Realization of Train Rescheduling Software System

Snežana Mladenović[1] and Slavko Vesković[2]

Abstract - **During the past decades numerous methods have been developed, resolving more or less successfully the NP-hard (re)scheduling problems. If, however, there is an ambition to bring into practice one of the scheduling methods, the research must focus on both the development of algorithms and of corresponding software systems. The paper presents the most interesting realization details of the first prototype of train rescheduling system.**

Keywords - **Train rescheduling, Software system, OPL, CP**

I. INTRODUCTION

The train scheduling problem belongs to a category of NP-hard problems of combinatorial optimization, and hence is complex for both modeling and solving. On the other hand, that problem must be solved as a part of tactical planning process in real railway systems.

The assignment of train rescheduling is that on a smaller fragment of railway network, over a shorter planning period an operational reconstruction of timetable is made, in respond to disturbances that have arisen.

The rescheduling in general may be considered to be a more difficult problem than an initial scheduling because additional requirements are imposed to it [1, 2]: to find a solution in a given real time; to have a recovered schedule which will deviate from the initial one as little as possible; the solution if not optimal, to be at least "good enough" with respect to the assigned objective function, etc.

Authors have been agreed that train rescheduling is quite a difficult work. According Norio et al. [3], major reasons of this are as follows:

- it is difficult to decide an objective criterion of rescheduling which is uniformly applicable;
- train rescheduling is a large size combinatorial problem;
- a high immediacy is required;
- no necessary information can be always obtained.

Since train rescheduling is such difficult job, assistance of software systems have been longed for, and nowadays train rescheduling systems are being practically used.

However, only a few published papers deal with train rescheduling software in real time. In fact, the current rescheduling systems test mostly if the solution proposed by the user is feasible one, and they are not doing full schedule regeneration. It also can be noted that authors simplify the scheduling problem in two ways: by simplification of the network structure and omitting and/or approximating constraints that govern the train movement. Thus, the model used by Isaai and Singh in [4] does not allow sequencing on single-track line between two consecutive stations. The first train must complete its arrival at the next station before the next train departs from the previous station. This rule relaxes problem but leads to a poorer solution, as slower trains can hold up other faster trains while traversing from one station to another.

The design, development and implementation of train rescheduling system are a specific assignment of the software engineering. The paper presents recommendations to be followed as general during design, development and implementation of the rescheduling system. In accordance with these recommendations, the first prototype of train rescheduling system has been realized.

The rest of the paper is organized as follows: Section 2 points to some particularities in the rescheduling software life cycle. Section 3 records the expected design requirements. Section 4 deals with the implementation issues. A special attention must be paid to testing in the incremental software development, and this is discussed in Section 5. An analysis of the first prototype results is a prerequisite for specification of new requirements and development of a new, more perfect prototype, which is discussed in Section 6. Finally, Section 7 presents conclusions.

II. RESCHEDULING SOFTWARE LIFE CYCLE

The spiral model, according to authors' opinion, is the model of the first choice for development of rescheduling applications. The basic assumption is that the specification of users' requirements is not completely finalized before the stages of design and implementation. The software system is developed incrementally, by developing a series of prototypes, that being verified and validated, considering the new user requirements.

A particularity of the proposed spiral model is a risk analysis that must be carried out before design of any new prototype. The risk in developing a rescheduling software system is not low; the prototype may simply "fail", if too tight time limits have been imposed

Herein, only the details of the train rescheduling software life cycle that differentiate from other software systems will be highlighted.

III. DETERMINATION OF REQUIREMENTS

Essential requirements put to the train rescheduling software are:

- interaction with other software systems,
- existence of graphic user interface,
- respect to time limits.

The rescheduling system must receive information from hierarchically higher planning levels. Thus, an initial train timetable and network topology must be accessible to the train rescheduling software. The rescheduling system also must

Authors are with Faculty of Transport and Traffic Engineering University of Belgrade, Vojvode Stepe 305, 11000 Belgrade, Serbia , E-mail's: snezanam@sf.sf.bg.ac.yu; E-mail: veskos@sf.sf.bg.ac.yu

receive the latest information regarding the resources availability, job progress, etc. from the process monitoring system. A general idea is for all data available to be found in the information system **D**ata**B**ase DB, wherefrom they will be accessible to the rescheduling software. Usually, a significant effort is required to adapt the real system database for the rescheduling system input. The requirement to make the database correct, consistent and complete often assumes a designing of a series of tests the data must be run through before being used. Since a real database from the railway company was not accessible during the research, a demo MS Access database was created, assuming that it is correct, consistent and complete.

Within the database, it is possible to distinguish the static and dynamic data. The static data are all data on jobs and resources that do not depend on scheduling. E.g., data on train categories belong to static data. Data on resources are relatively static data, as well as on the regular timetable. The job priorities are also static data, not depending on scheduling, either. The priority may be based on the planner's assessment or may be result from the procedure taking into account other data from the information system database. The job priority change may depend on the scheduling period, but also by some external, hardly predictable events; thus suburban and urban trains may temporarily, during the peak hour period, get a higher priority than international trains. In our model, the change of priority procedure takes into consideration the expert assessments in the database made for individual sections and individual periods during the day.

The dynamic base consists of all schedule-dependant data: job start and completion times, current job positions, number of delayed jobs etc. Some data may be considered as both static and dynamic, e.g. resource setup time. Occurrence of unexpected dynamic data in the DB is actually a trigger for the train rescheduling procedure.

The scheduling **M**odel **B**ase MB holds an important place in our rescheduling system also. It collects the models that optimize one or the other objective function, imposing or relaxing certain constraints. A special procedure selects a model from the MB, taking into account the user's wish the function wants to optimize.

The architecture of a hypothetical information system with incorporating a train rescheduling system – schedule recovery module, is presented in figure 1 in the form of data flow diagram between the processes. The schedule recovery module should enable the model management: choice, combining, sequencing, running etc.

The user interfaces can determine the scheduling system usability. Obviously, the scheduling visualization must resemble the one the users are accustomed to during their work for many years. Also, since the inference engines and decision-making are hidden from planners–users, the presentation of scheduling results must be such to make him assured quickly and easily on the validity of the "good-enough" solution found.

The scheduling software must have an interface based on "WIMP paradigm" (**W**indow, **I**nteraction, **M**ouse, **P**ointer).

The user interfaces for database modules are in a standardized form and are determined by a used database management package.

The realized schedule recovery module contains 7 models corresponding to different objective functions (maximum tardiness, maximum weighted tardiness, total tardiness, total weighted tardiness, makespan, maximum slack of trains in stations, number of late trains). If we wish to offer the planner–user a possibility of a direct model choice, we must furnish him with a user interface that will enable such manipulation. No other modes of interactive manipulations are necessary because the very objective of the rescheduling system is a full automation that eliminates the user's slow actions.

The literature describes numerous standard user interfaces for presentation of scheduling information [5]. It is interesting that none of the standard graphic interfaces was fully suitable for schedule visualization in the train rescheduling problem for objective and subjective reasons. Objectively, none of the interfaces presents transparently enough the sequencing, overtaking, crossing, waiting and movement of trains. On the other hand, the planners–users are used to visualization known as the train diagram, which is, actually a slightly modified Gantt chart. The modification consists of "touching" the resources on y-axis (in fact, the real infrastructure objects touch each other) and the replacement of bars by their diagonals, which symbolizes the train movement on a resource. Figure 2 represents the realized graphic presentation of a found schedule of the train rescheduling system.

The modified Gantt chart is made automatically by MS Excel. In the graphic presentation, "background" is always a tabular presentation of train diagram, presenting the start and end times of each job activity, along with the information on which resource it will be accomplished.

IV. IMPLEMENTATION

The implementation of the scheduling system is accompanied, from the very beginning by perplexities. Namely, dozens of software companies claim they have developed automatic scheduling generators, that can be applied in different real systems, these being either a choice of a commercially available scheduling generator, development of a "from zero" system, or a combined approach?

In attempts to implement the train rescheduling system solely by scheduling generators available, a series of problems arose: an immensely oversized scheduling problem, difficulties in establishing a link between the scheduling system and the schedule implementation monitoring system, difficulties in coding special cases (e.g. station tracks under specific conditions behave as a single resource, and as separate resources otherwise), timing requirements, ... On the other hand, there was a idea to speed up the stage of implementation by using the scheduling generators available.

The **I**ntegrated **D**evelopment **E**nvironment (IDE) OPL Studio [7] enables us to create and modify the **C**onstraint **P**rograming (CP) and scheduling models using the **O**ptimization **P**rogramming **L**anguage (OPL), to compose and control models using the procedural language OPL Script, to run models by ILOG Solver and ILOG Scheduler as well as to presentation of results (schedules) in a tabular and graphic form. The OPL Studio trial version is available on site

http://www.ilog.com/products/oplstudio/trial.cfm and it was used for implementation of the first prototype of train rescheduling system.

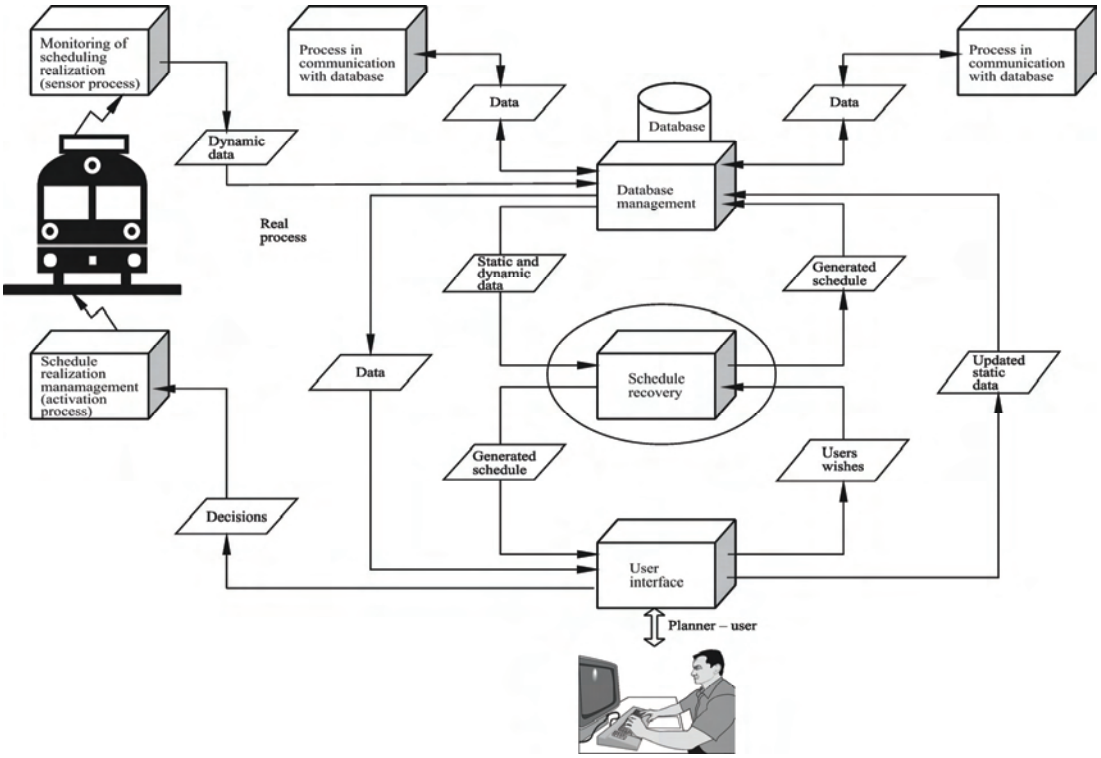

Fig. 1. Position of schedule recovery module in a real information system

In order to improve the time performance the software tools available was added to modules implementing heuristics specific for the train rescheduling problem. By **separation heuristics**, a global problem was split up into a series of smaller size subproblems the ILOG Scheduler is able to solve up to optimality. The **bound heuristics** have initially limited the decision variable domains and objective functions, and the **search heuristics** have directed the search into the space promising good solutions. These heuristics were described in details in [6]. A declarative nature of OPL was used for a simple formulation of the scheduling model. The schedule recovery module itself represents in fact a procedure formulated in the OPL Script with controlling the optimization models in a suitable way constructs "good enough" schedule in the limited time.

V. Prototype Testing

The software system testing is traditionally carried out through verification and validation. The prototype verification and validation must be carried out according to the standard criteria for software, these being efficiency: reliability, usability, modifiability, portability, testability, reusability, maintainability, interoperability, and correctness. The first train rescheduling system prototype was assessed for all quality criteria. The specificity of the train rescheduling application imposes to clarify in this point, in particular:

- **efficiency** – the software must work within the envisaged time limit. This is a key property of the rescheduling system. The testing of systems that have to operate within the given time limit is extremely difficult. The idea is to reiterate each test example for numerous times, and a good direction for support to this idea is the automatic testing. The testing of the first train rescheduling system prototype was made manually, on a large number of input timetables with existing disturbances; each test example, however, was run at least 10 times to determine the mean value of CPU time. It was found during the testing that the variation of CPU time still is not drastic (not more than ± 10 % of the mean value);
- **reusability** – the property of the software that, as a whole or in parts, it may be used for development of similar systems, thus increasing the development productivity of related systems. E.g., it is known that the train rescheduling is a core of the systems which deal in railway traffic with: timetable preparation, determining of economically acceptable utilization of capacity interval, estimate of costs of stopping and tardiness of trains for operational reasons, forecasting the effects of investments, identification of bottlenecks in infrastructure, choice of possible solution of a conflict point, testing the arrangement and layout of block sections and signals along the railway line etc. Standardization of software modules and their communication is a good direction for upgrading their reuse;
- **modifiability** – expresses a property of easy modifications on the software in case of changed user's requirements. E.g., it is reasonable to expect that the set of constraints be to subject to changes. A declarative nature of CP approach, separation between the constraint

component and search component, available CP tools, offer the programmer fantastic possibilities to achieve the highest level of modifiability.

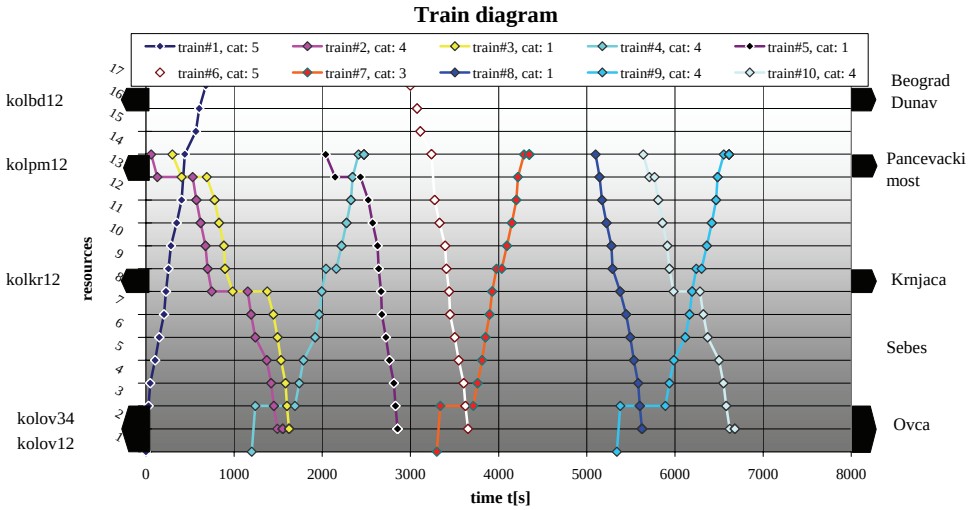

Fig. 2. Realized window of user interface of the first prototype of train rescheduling system – a graph presentation of the recovered train schedule in the form of modified Gantt chart

VI. Analysis of the Prototype Results

The testing of the first train rescheduling system prototype was carried out on a large number of input timetables with existing conflicts Experiments have been carried out on a fragment of real railway network (a part of Belgrade Railway Junction), with actual train categories operating there, but **with traffic frequency immensely exceeding the real one**. The jobs (trains) are "piled up" on purpose to test the endurance of the method. All seven relevant objective functions participated in the experiment. Each set of jobs suffering disturbances includes trains of different categories and different movement directions. All experiments have been implemented on personal computer Intel (R) Pentium(R) 4 CPU, 2GHz. For example, CPU time needed for generation of schedule, given on figure 2, was 36.65 seconds where `objective function was maximum tardiness.` From the analysis of experiment results the following conclusion may be drawn:

- CPU time of schedule recovery depends on the number of activities and number of conflicts;
- solving of initial conflicts may bring up additional conflicts;
- in most cases the time performance is satisfactory;
- a heuristic nature of the approach has been demonstrated (in an insignificant number of cases the best-known solution for the given objective function has not been found).

Based on this analysis, we can formulate the proposals the acceptance of which should lead to the improved prototype.

VII. Conclusion

A major part of theoretical research carried out during the past decades in the field of scheduling has a limited application in real systems. Therefore, **the research must be concentrated on the development of algorithms, but also on development of software systems.** This paper is one of such attempts.

The rescheduling software system should enable the planner to produce faster a better quality schedule. There are also other reasons for introducing automatic scheduling systems. The scheduling system requires advanced "disciplines" from other subsystems of the real information system. It oblige and ensure the real process to develop according to schedule.

Experiments made on the first train rescheduling system prototype, on a real railway network fragment, with real traffic structure, and possible disturbances, make us believe that the approach proposed in this paper may offer a full support to the railway operational management.

References

[1] Cowling, P. and M. Johansson, “Using real time information for effective dynamic scheduling”, European Journal of Operational Research, 139(2), pp. 230-244, 2002.

[2] Vieira, E. G, J. W. Herrmann and E. Lin, “Rescheduling manufacturing systems: a framework of strategies, policies and methods”, Journal of Scheduling, 6, pp. 39-62, 2003.

[3] Norio T., Y. Tashiro, T. Noriyuki, H. Chikara and M. Kunimitsu, “Train rescheduling algorithm which minimizes passengers’ dissatisfaction”, in Innovations in Applied Artificial Intelligence, Lecture Notes in Artificial Intelligence 3533, Springer Verlag, pp. 829-838, 2005.

[4] Isaai, M. T. and M. G. Singh, “An object-oriented, constraint-based heuristic for a class of passenger-train scheduling problems”, IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, 30(1), pp. 12-21, 2000.

[5] Pinedo, M., *Scheduling: Theory, Algorithms and Systems*, Prentice Hall, 1995.

[6] Mladenović, S., M. Čangalović, D. Bečejski–Vujaklija and M. Marković, “Constraint programming approach to train scheduling on railway network supported by heuristics”, 10th World Conference on Transport Research, CD of Selected and Revised Papers, Paper number 807, Abstract book I, pp. 642-643, Istanbul, Turkey, 2004.

[7] ILOG OPL Studio http://www.ilog.com