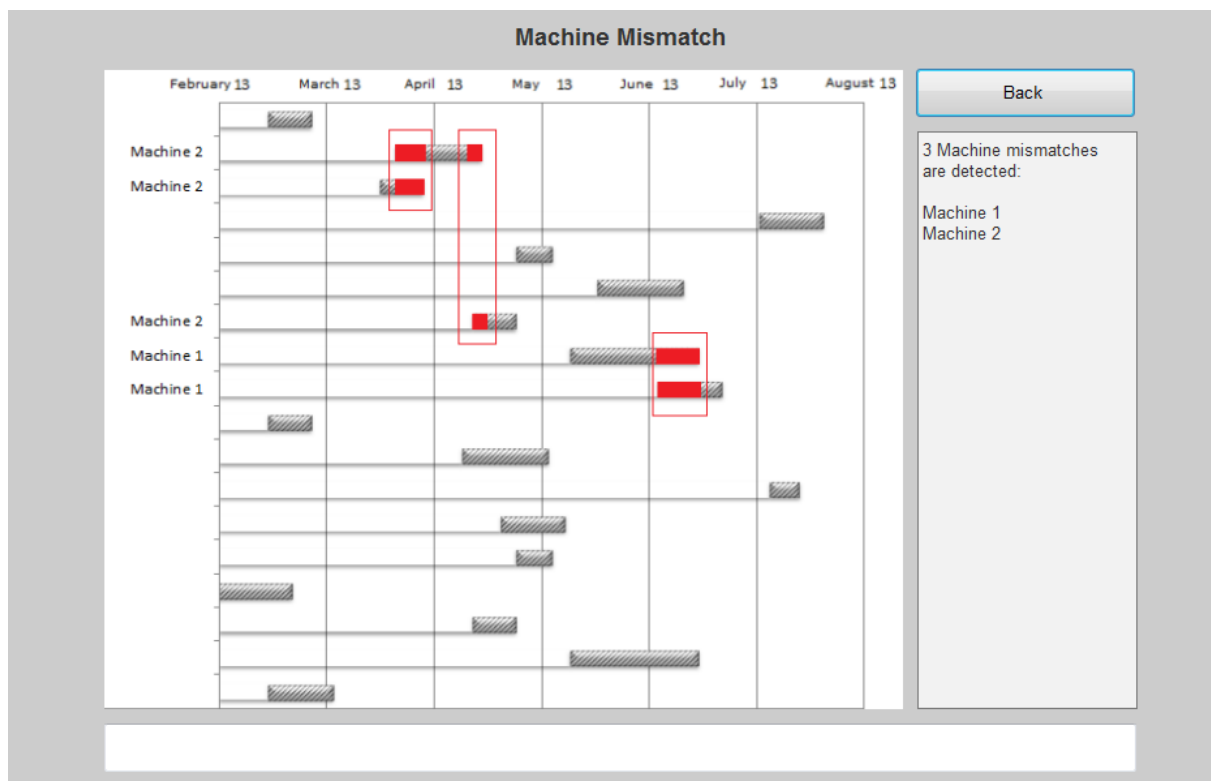


Figure 11 – Overview of conflicts of the optimisation result





Page 6: Evaluation of result

Structure:

The central area is a bar chart that shows the value of the KPIs for some possible planning scenarios. The “Navigation/help” section at the bottom of the page provides clear instructions to all actions that the user can take. The title “Evaluation of results” helps by navigation and orientation.

Features and functionality

The “Back” button leads back to the optimisation results (page 4), “Delete Scenario” to delete a scenario, and “Choose Scenario” to choose a scenario.

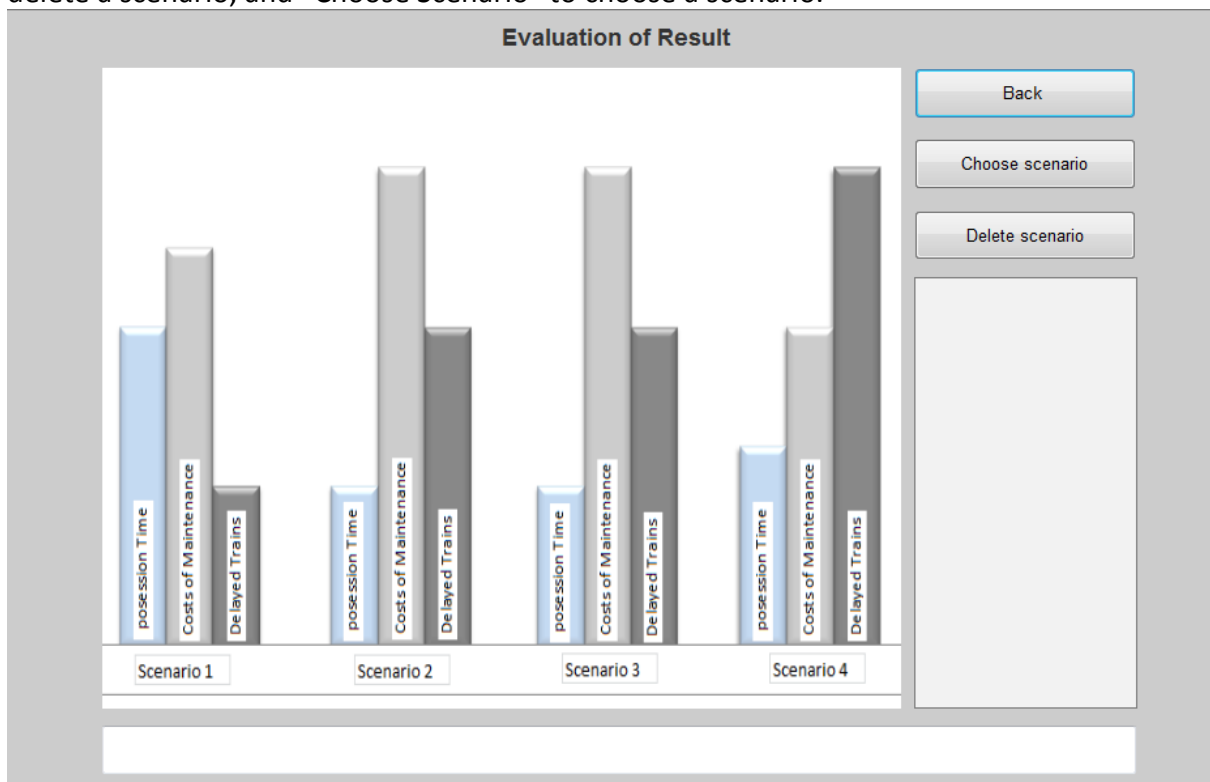


Figure 13 – Evaluation of the results of several optimisation scenarios

Data interface

As observed by Sashihara (2011), getting good data is a key factor for a successful Operations Research project. Indeed, an OR model typically collects data from different sources and data inconsistencies usually have a great impact on the solutions computed by the algorithms: if the duration of a move between two points is missing (or strongly underestimated), the solution will often move the machine between these two points! In order to limit the risks related to the availability of data, the development of an OR model should start only after the data are collected and analysed by OR experts.

In research projects and especially in collaborative research projects, there is usually no reference information system and existing information systems usually contain only a partial amount of the data required by the models. In other words, the implementation of an operations research tool must go together with a major upgrade of the information system.



When a tool is not designed to be integrated in an existing system, as is the case for our planning and scheduling tool, a clear and open interface must be designed to make possible a future integration in a system. RailML (Nash *et al.*, 2010) is an emerging XML-based interface language for railway applications. Its specification contains three subschemas for three areas: infrastructure, timetable and rolling stock. In the AUTOMAIN project, a subtask of WP3 is aimed at proposing a new subschema for infrastructure maintenance.

However, when we started the development of the tool, we noticed that railML is quite comprehensive and therefore complex but there is no free software library to read and write files. Moreover, as the work in WP3 was at a preliminary stage and did not intend to contain all the data required by our models, we decided to make public our own data interface instead of working with railML. This decision was also motivated by the fact that we have no real data in railML format and the risk of spending too much time understanding railML and writing libraries for it was very high. Our data interface is also an XML-file and is directly derived from our data model, that is the “reference data” introduced in the “Architecture of the tool” section.

In order to test the tool, we produced a complete data set with a network of 6722 km of tracks with 1274 edges and 704 nodes in which there are 280 commercial trains and 200 requests for maintenance operations. In order to generate this file, we used several real data sources (the data in the file are “blurred” for confidentiality). Similarly, new data from other sources (typically from WP3 and WP4, eventually in railML files) are available: a converter to our format must be written.

Conclusion

Tool implementation and preliminary tests

The three planning and scheduling algorithms presented in this report are implemented in C++. We use a subversion (SVN) repository to manage the versioning and share the code between the Technical University of Braunschweig, EURODECISION and SNCF.

These three algorithms work independently but the integration of the three modules is not yet fully available, in the sense that the algorithms cannot yet use results from each other. For instance, as of now (January 2013), LTP cannot use TWG to validate the estimated plan which is generated during the first step of the computation. Likewise, TWIP cannot be used by TWG. The reasons behind these issues are related to the fact that most of the work was done with data sets generated by each partner for its own module, which eventually turns out to be different (in data size and in complexity) from what is really provided in the test instances.

Nevertheless, the overall architecture is now well settled, data sets can be read from xml files and data are available to all modules. Some tests were run for each of them. The syntax of the command line used to launch the executable file (program) is as follows:

automain [LTP|TWG|TWIP] inputData.xml dataSchema.xsd



This enables each module to be called independently, and to precise the input XML data set as well as the XML schema to use.

In response to this command, LTP is able to produce, within less than a second an estimated plan for the complete data set described in the previous section. A sample program output is given in Appendix 4.

In appendix 5, the performance TWG is demonstrated by means of two case studies, where both the ability to find shortest paths within a railway network and the ability to subdivide operations into several short operations are tested.

Appendix 6 is devoted to the preliminary results for TWIP. They show that we can only solve instances that are smaller than real-size instances (with about one hundred commercial trains).

Future works

Clearly, the next objective is the integration of the three modules, so that they can work with each other. This objective is a challenging but necessary step to get an efficient planning and scheduling prototype able to deal with both macroscopic and microscopic constraints. To reach this goal, we must improve the computational efficiency of our algorithms. The efficiency of TWIP is of prime importance since several hundred instances of this problem must be solved to validate the feasibility of the long-term plan. The current implementation of TWIP is more general than what is strictly required for the collaborative approach. We must then address this special use case of TWIP (see Section 4.2 of AUTOMAIN D1.2).

The second objective is to demonstrate the results of WP3 and WP4. Such demonstration is very difficult since it requires much data and a close cooperation. To limit the risk the models were presented several times to the AUTOMAIN Project Management Team. They were also presented at the AUTOMAIN dissemination session in October 2012. Finally this document is finally very important so that every member of the project team can understand what data are required for the demonstration. Appendix 2 lists them while Appendix 7 is the formal description of the data format.

The data from WP3 and WP4 are also required to show that the capability requirements specified by AUTOMAIN D1.2 (and listed in the introduction) are fulfilled. Some enhancements of our models may be required for example to show the impact of grouping operations.

To date, Task 5.1 has been completed (the state-of-the-art report will be updated if necessary) as well as Tasks 5.2.1 and 5.2.2. Tasks 5.2.2, 5.2.3, 5.3.2 and 5.3.3 are 80% completed. The rest of these tasks will consist in the completion of the tool. Tasks 5.2.4 and 5.3.4 are devoted to building the demonstration.



Appendix 1: References

Scientific literature

Abrecht A. (2009) *Train timetables and track maintenance: New mathematical techniques for producing integrated schedules*. VDM Verlag.

CHIC-2 (1998) *The CHIC2 Methodology User Guide*, Deliverable D3.3 for the European funded project 22165.

Dean B.C. (2004) *Shortest Paths in FIFO Time-Dependent Networks: Theory and Algorithms*, Technical report, MIT, Cambridge, MA.

Grimes, C.A. (1995) Application of Genetic Techniques to the Planning of Railway Track Maintenance Work. In: Zalzal, A.M.S., Hrsg.: *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, GALESIA, Bd. 414. Institution of Electrical Engineers. Pages 467-472.

Jenema, A.R. (2011) An optimization model for a train-free-period planning for ProRail based on the maintenance needs of the Dutch railway infrastructure. Masters thesis. Delft University of Technology.

Leung J.Y.-T. (2004) *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press.

Márquez, A.C. (2007) *The maintenance management framework: models and methods for complex systems maintenance*. Springer Series in Reliability Engineering.

Nash A., D. Huerlimann, J. Schuette and V.P. Krauss (2010) RailML – a standard data interface for railroad applications. In I. Hansen (Ed) *Timetable planning and information quality*. WITpress. Pages 3-10.

Oh, S.M., J. Lee, B. Park, H. Lee and S. Hong (2006). A study on a mathematical model of the track maintenance scheduling problem. *Computers in Railways X*, p. 85-97.

Oyama, T. and M. Miwa (2006). Mathematical modeling analysis for obtaining an optimal railway track maintenance schedule. *Japan journal of industrial and applied mathematics*, Volume **23**(2), pages 207-224.

Quiroga, L. M., M. Antoni and E. Schnieder (2011) Heuristic based track maintenance scheduling optimisation. 9th World Congress of Railway and Research (WCRR 2011), Lille, France.



Peng, F., S. Kang, X. Li, Y. Ouyang, K. Somani and D. Acharya (2011) A Heuristic Approach to the Railroad Track Maintenance Scheduling Problem. *Computer-Aided Civil and Infrastructure Engineering*, Volume **26**, pages 129-145.

Peng, F. and Y. Ouyang (2012) Track maintenance production team scheduling in railroad networks, *Transportation Research Part B: Methodological*, Volume **46**, Issue 10, pages 1474–1488.

Sashihara, S. (2011) *The Optimization Edge: Reinventing Decision Making to Maximize All Your Company's Assets*. McGraw-Hill.

Web sites and software

ACEM (2011) D1.1 Report on the state of practice of railway infrastructure maintenance. http://www.acem-rail.eu/documents/ACEM-Rail_%20D1.1_State%20of%20practice_r0.pdf

Bentley (2012) Bentley Optram: Railway Maintenance Decision Support System, <http://www.bentley.com/en-US/Products/Bentley+Optram/product-Overview.htm>. Last visited December 2012.

CHIC-2 (1999) Creating Hybrid Solutions for Industry and Commerce, Esprit Project 22165. <http://cordis.europa.eu/esprit/src/22165.htm>

CPLEX (2012) IBM ILOG CPLEX Optimizer: High-performance mathematical programming solver for linear programming, mixed integer programming, and quadratic programming. <http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/>

COIN-OR (2012) Computational Infrastructure for Operations Research: open source for the operations research community. <http://www.coin-or.org/>

GUROBI (2012) Gurobi Optimizer: State of the Art Mathematical Programming Solver. <http://www.gurobi.com>

INNOTRACK (2010) INNOTRACK reports. <http://www.innotrack.net/Reports>

LocalSolver (2012) LocalSolver: The first math programming solver based on local search. <http://www.localsolver.com/>

RailML (2012) RailML: the XML-Interface for railway applications. <http://www.railml.org//index.php/home.html>

RTS Solutions (2012) Work Infrastructure Management System (WIMS) datasheet. http://www.rts-solutions.net/images/datasheets/RTS_Rail_Datasheets_WIMS.pdf

SOG5 (2012) SOG 5, <http://www.ivembh.de/sog/programm.en.htm>. Last visited December 2012.



TILOS (2012) TILOS Overview: Project Management with Time Location Diagram, http://www.linearproject.com/tilos_overview.html. Last visited December 2012

XPRESS (2012) FICO® Xpress Optimization Suite.
<http://www.fico.com/en/products/dmtools/pages/fico-xpress-optimization-suite.aspx>

Reference documents

UIC Leaflet 406, *Capacity* (2004)

AUTOMAIN working documents

All these documents are available on AUTOMAIN website.
<http://www.automain.eu/documents>

Eurodecision (2012) *TWIP optimisation model*. April 2012.

Luiten T. (2012) *A more precise quantification of the different AUTOMAIN objectives*. Version 2.0.

Luiten T. (2012) *Work package results: datasheets to be used by WP5*.

Luiten T., U. Juntti, F. Sourd, C. Roberts (2012), *Description of demonstration scenarios and evaluation criteria*. AUTOMAIN D1.2.

WP5 (2012) *Problem Definition Document*. Version 1.6.

WP5(2012) *Problem Solution Document*. Version 6.3.



Appendix 2: Reference data

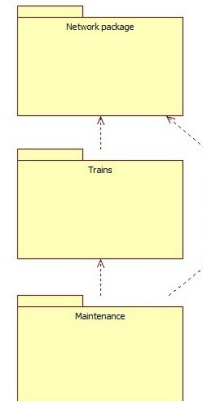
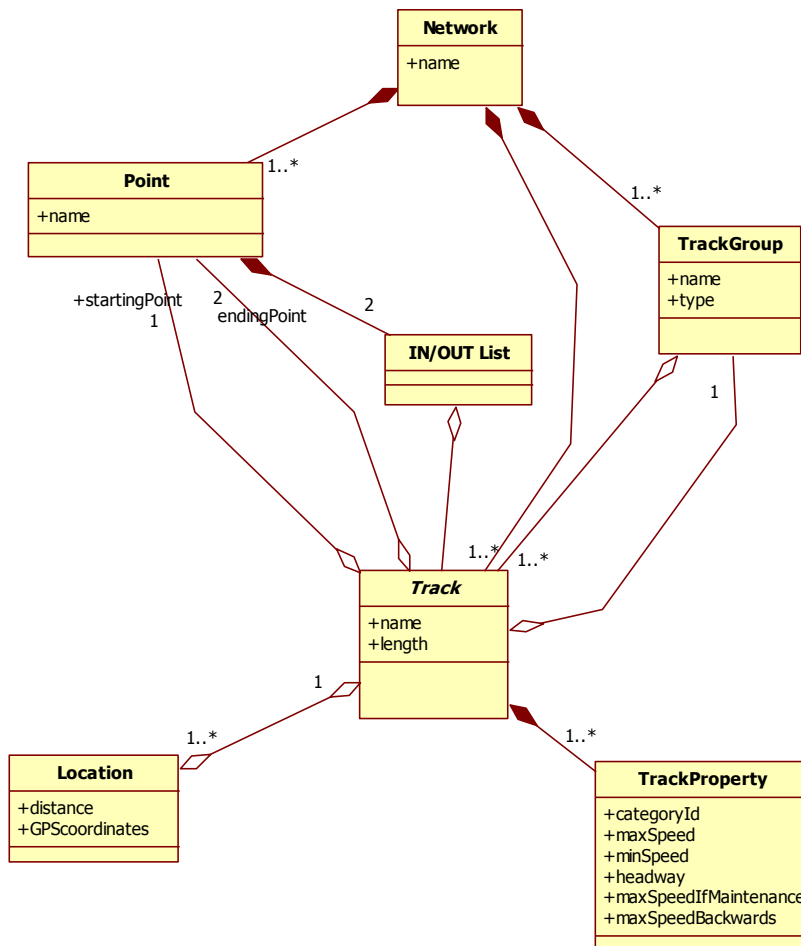
Global overview

The reference data consist of three main packages:

- the network,
- the trains,
- the maintenance operations.

We detail each of these packages.

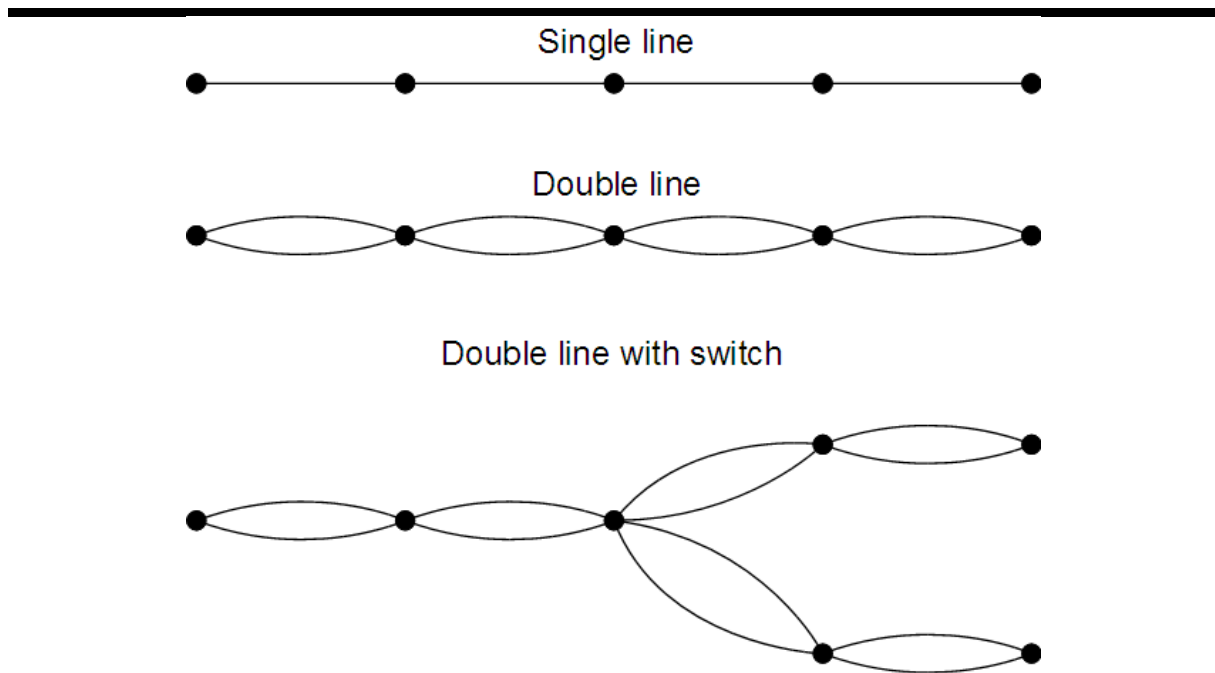
Network



The Network package contains two main objects: **Track** and **Point** objects. These objects are the respective equivalents of edge and vertex in the classical graph theory.

A **Track** physically corresponds to two parallel rails over a certain distance. It allows only one train in one direction at a time and it must be distinguished from a “railway line” which can allow multiple trains simultaneously depending on the number of parallel tracks.

A **Point** is a point in space, linking **Tracks** together.



The figure above shows how different types of railway configuration are represented by Tracks (curved lines) and Points (points at the extremities of the lines).

A **Location** can represent any place on a track and is specified by:

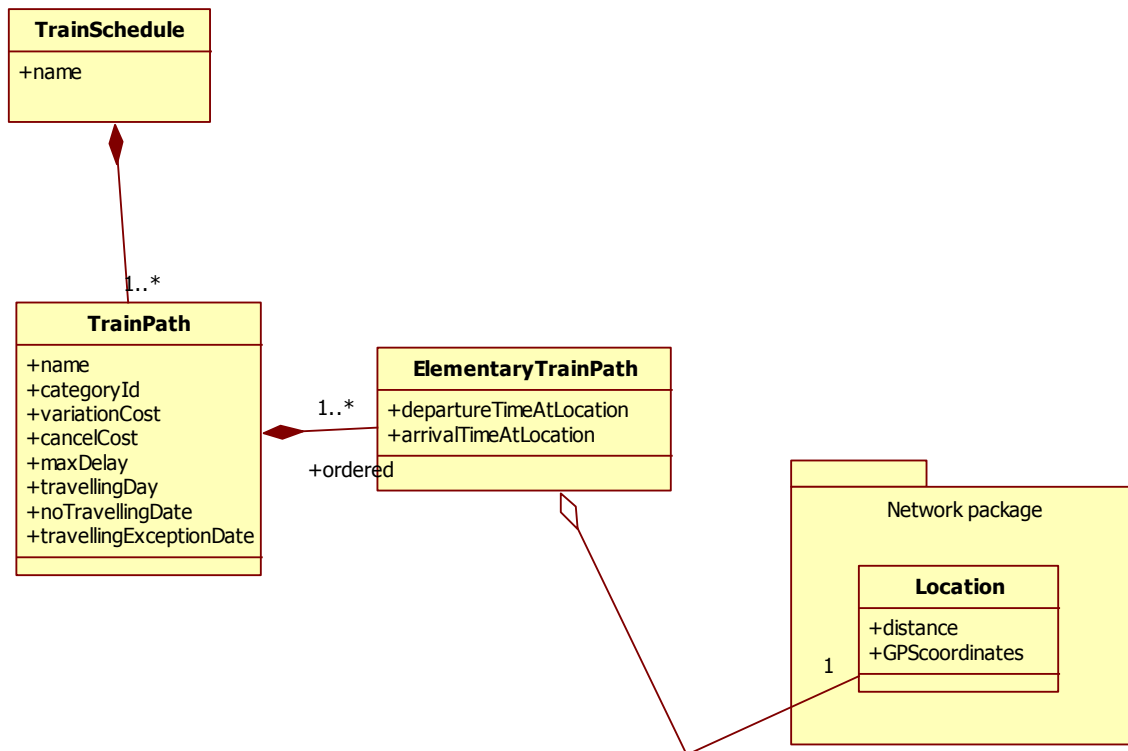
- The related **Track**
- The distance from the starting point of the **Track** to the location.

A **TrackGroup** represents a number of parallel tracks that need to be grouped for certain reasons. A TrackGroup corresponds to a portion of “railway line”.

The **TrackProperty** object represents all the properties of a **Track** for a certain train category in terms of maximum speed allowed.

Train Grid

Train Grid package establishes a simple yet complete model of a 24 h train schedule grid. It is only dependent on the **Network** package (specified earlier).



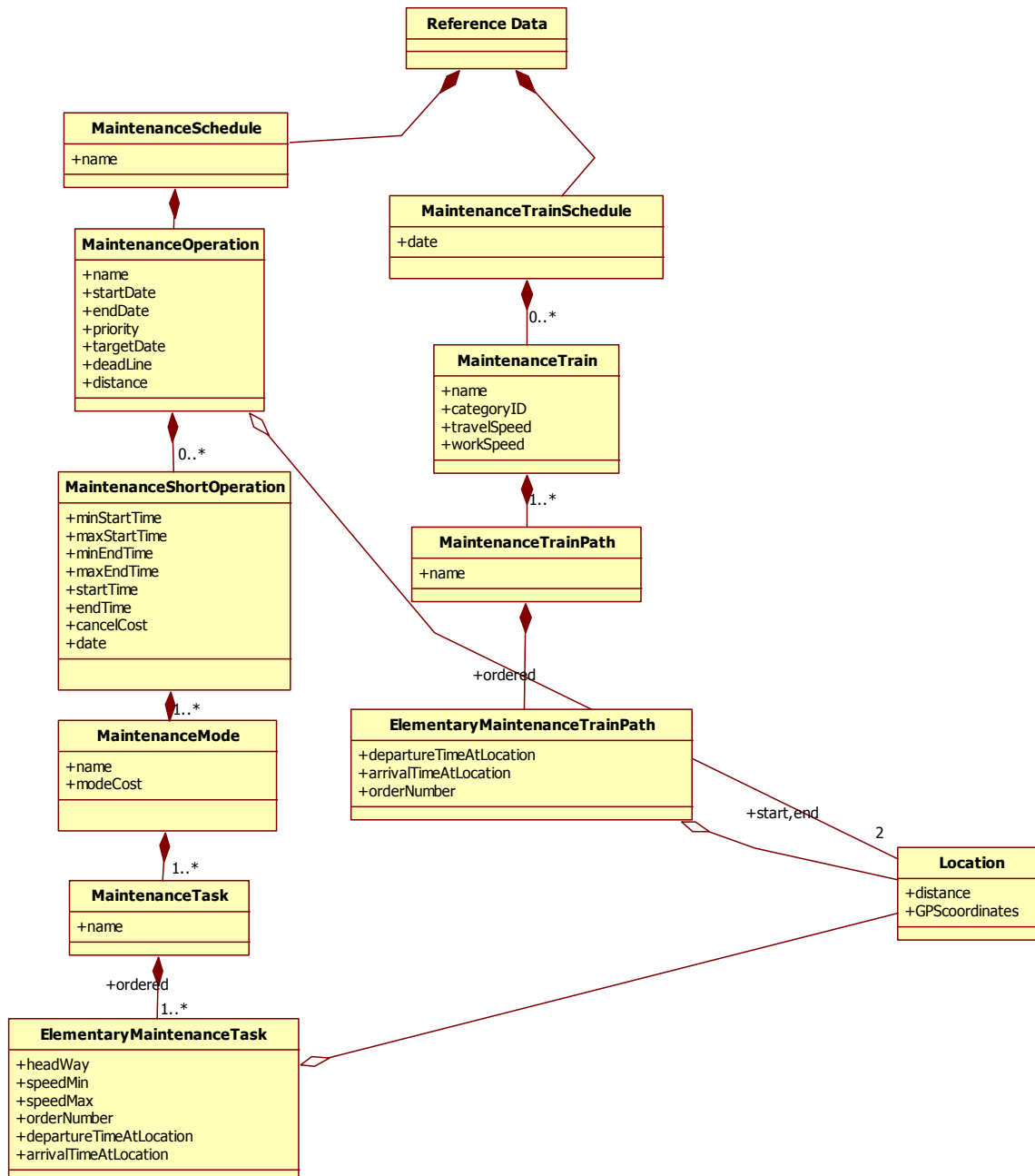
A **TrainSchedule** contains a list of **TrainPath** objects.

A **TrainPath** object contains an ordered list of **ElementaryTrainPath** objects, thus defining its path.

The **departureTimeFromLocation** and **arrivalTimeAtLocation** attributes are the time the train arrives at and departs from the specified location. A concrete example of how a train path is described is provided in Appendix A.

The **variationCost** attribute in the **TrainPath** object specifies the (linear) cost of a train being delayed. The **maxDelay** attribute determines the maximum delay for a train before cancelling it.

Maintenance data



A **MaintenanceSchedule** contains the maintenance operations (**MaintenanceOperation**) to be placed. A **MaintenanceOperation** can be performed in different ways, called modes (**MaintenanceMode**). Since there is a need for considering operations that have a small duration, a **MaintenanceShortOperation** class exists in between those two. A **MaintenanceMode** contains the maintenance tasks. Two maintenance tasks may not be adjacent to each other. This is why a **MaintenanceMode** can have several **MaintenanceTasks**. The smallest element is called **ElementaryMaintenanceTask**, which defines the time spent on a **Location**, since a task can be performed on a series of tracks.



The **ElementaryMaintenanceTask** of a single **MaintenanceTask** are adjacent and follow a strong precedence constraint (endTime of element i equals startTime of element i+1).

A **MaintenanceTrainSchedule** contains a list of available **MaintenanceTrains**. These trains are not fixed on the schedule since this information will be part of the TWIP module output. Each **MaintenanceTrain** is composed of **MaintenanceTrainPaths**. Each **MaintenanceTrainPath** contains an ordered list of **ElementaryMaintenanceTrainPaths**. **ElementaryMaintenanceTrainPath** indicates the path of the maintenance train. This path is described by **departureTimeFromLocation** and **arrivalTimeAtLocation** attributes, which correspond to the time the maintenance train arrives at and departs from the specified location.



Appendix 3: Data exchange between modules

In this section, the data exchange between the three modules LTP, TWG and TWIP is presented and described. It should be noticed that here only the data which is effectively exchanged is presented, i.e. it is assumed that all three modules have permanent read access to the reference data. However, only LTP will have write access to it.

Note : these interfaces are subject to modification if some improvement can be brought about during the test phase.

Data exchange between LTP and TWG

Use case : minimise the cost and duration of a maintenance operation along with the travel required to move the machine towards the operation location.

- Arguments :
 - *MaintenanceOperation* operation* : the maintenance operation to be performed
 - *MaintenanceTrain* machine* : the machine assigned to perform the operation
 - *ptime machineAvailableDate* : the earliest date at which the machine can start its move towards the considered operation
 - *ptime latestEndDate* : the latest date at which the operation must be finished
 - *ptime targetEndDate* : the ideal date at which the operation should be finished
 - *Location* initialLocation* : the place where the machine is initially located at *machineAvailableDate*. The machine has to move from this location to the place where *operation* has to be performed.
- Return value : a vector a solutions, each of them defined by :
 - *ptime beginDate* : the date when the machine starts its move towards the operation
 - *ptime endDate* : the date when the machine finishes the maintenance operation
 - *vector<TrainPath*> cancelledTrains* : the trains cancelled in the solution
 - *map<TrainPath*, time_duration> delayedTrains* : the trains delayed in the solution, along with the maximum delay at some point of the path, compared to the original path
 - *vector<MaintenanceShortOperation*> operationSplit* : the splitting of the maintenance operation into shorter, uninterrupted possessions.
 - *MaintenanceTrainPath* machineTravel* : the train path of the machine from *initialLocation* and during the maintenance operation.
- C++ interface :
 - Input data constructor (class *TwgMinOpCostOutput*) :
TwgMinOpCostInput(MaintenanceOperation operation,*
MaintenanceTrain machine,*



-
- boost::posix_time::ptime machineAvailableDate,*
 - boost::posix_time::ptime latestEndDate,*
 - boost::posix_time::ptime targetEndDate,*
 - Location* initialLocation);*
 - TWG optimisation call :
 - std::vector<TwgMinOpCostOutput*>*
 - optimizeOperation(TwgMinOpCostInput* input, bool callTWIP=true);*
 - Output data getters (class TwgMinOpCostOutput) :
 - boost::posix_time::ptime beginDate() const;*
 - boost::posix_time::ptime endDate();*
 - double cost() const;*
 - std::vector<TrainPath*> cancelledTrains() const;*
 - std::map<TrainPath*, boost::posix_time::time_duration> delayedTrains()*
 - const; // Trains delayed and the length of the delay*
 - std::vector<MaintenanceShortOperation*> operationSplit() const;*
 - MaintenanceTrainPath* machineTravel() const;*

Data exchange between TWG and TWIP

Use case: minimise the cost of performing a maintenance operation.

- Arguments
 - MaintenanceShortOperation shortOperation
 - MaintenanceTrain* maintenanceTrain
 - `const` MaintenanceOperation* maintenanceOperation,
 - `const` ReferenceData* referenceData,
 - `int&` cost,
 - `std::vector<TrainPath*>&` trainPathDelayed,
 - `std::vector<TrainPath*>&` canceledTrain

- Return value
 - void

- C++ Interface:

```
void Twip::addPair(
    MaintenanceShortOperation* shortOperation,
    MaintenanceTrain* maintenanceTrain,
    const MaintenanceOperation* maintenanceOperation,
    const ReferenceData* referenceData,
    int& cost,
    std::vector<TrainPath*>& trainPathDelayed,
    std::vector<TrainPath*>& canceledTrain
)
```

trainPathDelayed contains the list of delayed trainPath



cancelledTrain contains the list of cancelled trainPath

cost contains the optimal cost of the insertion of the maintenance operation



Appendix 4: LTP sample run log

This appendix shows a sample of what a call to the executable (with parameter 'LTP' activated) produces. First, the program reads the data provided as an XML file, and displays some information about the input:

```
INFO data/refData1.xml validates data/refdataxml.xsd
INFO Reading data/refData1.xml...
INFO Nb depot = 428
INFO Nb location = 1099
INFO Nb point = 704
INFO Nb track = 1274
INFO Nb trackGroup = 788
INFO Nb train = 280
INFO Mean nb etp per train = 53
INFO Nb maintenance op = 200
INFO Total length of tracks (m) = 6721879
INFO File data/refData1.xml was read.
INFO *****
INFO Graph is connex : explored 704 points.
```

Then, the LTP module is called to generate an optimised long-term planning. As described previously in the document, the first step consists in quickly producing an estimated plan. The estimated plan for our data set is displayed below, showing for two machines (out of 20 for the complete output) which operations it should perform and when. The dates are provided as [d1, d2, d3] where d1 is the date when the train begins moving towards the operation from its previous location, d2 is the date when the machine actually begins working on the operation, and d3 is the date when the machine finishes the operation :

```
INFO Calling LTP...
ALGO Starting planning build...
ALGO Planning built ok.
Initial estimated plan =
- GrindingTrain_5 : Grinding : (GrindingOp_111 (from Track_1004@4676 to
Track_1004@2455):2221m:[2012-Dec-10, 2012-Dec-12, 2012-Dec-18]), (GrindingOp_134 (from
Track_15@3151 to Track_15@689):2462m:[2012-Dec-18, 2012-Dec-20, 2012-Dec-27]), (GrindingOp_47 (from
Track_331@122 to Track_331@869):747m:[2012-Dec-27, 2012-Dec-29, 2012-Dec-31]), (GrindingOp_92 (from
Track_593@2532 to Track_593@3669):1137m:[2012-Dec-31, 2013-Jan-02, 2013-Jan-05]), (GrindingOp_41
(from Track_688@3237 to Track_688@387):2850m:[2013-Oct-21, 2013-Oct-23, 2013-Oct-31]),
- GrindingTrain_6 : ...
- GrindingTrain_7 : ...,
- GrindingTrain_8 : ...
- GrindingTrain_9 : ...
- Switch_inspectionTrain_10 : ...
- Switch_inspectionTrain_11 : ...
- Switch_inspectionTrain_12 : ...
- Switch_inspectionTrain_14 : ...
- TampingTrain_0 : Tamping : (TampingOp_90 (from Track_400@2458 to
Track_400@491):1967m:[2012-Dec-11, 2012-Dec-13, 2012-Dec-14]), (TampingOp_157 (from Track_960@3899
to Track_960@7599):3700m:[2012-Dec-14, 2012-Dec-16, 2012-Dec-18]), (TampingOp_177 (from
```



Track_357@38 to Track_357@419):381m:[2012-Dec-18, 2012-Dec-20, 2012-Dec-21]], (TampingOp_83 (from Track_727@984 to Track_727@1104):120m:[2012-Dec-21, 2012-Dec-23, 2012-Dec-24]), (TampingOp_153 (from Track_164@678 to Track_164@4421):3743m:[2012-Dec-24, 2012-Dec-26, 2012-Dec-28]), (TampingOp_103 (from Track_917@643 to Track_917@263):380m:[2012-Dec-28, 2012-Dec-30, 2012-Dec-31]), (TampingOp_110 (from Track_1246@1048 to Track_1246@82):966m:[2012-Dec-31, 2013-Jan-02, 2013-Jan-03]), (TampingOp_20 (from Track_841@74 to Track_841@142):68m:[2013-Jan-03, 2013-Jan-05, 2013-Jan-06]), (TampingOp_145 (from Track_523@259 to Track_523@25):234m:[2013-Jan-06, 2013-Jan-08, 2013-Jan-09]), (TampingOp_75 (from Track_287@13 to Track_287@2271):2258m:[2013-Jan-09, 2013-Jan-11, 2013-Jan-13]), (TampingOp_72 (from Track_42@1641 to Track_42@223):1418m:[2013-Jan-13, 2013-Jan-15, 2013-Jan-16]), (TampingOp_192 (from Track_191@118 to Track_191@541):423m:[2013-Jan-16, 2013-Jan-18, 2013-Jan-19]), (TampingOp_8 (from Track_438@883 to Track_438@168):715m:[2013-Jan-19, 2013-Jan-21, 2013-Jan-22]), (TampingOp_186 (from Track_1241@749 to Track_1241@1175):426m:[2013-Jan-27, 2013-Jan-29, 2013-Jan-30]), (TampingOp_127 (from Track_714@1064 to Track_714@278):786m:[2013-Jan-31, 2013-Feb-02, 2013-Feb-03]), (TampingOp_23 (from Track_1105@5638 to Track_1105@808):4830m:[2013-Feb-03, 2013-Feb-05, 2013-Feb-08]), (TampingOp_44 (from Track_17@6151 to Track_17@11486):5335m:[2013-Feb-09, 2013-Feb-11, 2013-Feb-14]), (TampingOp_2 (from Track_907@984 to Track_907@55):929m:[2013-Feb-16, 2013-Feb-18, 2013-Feb-19]), (TampingOp_146 (from Track_82@182 to Track_82@781):599m:[2013-Feb-22, 2013-Feb-24, 2013-Feb-25]), (TampingOp_4 (from Track_1016@1366 to Track_1016@10):1356m:[2013-Feb-25, 2013-Feb-27, 2013-Feb-28]), (TampingOp_7 (from Track_321@4721 to Track_321@3527):1194m:[2013-Mar-02, 2013-Mar-04, 2013-Mar-05]), (TampingOp_124 (from Track_1146@2031 to Track_1146@862):1169m:[2013-Mar-05, 2013-Mar-07, 2013-Mar-08]), (TampingOp_194 (from Track_366@7544 to Track_366@1672):5872m:[2013-Mar-09, 2013-Mar-11, 2013-Mar-14]), (TampingOp_21 (from Track_656@15 to Track_656@41):26m:[2013-Mar-16, 2013-Mar-18, 2013-Mar-19]), (TampingOp_42 (from Track_445@4 to Track_445@2151):2147m:[2013-Mar-23, 2013-Mar-25, 2013-Mar-27]), (TampingOp_117 (from Track_625@5403 to Track_625@1537):3866m:[2013-Apr-06, 2013-Apr-08, 2013-Apr-10]), (TampingOp_73 (from Track_617@1402 to Track_617@2490):1088m:[2013-Apr-20, 2013-Apr-22, 2013-Apr-23]), (TampingOp_6 (from Track_715@2315 to Track_715@4527):2212m:[2013-Apr-23, 2013-Apr-25, 2013-Apr-27]), (TampingOp_9 (from Track_1000@155 to Track_1000@1281):1126m:[2013-Apr-27, 2013-Apr-29, 2013-Apr-30]), (TampingOp_45 (from Track_967@4175 to Track_967@5115):940m:[2013-May-05, 2013-May-07, 2013-May-08]), (TampingOp_46 (from Track_679@34 to Track_679@2870):2836m:[2013-May-12, 2013-May-14, 2013-May-16]), (TampingOp_18 (from Track_1183@1740 to Track_1183@3879):2139m:[2013-Aug-11, 2013-Aug-13, 2013-Aug-15]), (TampingOp_69 (from Track_473@114 to Track_473@6570):6456m:[2013-Aug-24, 2013-Aug-26, 2013-Aug-30]), (TampingOp_165 (from Track_1196@39 to Track_1196@665):626m:[2013-Aug-31, 2013-Sep-02, 2013-Sep-03]), (TampingOp_13 (from Track_100@4738 to Track_100@20):4718m:[2013-Sep-03, 2013-Sep-05, 2013-Sep-08]), (TampingOp_193 (from Track_764@7026 to Track_764@1905):5121m:[2013-Sep-09, 2013-Sep-11, 2013-Sep-14]), (TampingOp_33 (from Track_367@3307 to Track_367@170):3137m:[2013-Sep-14, 2013-Sep-16, 2013-Sep-18]), (TampingOp_129 (from Track_1036@10084 to Track_1036@332):9752m:[2013-Sep-21, 2013-Sep-23, 2013-Sep-28]), (TampingOp_132 (from Track_68@2038 to Track_68@3251):1213m:[2013-Oct-06, 2013-Oct-08, 2013-Oct-09]), (TampingOp_48 (from Track_502@1202 to Track_502@77):1125m:[2013-Oct-19, 2013-Oct-21, 2013-Oct-22]), (TampingOp_16 (from Track_255@387 to Track_255@211):176m:[2013-Oct-26, 2013-Oct-28, 2013-Oct-29]), (TampingOp_24 (from Track_978@722 to Track_978@13114):12392m:[2013-Nov-01, 2013-Nov-03, 2013-Nov-10]), (TampingOp_50 (from Track_830@261 to Track_830@2742):2481m:[2013-Nov-10, 2013-Nov-12, 2013-Nov-14]), (TampingOp_28 (from Track_440@130 to Track_440@540):410m:[2013-Nov-16, 2013-Nov-18, 2013-Nov-19]),

- TampingTrain_1 : ...
- TampingTrain_2 : ...
- TampingTrain_3 : ...
- TampingTrain_4 : ...

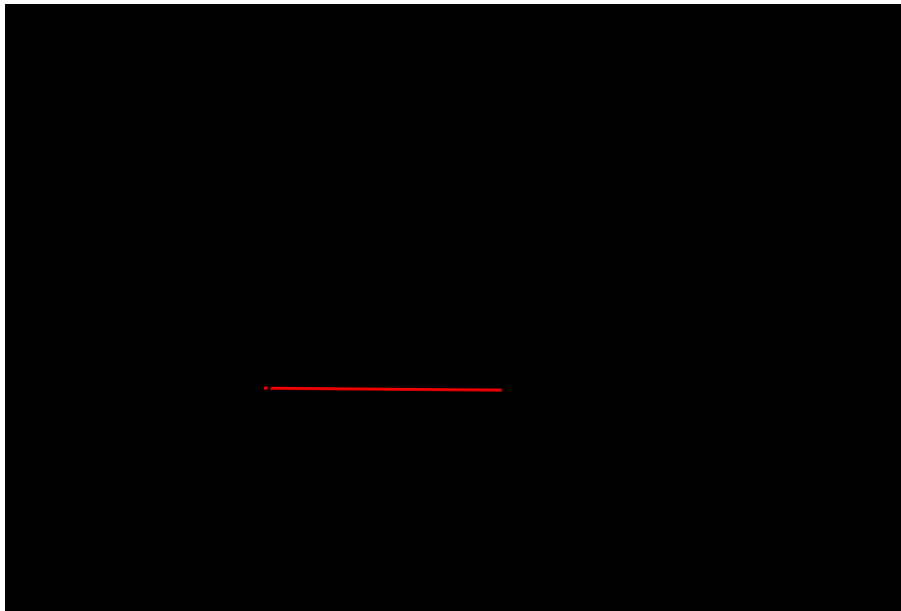
The next phases of the algorithm involve calls to TWG and TWIP. As mentioned above, the integration of the three modules is not yet available, so this work is still in progress.



Appendix 5: TWG experimental results

In this appendix we present the results generated by TWG when run on some example data. We perform two tests, the first one testing only the ability of TWG to find shortest path between couples of locations within a railway network, the second one testing also the ability to divide an operation into several short operations.

Consider the network of the figure below. The letters are node names, and the numbers are the track lengths. Node e is the only depot of the network. In the following, we express locations within the network in the form (ab, 5), meaning the location within the track that goes from a to b, 5 kilometres away from a.



In the first test, TWG is requested to find the shortest path from (ab, 0) to (ge, 5), from (ge, 2) to (bc, 8), from (cd, 1) to (cd, 10), from (bc, 2) to (cb, 8) and from (ab, 10) to (bc, 10).

The output produced is shown below:

TWG call n 1

Shortest path from (ab, 0) a to (ge, 5)

(ab, 0) (ab, 10) (bf, 5) (fg, 5) (ge, 5)

TWG call n 2

Shortest path from (ge, 2) a to (bc, 8)

(ge, 2) (ge, 0) (fg, 0) (bf, 0) (bc, 8)

TWG call n 3

Shortest path from (cd, 1) a to (cd, 10)

(cd, 10) (cd, 10)

TWG call n 4



Shortest path from (bc, 2) a to (cb, 8)

(bc, 2) (bc, 10) (cb, 8)

TWG call n 5

Shortest path from (ab, 10) a to (bc, 10)

(bc, 0) (bc, 10)

It can be verified that in each case the shortest path is found.

In the second test, the task is twofold: on the one hand, a tamping operation comprising the whole track bc has to be divided into three short operations, and the whole routing consisting of the following segments:

1. from the starting location at node a to the start point of the first short operation
2. from the start point of the first short operation to the end point of the first short operation
3. from the end point of the first short operation to the depot at node e
4. from the depot at node e to the start point of the second operation
5. from the start point of the second short operation to the end point of the second short operation
6. from the end point of the second short operation to the depot at node e
7. from the depot at node e to the start point of the third operation
8. from the start point of the third short operation to the end point of the third short operation

The output produced by TWG is as shown below:

Optimal operation split has 3 short-ops, namely:

Short op 0 has 1 tasks, namely:

Task test operation. Task Number 1:

start location: (bc, 0)

end location: bc, 3350)

Has 0 elementary tasks, namely:

Short op 1 has 1 tasks, namely:

Task test operation. Task Number 1:

start location: (bc, 3350)

end location: bc, 6700)

Has 0 elementary tasks, namely:

Short op 2 has 1 tasks, namely:

Task test operation. Task Number 1:

start location: (bc, 6700)

end location: bc, 10000)

The corresponding routing is

(hi, 5) (hi, 0) (ch, 0) (bc, 0) (bc, 3350) (bc, 0) (bf, 5000) (gd, 0) (ge, 5000) (ge, 5000) (ge, 0) (fg, 0) (bf, 0) (bc, 3350) (bc, 6700) (bc, 0) (bf, 5000) (gd, 0) (ge, 5000) (ge, 5000) (ge, 0) (fg, 0) (bf, 0) (bc, 6700) (cd, 0) (cd, 10000) (gd, 0) (ge, 5000)



It can be seen that the operation is divided into three segments 0-3350, 3350-6700 and 6700-10000 (expressed in meters within segment bc). It can also be proven that the found route is the shortest containing all the points enumerated above.

The results shown are produced on a fraction of a second in a standard desktop computer.



Appendix 6: TWIP experimental results

In order to test the implemented algorithm performance, we have randomly generated data with different sizes and difficulty levels.

Instance	Description
A1	2 commercial trains, 1 maintenance train, 2 maintenance tasks
A2	4 commercial trains, 2 maintenance tasks having respectively 4 and 5 modes.
A3	3 commercial trains, 2 maintenance tasks having respectively 4 and 7 modes
A4	2 commercial trains (on the same track), 1 maintenance train, 1 maintenance task having 2 modes.
A5	Similar to A4, but with 1 additional commercial train
A6	2 possible paths, 10 commercial trains, 1 maintenance tasks per path with respectively 2 and 3 modes.
A7	Similar to A2, but with different allowed deviations and makespans.

For each instance, the algorithm generates a mixed-integer model (MIP) whose size depends on the number of trains and operations.

Instance	# variables	# Integer Variables	# Constraints
A1	242	150	495
A2	1692	1390	4147
A3	2022	1694	5142
A4	2319	2183	5129
A5	3482	3302	7800
A6	19461	18689	46795
A7	4092	3766	9631

Experimental conditions & results

The results were performed on a DELL computer with an Intel Core i5-2520M CPU 2.50 GHz and 4 GB of RAM.

The LP solver used is CPLEX 12.4, which is quite recent.

We have used it with the default mode, using 4 threads. The threads are helpful to parallelise the Branch & Bound algorithm.

Instance	PL (seconds)
A1	0.25



A2	4.3
A3	1.9
A4	4.3
A5	281.6
A6	7544
A7	55

Other experiments

We have also explored another mode, using only 1 thread. In this mode there was no possibility to parallelise the Branch & Bound.

The result is that for large instances, the results with 4 threads are far better. Strangely, for some small instances, using only 1 thread should be more efficient. A possible reason for that could be that modern CPU can overclock the frequency when only one thread is use.



Appendix 7: Train path description : XML sample

In this section we provide an XML example of a train path description. The first sample corresponds to an original train path : times of arrival and departure are provided for each node of the network. The second sample shows the same train path which has been modified due to maintenance on the tracks : new Locations are inserted to provide a more detailed description around the maintenance operation.

Sample XML - original data : a train departs from location 1978 at 6:43 (arrival time is unnecessary for the first ElementaryTrainPath, likewise for the departure time of the last ElementaryTrainPath)

```
=====
Sample XML - original data : a train departs from location 1978 at 6:43 (arrival time is unnecessary for
the first ElementaryTrainPath, likewise for the departure time of the last ElementaryTrainPath)
=====

<TrainPath maxDelay="120" cancelCost="100" variationCost="25" categoryId="freight"
name="TrainPath_1" _xmlId="TrainPath_1968">
  <elementaryTrainPathList>
    <ElementaryTrainPath departureTimeFromLocation="6:43:00" arrivalTimeAtLocation="6:43:00"
_xmlId="ElementaryTrainPath_1977">
      <location>
        <Location distance="0" track="Track_82" _xmlId="Location_1978">
          <coordinate>
            <GpsCoord elevation="0.0" longitude="0.9361202688942791" latitude="49.39769899524623"
_xmlId="GpsCoord_1979"/>
          </coordinate>
        </Location>
      </location>
    </ElementaryTrainPath>
    <ElementaryTrainPath departureTimeFromLocation="6:44:00" arrivalTimeAtLocation="6:44:00"
_xmlId="ElementaryTrainPath_1980">
      <location>
        <Location distance="0" track="Track_319" _xmlId="Location_1981">
          <coordinate>
            <GpsCoord elevation="0.0" longitude="0.9483395212244874" latitude="49.39515213479969"
_xmlId="GpsCoord_1982"/>
          </coordinate>
        </Location>
      </location>
    </ElementaryTrainPath>
    <ElementaryTrainPath departureTimeFromLocation="6:45:00" arrivalTimeAtLocation="6:45:00"
_xmlId="ElementaryTrainPath_1983">
      <location>
        <Location distance="0" track="Track_958" _xmlId="Location_1984">
          <coordinate>
            <GpsCoord elevation="0.0" longitude="0.961985094615903" latitude="49.39752826617542"
_xmlId="GpsCoord_1985"/>
          </coordinate>
        </Location>
      </location>
    </ElementaryTrainPath>
    <ElementaryTrainPath departureTimeFromLocation="6:47:00" arrivalTimeAtLocation="6:47:00"
_xmlId="ElementaryTrainPath_1986">
      <location>
        <Location distance="0" track="Track_1478" _xmlId="Location_1987">
          <coordinate>
            <GpsCoord elevation="0.0" longitude="1.0195837407647188" latitude="49.41043716841177"
_xmlId="GpsCoord_1988"/>
          </coordinate>
        </Location>
      </location>
    </ElementaryTrainPath>
    <ElementaryTrainPath departureTimeFromLocation="6:49:00" arrivalTimeAtLocation="6:49:00"
_xmlId="ElementaryTrainPath_1989">
      <location>
        <Location distance="0" track="Track_548" _xmlId="Location_1990">
          <coordinate>
```



```
<GpsCoord elevation="0.0" longitude="1.0751552961679838" latitude="49.422625508711015"
_xmlId="GpsCoord_1991"/>
</coordinate>
</Location>
</location>
</ElementaryTrainPath>
</elementaryTrainPathList>
<travellingDay>
<WeekDay weekDayName="Monday" _xmlId="WeekDayJaxb_1969"/>
<WeekDay weekDayName="Tuesday" _xmlId="WeekDayJaxb_1970"/>
<WeekDay weekDayName="Wednesday" _xmlId="WeekDayJaxb_1971"/>
<WeekDay weekDayName="Thursday" _xmlId="WeekDayJaxb_1972"/>
<WeekDay weekDayName="Friday" _xmlId="WeekDayJaxb_1973"/>
<WeekDay weekDayName="Saturday" _xmlId="WeekDayJaxb_1974"/>
</travellingDay>
<noTravellingDate>
<Date date="2012-Dec-25" _xmlId="DateJaxb_1975"/>
</noTravellingDate>
<travellingDate>
<Date date="2012-Dec-23" _xmlId="DateJaxb_1976"/>
</travellingDate>
</TrainPath>
```

=====

Sample of reference data during the run, with a modified train path - see the last two ElementaryTrainPath elements : there are works on track 1478, which force the train to slow down and arrive at destination a bit later, with an additional location introduced 4,102 km away from Location 1987

=====

```
<TrainPath maxDelay="120" cancelCost="100" variationCost="25" categoryId="freight"
name="TrainPath_1" _xmlId="TrainPath_1968">
<elementaryTrainPathList>
<ElementaryTrainPath departureTimeFromLocation="6:43:00" arrivalTimeAtLocation="6:43:00"
_xmlId="ElementaryTrainPath_1977">
<location>
<Location distance="0" track="Track_82" _xmlId="Location_1978">
<coordinate>
<GpsCoord elevation="0.0" longitude="0.9361202688942791" latitude="49.39769899524623"
_xmlId="GpsCoord_1979"/>
</coordinate>
</Location>
</location>
</ElementaryTrainPath>
<ElementaryTrainPath departureTimeFromLocation="6:44:00" arrivalTimeAtLocation="6:44:00"
_xmlId="ElementaryTrainPath_1980">
<location>
<Location distance="0" track="Track_319" _xmlId="Location_1981">
<coordinate>
<GpsCoord elevation="0.0" longitude="0.9483395212244874" latitude="49.39515213479969"
_xmlId="GpsCoord_1982"/>
</coordinate>
</Location>
</location>
</ElementaryTrainPath>
<ElementaryTrainPath departureTimeFromLocation="6:45:00" arrivalTimeAtLocation="6:45:00"
_xmlId="ElementaryTrainPath_1983">
<location>
<Location distance="0" track="Track_958" _xmlId="Location_1984">
<coordinate>
<GpsCoord elevation="0.0" longitude="0.961985094615903" latitude="49.39752826617542"
_xmlId="GpsCoord_1985"/>
</coordinate>
</Location>
</location>
</ElementaryTrainPath>
<ElementaryTrainPath departureTimeFromLocation="6:47:00" arrivalTimeAtLocation="6:47:00"
_xmlId="ElementaryTrainPath_1986">
<location>
<Location distance="0" track="Track_1478" _xmlId="Location_1987">
<coordinate>
```



```
<GpsCoord elevation="0.0" longitude="1.0195837407647188" latitude="49.41043716841177"
_xmlId="GpsCoord_1988"/>
</coordinate>
</Location>
</location>
</ElementaryTrainPath>
<ElementaryTrainPath departureTimeFromLocation="6:49:10" arrivalTimeAtLocation="6:49:10"
_xmlId="ElementaryTrainPath_2300">
<location>
<Location distance="4102" track="Track_1478" _xmlId="Location_2000">
<coordinate>
<GpsCoord elevation="0.0" longitude="1.013" latitude="49.415" _xmlId="GpsCoord_2200"/>
</coordinate>
</Location>
</location>
</ElementaryTrainPath>
<ElementaryTrainPath departureTimeFromLocation="6:53:00" arrivalTimeAtLocation="6:53:00"
_xmlId="ElementaryTrainPath_1989">
<location>
<Location distance="0" track="Track_548" _xmlId="Location_1990">
<coordinate>
<GpsCoord elevation="0.0" longitude="1.0751552961679838" latitude="49.422625508711015"
_xmlId="GpsCoord_1991"/>
</coordinate>
</Location>
</location>
</ElementaryTrainPath>
</elementaryTrainPathList>
<travellingDay>
<WeekDay weekDayName="Monday" _xmlId="WeekDayJaxb_1969"/>
<WeekDay weekDayName="Tuesday" _xmlId="WeekDayJaxb_1970"/>
<WeekDay weekDayName="Wednesday" _xmlId="WeekDayJaxb_1971"/>
<WeekDay weekDayName="Thursday" _xmlId="WeekDayJaxb_1972"/>
<WeekDay weekDayName="Friday" _xmlId="WeekDayJaxb_1973"/>
<WeekDay weekDayName="Saturday" _xmlId="WeekDayJaxb_1974"/>
</travellingDay>
<noTravellingDate>
<Date date="2012-Dec-25" _xmlId="DateJaxb_1975"/>
</noTravellingDate>
<travellingDate>
<Date date="2012-Dec-23" _xmlId="DateJaxb_1976"/>
</travellingDate>
</TrainPath>
```



Appendix 7: XSD Schema

In this section we provide the xsd schema defining the XML format in which the reference data must be written to be read by the software prototype.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
  jaxb:version="2.1"
  xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
  jaxb:extensionBindingPrefixes="xjc">

  <!-- START JAXB configuration -->
  <xs:annotation>
    <xs:appinfo>
      <jaxb:globalBindings>
        <!-- Will defines a common ancestor for all generated objects. -->
        <!-- Class used allow automatic serial generation, as JAXBElement reference creation
helpers -->
        <xjc:superClass name="com.eurodecision.jaxb.GenDmXmlObject"/>

        <!-- Defines adapters to convert cpp strings to/from date -->
        <jaxb:javaType name="java.util.Date"
          xmlType="dateCPP"

parseMethod="com.eurodecision.jaxb.types.EDJaxbCppDateTypesConverter.cppStringToDate"

printMethod="com.eurodecision.jaxb.types.EDJaxbCppDateTypesConverter.dateToCppString" />
        <jaxb:javaType name="java.util.Date"
          xmlType="dateTimeCPP"

parseMethod="com.eurodecision.jaxb.types.EDJaxbCppDateTypesConverter.cppStringToDateTime"

printMethod="com.eurodecision.jaxb.types.EDJaxbCppDateTypesConverter.dateTimeToCppString" />
        <jaxb:javaType name="java.util.Date"
          xmlType="durationCPP"

parseMethod="com.eurodecision.jaxb.types.EDJaxbCppDateTypesConverter.cppStringToTime"

printMethod="com.eurodecision.jaxb.types.EDJaxbCppDateTypesConverter.timeToCppString" />
        <jaxb:javaType name="com.eurodecision.jaxb.types.Period"
          xmlType="periodCPP"

parseMethod="com.eurodecision.jaxb.types.EDJaxbCppDateTypesConverter.cppStringToPeriod"

printMethod="com.eurodecision.jaxb.types.EDJaxbCppDateTypesConverter.periodToCppString" />
        <jaxb:javaType name="boolean"
          xmlType="xs:boolean"

parseMethod="com.eurodecision.jaxb.types.EDJaxbCppDateTypesConverter.cppBoolToJavaBoolean"

printMethod="com.eurodecision.jaxb.types.EDJaxbCppDateTypesConverter.javaBooleanToCppBool" />
        <jaxb:javaType name="com.sncf.automain.utils.WeekDay"
          xmlType="weekDayNameCPP"
          parseMethod="com.sncf.automain.utils.WeekDay.cppStringToJava"
          printMethod="com.sncf.automain.utils.WeekDay.javaToCppString" />
      </jaxb:globalBindings>

      <!-- Rename the attribute generated in SimpleRef type that will contain referenced objects
-->
      <jaxb:bindings node="//xs:attribute[@name='_xmlId' and @type='xs:IDREF']" >
        <jaxb:property name="objInstance"/>
      </jaxb:bindings>

    </xs:appinfo>
  </xs:annotation>

  <!--
  <!-- Will prefix all generated classes names -->
  <jaxb:schemaBindings>
```



```
<jaxb:nameXmlTransform>
  <jaxb:typeName prefix="Cpp"/>
  <jaxb:elementName prefix="Cpp"/>
</jaxb:nameXmlTransform>
</jaxb:schemaBindings>
-->

</xs:appinfo>
</xs:annotation>
<!-- END JAXB configuration -->

<!-- commons -->
<xs:complexType name="objectInstance">
  <xs:attribute name="_xmlId" type="xs:ID" use="required"/>
</xs:complexType>

<xs:complexType name="simpleRef">
  <xs:attribute name="_xmlId" type="xs:IDREF" use="required"/>
</xs:complexType>

<xs:simpleType name="char">
  <xs:restriction base="xs:string">
    <xs:length value="1"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="dateCPP">
  <xs:restriction base="xs:token">
    <xs:pattern value="[0-9]{4}-([0-9]{2}|[a-zA-Z]{3})-[0-9]{2}"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="dateTimeCPP">
  <xs:restriction base="xs:token">
    <xs:pattern value="[0-9]{4}-([0-9]{2}|[a-zA-Z]{3})-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}(\.[0-9]{1,6})?" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="durationCPP">
  <xs:restriction base="xs:token">
    <xs:pattern value="[0-9]+:[0-9]{2}:[0-9]{2}(\.[0-9]+)?" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="periodCPP">
  <xs:restriction base="xs:token">
    <xs:pattern value="\[[0-9]{4}-([0-9]{2}|[a-zA-Z]{3})-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}(\.[0-9]{1,6})?/[0-9]{4}-([0-9]{2}|[a-zA-Z]{3})-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}(\.[0-9]{1,6})?\]" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="weekDayNameCPP">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Monday" />
    <xs:enumeration value="Tuesday" />
    <xs:enumeration value="Wednesday" />
    <xs:enumeration value="Thursday" />
    <xs:enumeration value="Friday" />
    <xs:enumeration value="Saturday" />
    <xs:enumeration value="Sunday" />
  </xs:restriction>
</xs:simpleType>

<!-- XSD description of serialized model for class RefDataXml -->
<!-- serialized packages : [referencedata] -->

<!-- root and root containers -->
<xs:element name="root">
  <xs:complexType>
```



```
<xs:choice minOccurs="0" maxOccurs="unbounded">
  <xs:element name="AutomainSession" type="automainSession"/>
</xs:choice>
</xs:complexType>
</xs:element>

<!-- GenDmSessionClass (AutomainSession) -->
<xs:complexType name="automainSession">
  <xs:complexContent>
    <xs:extension base="objectInstance">
      <xs:sequence>
        <!-- GenDmAttribute (referenceData) [L19;C24->L19;C36] -->
        <xs:element name="referenceData" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="1">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="referenceData"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="ReferenceData" type="referenceData"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- GenDmClass (DatedItem) [L62;C3->L62;C7] -->
<xs:complexType name="datedItem">
  <xs:complexContent>
    <xs:extension base="objectInstance">
      <xs:attribute name="date" type="dateCPP" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- GenDmClass (ReferenceData) [L40;C2->L40;C6] -->
<xs:complexType name="referenceData">
  <xs:complexContent>
    <xs:extension base="objectInstance">
      <xs:sequence>
        <!-- GenDmAttribute (maintenanceSchedule) [L41;C31->L41;C49] -->
        <xs:element name="maintenanceSchedule" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="1">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="maintenanceSchedule"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="MaintenanceSchedule" type="maintenanceSchedule"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
        <!-- GenDmAttribute (maintenanceTrainSchedule) [L42;C36->L42;C59] -->
        <xs:element name="maintenanceTrainSchedule" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="1">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="maintenanceTrainSchedule"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="MaintenanceTrainSchedule" type="maintenanceTrainSchedule"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
        <!-- GenDmAttribute (network) [L44;C19->L44;C25] -->
        <xs:element name="network" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="1">
```



```
<xs:annotation>
  <xs:appinfo>
    <jaxb:property name="network"/>
  </xs:appinfo>
</xs:annotation>
<xs:element name="Network" type="network"/>
</xs:choice>
</xs:complexType>
</xs:element>
<!-- GenDmAttribute (trainSchedule) [L43;C25->L43;C37] -->
<xs:element name="trainSchedule" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="1">
      <xs:annotation>
        <xs:appinfo>
          <jaxb:property name="trainSchedule"/>
        </xs:appinfo>
      </xs:annotation>
      <xs:element name="TrainSchedule" type="trainSchedule"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- GenDmClass (MaintenanceSchedule) [L201;C3->L201;C7] -->
<xs:complexType name="maintenanceSchedule">
  <xs:complexContent>
    <xs:extension base="namedItem">
      <xs:sequence>
        <!-- GenDmAttribute (maintenanceOperationList) [L203;C37->L203;C60] -->
        <xs:element name="maintenanceOperationList" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="maintenanceOperationList"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="MaintenanceOperation" type="maintenanceOperation"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- GenDmClass (MaintenanceOperation) [L206;C3->L206;C7] -->
<xs:complexType name="maintenanceOperation">
  <xs:complexContent>
    <xs:extension base="namedItem">
      <xs:sequence>
        <!-- GenDmAttribute (maintenanceShortOperationList) [L229;C42->L229;C70] -->
        <xs:element name="maintenanceShortOperationList" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="maintenanceShortOperationList"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="MaintenanceShortOperation" type="maintenanceShortOperation"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
  <!-- GenDmAttribute (startLocation) [L225;C21->L225;C33] -->
  <xs:element name="startLocation" minOccurs="0" maxOccurs="1">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="1">
        <xs:annotation>
```



```
<xs:appinfo>
  <jaxb:property name="startLocation"/>
</xs:appinfo>
</xs:annotation>
<xs:element name="Location" type="simpleRef"/>
</xs:choice>
</xs:complexType>
</xs:element>
<!-- GenDmAttribute (endLocation) [L227;C21->L227;C31] -->
<xs:element name="endLocation" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="1">
      <xs:annotation>
        <xs:appinfo>
          <jaxb:property name="endLocation"/>
        </xs:appinfo>
      </xs:annotation>
      <xs:element name="Location" type="simpleRef"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
<!-- GenDmAttribute (usedMaintenanceTrains) [L231;C37->L231;C57] -->
<xs:element name="usedMaintenanceTrains" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <jaxb:property name="usedMaintenanceTrains"/>
        </xs:appinfo>
      </xs:annotation>
      <xs:element name="MaintenanceTrain" type="simpleRef"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="priority" type="xs:double"/>
<xs:attribute name="categoryId" type="xs:string"/>
<xs:attribute name="targetDate" type="dateTimeCPP"/>
<xs:attribute name="deadLine" type="dateTimeCPP"/>
<xs:attribute name="distance" type="xs:double"/>
<xs:attribute name="startDate" type="dateTimeCPP"/>
<xs:attribute name="endDate" type="dateTimeCPP"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- GenDmClass (MaintenanceShortOperation) [L234;C3->L234;C7] -->
<xs:complexType name="maintenanceShortOperation">
  <xs:complexContent>
    <xs:extension base="datedItem">
      <xs:sequence>
        <!-- GenDmAttribute (maintenanceModeList) [L252;C32->L252;C50] -->
        <xs:element name="maintenanceModeList" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="maintenanceModeList"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="MaintenanceMode" type="maintenanceMode"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="minStartTime" type="dateTimeCPP" use="required"/>
      <xs:attribute name="maxStartTime" type="dateTimeCPP" use="required"/>
      <xs:attribute name="minEndTime" type="dateTimeCPP" use="required"/>
      <xs:attribute name="maxEndTime" type="dateTimeCPP" use="required"/>
      <xs:attribute name="cancelCost" type="xs:int" use="required"/>
      <xs:attribute name="startTime" type="dateTimeCPP"/>
      <xs:attribute name="endTime" type="dateTimeCPP"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```



```
</xs:complexContent>
</xs:complexType>

<!-- GenDmClass (MaintenanceMode) [L256;C3->L256;C7] -->
<xs:complexType name="maintenanceMode">
  <xs:complexContent>
    <xs:extension base="namedItem">
      <xs:sequence>
        <!-- GenDmAttribute (maintenanceTaskList) [L260;C32->L260;C50] -->
        <xs:element name="maintenanceTaskList" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="maintenanceTaskList"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="MaintenanceTask" type="maintenanceTask"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="modeCost" type="xs:int"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- GenDmClass (MaintenanceTask) [L264;C3->L264;C7] -->
<xs:complexType name="maintenanceTask">
  <xs:complexContent>
    <xs:extension base="namedItem">
      <xs:sequence>
        <!-- GenDmAttribute (elementaryMaintenanceTaskList) [L266;C42->L266;C70] -->
        <xs:element name="elementaryMaintenanceTaskList" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="elementaryMaintenanceTaskList"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="ElementaryMaintenanceTask" type="elementaryMaintenanceTask"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
        <!-- GenDmAttribute (locationStart) [L268;C21->L268;C33] -->
        <xs:element name="locationStart" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="1">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="locationStart"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="Location" type="simpleRef"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
        <!-- GenDmAttribute (locationEnd) [L270;C21->L270;C31] -->
        <xs:element name="locationEnd" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="1">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="locationEnd"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="Location" type="simpleRef"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```



```
</xs:complexContent>
</xs:complexType>

<!-- GenDmClass (ElementaryMaintenanceTask) [L274;C3->L274;C7] -->
<xs:complexType name="elementaryMaintenanceTask">
  <xs:complexContent>
    <xs:extension base="objectInstance">
      <xs:sequence>
        <!-- GenDmAttribute (location) [L288;C21->L288;C28] -->
        <xs:element name="location" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="1">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="location"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="Location" type="simpleRef"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="orderNumber" type="xs:int" use="required"/>
      <xs:attribute name="startTime" type="dateTimeCPP"/>
      <xs:attribute name="endTime" type="dateTimeCPP"/>
      <xs:attribute name="securityTime" type="xs:int"/>
      <xs:attribute name="speedMin" type="xs:int"/>
      <xs:attribute name="speedMax" type="xs:int"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- GenDmClass (MaintenanceTrainSchedule) [L292;C3->L292;C7] -->
<xs:complexType name="maintenanceTrainSchedule">
  <xs:complexContent>
    <xs:extension base="namedItem">
      <xs:sequence>
        <!-- GenDmAttribute (maintenanceTrainList) [L294;C42->L294;C61] -->
        <xs:element name="maintenanceTrainList" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="maintenanceTrainList"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="MaintenanceTrain" type="maintenanceTrain"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- GenDmClass (MaintenanceTrain) [L298;C3->L298;C7] -->
<xs:complexType name="maintenanceTrain">
  <xs:complexContent>
    <xs:extension base="namedItem">
      <xs:sequence>
        <!-- GenDmAttribute (possiblePaths) [L309;C37->L309;C49] -->
        <xs:element name="possiblePaths" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="possiblePaths"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="MaintenanceTrainPath" type="maintenanceTrainPath"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```



```
<!-- GenDmAttribute (listAllocatedOperation) [L306;C42->L306;C63] -->
<xs:element name="listAllocatedOperation" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <jaxb:property name="listAllocatedOperation"/>
        </xs:appinfo>
      </xs:annotation>
      <xs:element name="MaintenanceOperation" type="simpleRef"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="categoryId" type="xs:string"/>
<xs:attribute name="travelSpeed" type="xs:double"/>
<xs:attribute name="workSpeed" type="xs:double"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- GenDmClass (MaintenanceTrainPath) [L314;C3->L314;C7] -->
<xs:complexType name="maintenanceTrainPath">
  <xs:complexContent>
    <xs:extension base="namedItem">
      <xs:sequence>
        <!-- GenDmAttribute (pathDetail) [L316;C47->L316;C56] -->
        <xs:element name="pathDetail" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="pathDetail"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="ElementaryMaintenanceTrainPath"
type="elementaryMaintenanceTrainPath"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- GenDmClass (ElementaryMaintenanceTrainPath) [L320;C3->L320;C7] -->
<xs:complexType name="elementaryMaintenanceTrainPath">
  <xs:complexContent>
    <xs:extension base="objectInstance">
      <xs:sequence>
        <!-- GenDmAttribute (location) [L331;C21->L331;C28] -->
        <xs:element name="location" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="1">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="location"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="Location" type="simpleRef"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="orderNumber" type="xs:int" use="required"/>
      <xs:attribute name="startTime" type="dateTimeCPP"/>
      <xs:attribute name="endTime" type="dateTimeCPP"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- GenDmClass (Network) [L73;C3->L73;C7] -->
<xs:complexType name="network">
```



```
<xs:complexContent>
  <xs:extension base="namedItem">
    <xs:sequence>
      <!-- GenDmAttribute (pointList) [L75;C31->L75;C39] -->
      <xs:element name="pointList" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
              <xs:appinfo>
                <jaxb:property name="pointList"/>
              </xs:appinfo>
            </xs:annotation>
            <xs:element name="Point" type="point"/>
          </xs:choice>
        </xs:complexType>
      </xs:element>
      <!-- GenDmAttribute (trackGroupList) [L79;C36->L79;C49] -->
      <xs:element name="trackGroupList" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
              <xs:appinfo>
                <jaxb:property name="trackGroupList"/>
              </xs:appinfo>
            </xs:annotation>
            <xs:element name="TrackGroup" type="trackGroup"/>
          </xs:choice>
        </xs:complexType>
      </xs:element>
      <!-- GenDmAttribute (trackList) [L77;C31->L77;C39] -->
      <xs:element name="trackList" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
              <xs:appinfo>
                <jaxb:property name="trackList"/>
              </xs:appinfo>
            </xs:annotation>
            <xs:element name="Track" type="track"/>
          </xs:choice>
        </xs:complexType>
      </xs:element>
      <!-- GenDmAttribute (locationList) [L81;C26->L81;C37] -->
      <xs:element name="locationList" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
              <xs:appinfo>
                <jaxb:property name="locationList"/>
              </xs:appinfo>
            </xs:annotation>
            <xs:element name="Location" type="location"/>
          </xs:choice>
        </xs:complexType>
      </xs:element>
      <!-- GenDmAttribute (depotList) [L82;C26->L82;C34] -->
      <xs:element name="depotList" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
              <xs:appinfo>
                <jaxb:property name="depotList"/>
              </xs:appinfo>
            </xs:annotation>
            <xs:element name="Location" type="simpleRef"/>
          </xs:choice>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```



```
<!-- GenDmClass (Point) [L117;C3->L117;C7] -->
<xs:complexType name="point">
  <xs:complexContent>
    <xs:extension base="namedItem">
      <xs:sequence>
        <!-- GenDmAttribute (inputOutputLists) [L119;C24->L119;C39] -->
        <xs:element name="inputOutputLists" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="inputOutputLists"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="IOList" type="IOList"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- GenDmClass (IOList) [L123;C3->L123;C7] -->
<xs:complexType name="IOList">
  <xs:complexContent>
    <xs:extension base="objectInstance">
      <xs:sequence>
        <!-- GenDmAttribute (trackList) [L125;C23->L125;C31] -->
        <xs:element name="trackList" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="trackList"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="Track" type="simpleRef"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- GenDmClass (TrackGroup) [L109;C3->L109;C7] -->
<xs:complexType name="trackGroup">
  <xs:complexContent>
    <xs:extension base="namedItem">
      <xs:attribute name="type" type="xs:string"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- GenDmClass (Track) [L88;C3->L88;C7] -->
<xs:complexType name="track">
  <xs:complexContent>
    <xs:extension base="namedItem">
      <xs:sequence>
        <!-- GenDmAttribute (infoByCategory) [L94;C30->L94;C43] -->
        <xs:element name="infoByCategory" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="infoByCategory"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="TrackProperty" type="trackProperty"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```



```
<!-- GenDmAttribute (startPoint) [L97;C18->L97;C27] -->
<xs:element name="startPoint" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="1">
      <xs:annotation>
        <xs:appinfo>
          <jaxb:property name="startPoint"/>
        </xs:appinfo>
      </xs:annotation>
      <xs:element name="Point" type="simpleRef"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
<!-- GenDmAttribute (endPoint) [L100;C18->L100;C25] -->
<xs:element name="endPoint" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="1">
      <xs:annotation>
        <xs:appinfo>
          <jaxb:property name="endPoint"/>
        </xs:appinfo>
      </xs:annotation>
      <xs:element name="Point" type="simpleRef"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="trackGroup" type="xs:IDREF" use="required"/>
<xs:attribute name="length" type="xs:int"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- GenDmClass (TrackProperty) [L129;C3->L129;C7] -->
<xs:complexType name="trackProperty">
  <xs:complexContent>
    <xs:extension base="objectInstance">
      <xs:sequence>
        <!-- GenDmAttribute (track) [L135;C18->L135;C22] -->
        <xs:element name="track" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="1">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="track"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="Track" type="simpleRef"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="categoryId" type="xs:string" use="required"/>
      <xs:attribute name="minSpeed" type="xs:int"/>
      <xs:attribute name="maxSpeed" type="xs:int"/>
      <xs:attribute name="headWay" type="xs:int"/>
      <xs:attribute name="maxSpeedIfMaintenance" type="xs:int"/>
      <xs:attribute name="maxSpeedBackwards" type="xs:int"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- GenDmClass (Location) [L150;C3->L150;C7] -->
<xs:complexType name="location">
  <xs:complexContent>
    <xs:extension base="objectInstance">
      <xs:sequence>
        <!-- GenDmAttribute (coordinate) [L156;C21->L156;C30] -->
        <xs:element name="coordinate" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="1">
              <xs:annotation>
                <xs:appinfo>
```



```
<jaxb:property name="coordinate"/>
</xs:appinfo>
</xs:annotation>
<xs:element name="GpsCoord" type="gpsCoord"/>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="track" type="xs:IDREF" use="required"/>
<xs:attribute name="distance" type="xs:int"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- GenDmClass (GpsCoord) [L50;C3->L50;C7] -->
<xs:complexType name="gpsCoord">
  <xs:complexContent>
    <xs:extension base="objectInstance">
      <xs:attribute name="latitude" type="xs:double"/>
      <xs:attribute name="longitude" type="xs:double"/>
      <xs:attribute name="elevation" type="xs:double"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- GenDmClass (TrainSchedule) [L166;C3->L166;C7] -->
<xs:complexType name="trainSchedule">
  <xs:complexContent>
    <xs:extension base="namedItem">
      <xs:sequence>
        <!-- GenDmAttribute (trainList) [L168;C35->L168;C43] -->
        <xs:element name="trainList" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="trainList"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="TrainPath" type="trainPath"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- GenDmClass (TrainPath) [L172;C3->L172;C7] -->
<xs:complexType name="trainPath">
  <xs:complexContent>
    <xs:extension base="namedItem">
      <xs:sequence>
        <!-- GenDmAttribute (elementaryTrainPathList) [L181;C36->L181;C58] -->
        <xs:element name="elementaryTrainPathList" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="elementaryTrainPathList"/>
                </xs:appinfo>
              </xs:annotation>
              <xs:element name="ElementaryTrainPath" type="elementaryTrainPath"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>

        <xs:element name="travellingDay" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:appinfo>
                  <jaxb:property name="travellingDay"/>
                </xs:appinfo>
              </xs:annotation>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```



```
        </xs:appinfo>
        </xs:annotation>
        <xs:element name="WeekDay" type="weekDayJaxb"/>
    </xs:choice>
</xs:complexType>
</xs:element>

<!-- FR : Ajout manuel du vecteur des dates non circulées. -->
<xs:element name="noTravellingDate" minOccurs="0" maxOccurs="1">
    <xs:complexType>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
                <xs:appinfo>
                    <jaxb:property name="noTravellingDate"/>
                </xs:appinfo>
            </xs:annotation>
            <xs:element name="Date" type="dateJaxb"/>
        </xs:choice>
    </xs:complexType>
</xs:element>

<xs:element name="travellingDate" minOccurs="0" maxOccurs="1">
    <xs:complexType>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
                <xs:appinfo>
                    <jaxb:property name="travellingExceptionDate"/>
                </xs:appinfo>
            </xs:annotation>
            <xs:element name="Date" type="dateJaxb"/>
        </xs:choice>
    </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="categoryId" type="xs:string"/>
<xs:attribute name="variationCost" type="xs:int"/>
<xs:attribute name="cancelCost" type="xs:int"/>
<xs:attribute name="maxDelay" type="xs:int"/>

</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="dateJaxb">
    <xs:complexContent>
        <xs:extension base="objectInstance">
            <xs:attribute name="date" type="dateCPP"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="weekDayJaxb">
    <xs:complexContent>
        <xs:extension base="objectInstance">
            <xs:attribute name="weekDayName" type="weekDayNameCPP"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<!-- GenDmClass (ElementaryTrainPath) [L185;C3->L185;C7] -->
<xs:complexType name="elementaryTrainPath">
    <xs:complexContent>
        <xs:extension base="objectInstance">
            <xs:sequence>
                <!-- GenDmAttribute (location) [L192;C21->L192;C28] -->
                <xs:element name="location" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                        <xs:choice minOccurs="0" maxOccurs="1">
                            <xs:annotation>
                                <xs:appinfo>
                                    <jaxb:property name="location"/>
                                </xs:appinfo>
                            </xs:annotation>
                        </xs:choice>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```



```
<xs:element name="Location" type="simpleRef"/>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="arrival" type="durationCPP" use="required"/>
<xs:attribute name="departure" type="durationCPP" use="required"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- GenDmClass (NamedItem) [L56;C3->L56;C7] -->
<xs:complexType name="namedItem">
  <xs:complexContent>
    <xs:extension base="objectInstance">
      <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

</xs:schema>
```