Hynek Bachraty – Emil Krsak – Marek Tavac *

# ALGORITHM FOR GENERATING TEXT DESCRIPTIONS OF BIT CALENDARS

*The article deals with the problematics of generating calendar text descriptions based on their bit map. We introduce the description of the algorithm we have designed, its basic ideas and parts. We also present the results of testing the successfulness of the algorithm before it is primarily used.*

## 1. Problem characterization

Calendar data are a significant part of information on provided services, executed activities, frequent events etc. In connection to our work we come across them particularly in the area of transport, where processing and displaying them has been an important component of many information systems. Most frequently, they describe means of transport operation days or validity days of some further transport characteristics. Therefore, in our article we are going to talk about the calendars describing a train operation days. But the text-generating algorithms created and described hereby are to a great extend universal, and by changing the text being used it is possible to adapt the outputs also to other, different situations.

Most frequently, calendar data are entered and stored in two ways: in bit maps and by means of text descriptions. We use the term bit map to denote such form of information which, for each day of a particular period, sets whether a train operates or not on a given day. The calendar then can be expressed (and this is also how it is usually represented in a data way) as a sequence of ones (1s) (the train operates) and zeros (0s) (the train does not operate) corresponding to individual days [1]. A part of the information on calendar is the beginning and the end of its validity period. This, most often, also secures the link-up between the data and real calendar.

The calendar defined by a *bit map* is optimal from the viewpoint of informatics. By means of the bit map it is easy to edit or find out about a train operation days, to compare and combine data of several calendars, to find out about their characteristics etc. Basically it is possible to picture only graphically, which is reasonable to do only on a monitor of sufficient size. The user needs to have necessary equipment, as well as time to use such way of the data information display. This form of a calendar set-up is, on the other hand, unsuitable for mass print outputs, depiction on small screens (e. g. PDA devices), for other than graphical transfer of train rides information etc.

Therefore, the calendars defined by a bit map are suitable to work with for information systems and specialized users who, as a part of their job, create, change, analyse etc. date data. Reversely, they are not suitable for mass usage, for common human interaction and so on.

A calendar *text description* has been most frequently used in date annotations in various types of timetables and other information utilities. This date information description attributes are basically opposite to those with the bit map. The contents of the calendar are composed into a relatively short text. That is why it suits in print outputs or for small displaying devices. The calendar validity is possible to communicate verbally in an easy way. Regarding our experience with using this form of date description, we can easily gain some types of information from the text description, e. g. whether a train operates tomorrow, next Monday or on Christmas Day. We would, on the other hand, have problems, if we were supposed to use the text description to define the number of a train operation days within the observed period, to find out which of the two given trains operates more frequently, or on which days both of the trains operate.

A trully existing problem which we needed to solve lies in the algorithmic interconnection between the two means of presenting date information. To the ways of work used in the transport area mostly applies that first of all bit calendars are at disposal, and they consequently need to be expressed in a text form. Bit maps are gained either from information systems for transport planning, or gained from international, whole-Europe databases, where using text expressions is faced with language problems. In the article we are describing an algorithm for generating text descriptions from

* **Hynek Bachraty, Emil Krsak, Marek Tavac,**
Department of Software Technology, Faculty of Management and Informatics, University of Zilina, Slovakia, E-mail: Hynek.Bachraty@fri.utc.sk

a calendar bit map, which has been created for the above purposes, as well as the first results of testing the algorithm.

## 2. Solution starting points

The basic task is to generate text corresponding with a given calendar bit map. For the bit map we also suppose setting the initial and finishing days of the calendar validity period. These at the same time determine the length of the validity period. As long as we are considering several calendars, we assume identical initial and finishing days of the validity of all of them. In case this condition is not complied with, we are able to provide it formally by prolonging the validity periods of all calendars to the maximum used extent. The algorithm for genereting text we have designed can manage unifying the validity period by extending it without any difficulties.

Further on, we assume we are solving the problem for calendars used in the area of public personal transport. They therefore serve a large group of citizens as important access information in relation to a certain service. The service presupposes a certain degree of their regularity, simplicity and lucidity. English speaking people would say the operation days have to have created a certain ´pattern´. Otherwise date information not lucid enough would make using the service difficult. Searching and finding the regularity, or pattern, is the essence of the problem solution.

The algorithm is of course able to process a random calendar from a different area, too, e. g. a calendar of cargo or special transport, service calendar etc. But here we can expect lower effectivity and success rate of the algorithm. However, a narrow group of users of such special calendars enables to presume using the specific possibilities to display the calendar, e. g. graphical picturing, extensive text, possibility to enquire about individual operation days etc.

The calendars text descripitons which are to be the algorithm output also have to meet certain principles. The text needs to be clear, explicit, structured and standardized. As far as it is supposed to describe a wider type range of calendars, it will be necessary to use several text types. Nevertheless, their number should not be too high. Recognizing a calendar type and consequent assignement and specification of a particular text type is another key goal of the solution. The text needs to have a structure, preferably common for all of the types, and not extensively complicated. It should contain a ´pattern´ part where the information gets compressed most remarkably, by means of the found calendar regularity description. The regularity, however, will not always be possible to achieve absolutely exactly, and it will be necessary to specify exceptions to the regularity. Both information need to be expressed in a compact and inter-related way, not to confuse the user.

Important criteria of the created algorithm quality will be the algorithm ability to generate text as short as possible or at least texts of acceptable length, and as high as possible share of calendars which the algorithm will be able to generate a text for.

## 3. Determining the calendar regularity

Searching a pattern of calendar validity is based on the weekly rhythm of our life. The basic patterns correspond to the repetition of individual weekdays (Monday to Sunday). The basic bit map is compared to all 128 combinations of the seven basic calendars. They are so called sample calendars. If necessary, the 7-day cycle is possible to modify into a different number of days, although another one than the 7-day calendar is in fact not used [2]. This option can be used for example when processing calendars related to others than weekly maintenance cycles of different equipment.

To detemine the most apropriate sample calendar, as a criterion we have used the lowest possible number of exceptions, i. e. days of the entered calendar the validity of which does not correspond to the tested sample calendar. They might be so called positive exceptions (the train also operates on the days beyond the tested range) and negative exceptions (the train does not operate on the days of the tested sample). In optimal case the tested sample calendar fully agrees with the entered calendar and the number of exceptions equals zero. It that case it is possible to terminate the testing. On the contrary, one of the algorithm parameters is the value of the ultimate acceptable number of exceptions. As far as it is exceeded, the tested sample calendar is not considered as conforming. It is possible we will not consider any sample calendar as conforming, and searching for regularity will not be successful.

When testing calendars in connection to a particular country, it is apropriate to complement two more sample calendars. They are so called holidays (Sundays and bank holidays) and working days (Monday to Friday, holidays excluded), which we usually mark with + and X. Thus we gain as many as 9 basic calendars that we combine with each other. However, there is no point in combining the holiday's calendar with the calendars containing Sundays, and the working days calendar with the calendars containing only Monday to Friday days.

The description of the found sample represents the main component of the generated text. The way the week days are marked is optional, most frequently they are abbreviations or serial numbers of individual days (Mon to Sun with the sequence 1 to 7). With a particular algorithm implementation it is possible to use special fonts for the marking. Standardly the text looks as follows:

*´The train departs on Mon, Thu, Fri´ or ´The train departs on (1), (4), (5)´.*

In some special cases simplified and commonly utilized texts are used, such as *´The train operates daily´, ´The train does not operate´, ´The train operates on Mon to Fri´* or *´The train operates on Sat and Sun´*.

Eligibly it is possible to use also so called negative notes. As long as listing the weekdays is too long, a modified text can be used.

*´The train operates daily except (3), (4)´* or *´The train does not operate on (3), (4)´* instead of *´The train operates on (1), (2), (5), (6), (7)´*.

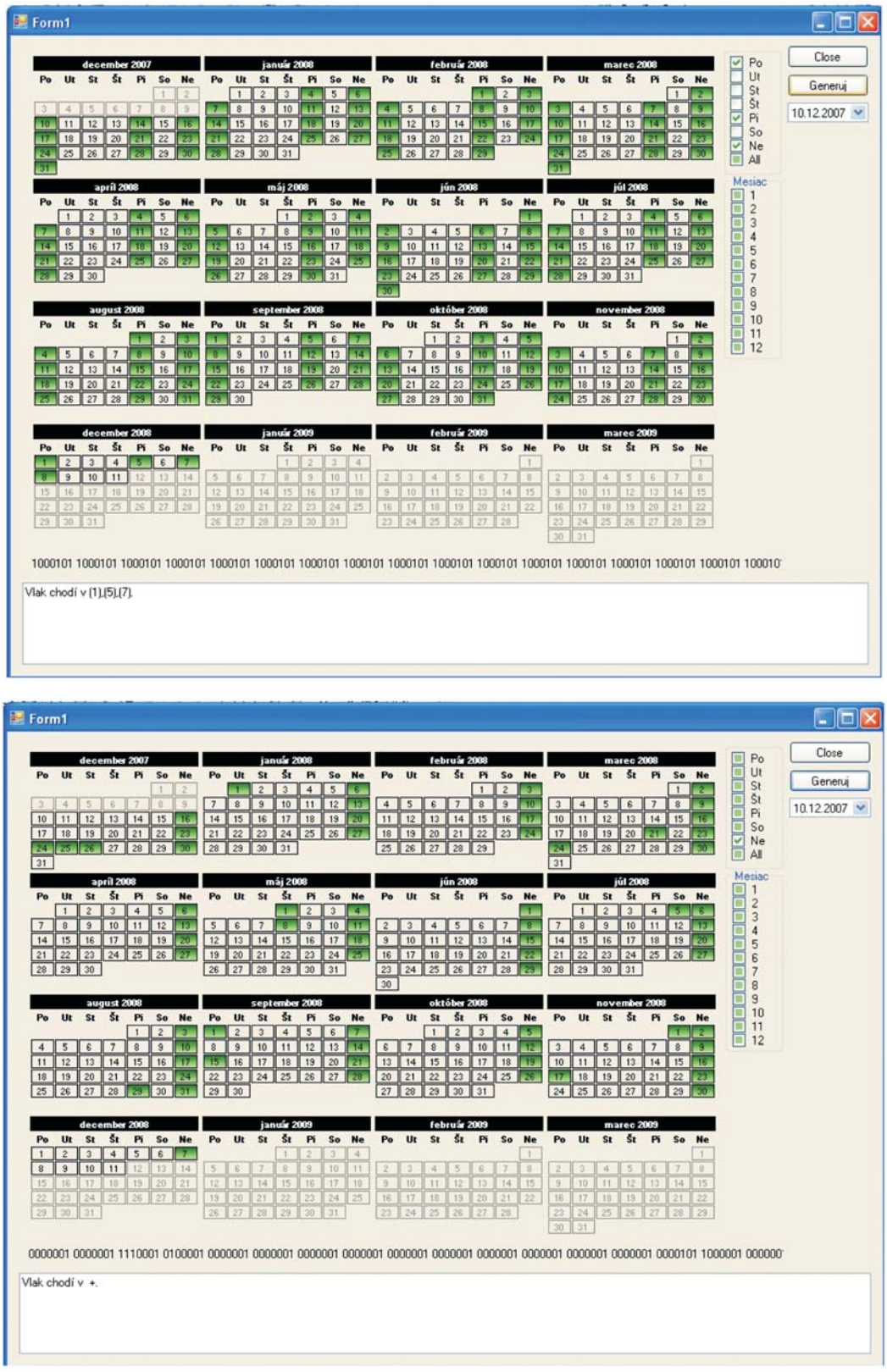


*Fig. 1 GUI of bit map editting component.*
*Example of simple ´Mon+Fri+ Sun´ regularity and ´holidays´ sample calendar*

With the second text above the information that the train operates on all days but Wednesdays and Thursdays is only implicit, therefore the text is shorter but less precise. A small explorative enquiry has shown that naming three days in a negative form is not accepted by users, that is why we use it only for one or two ´negative´ days.

The regularity description is followed by listing exceptions as follow:

*´The train operates on (5), (6), except 20 XII, 14 III and 18 IV. The train operates on 15 I and 14 V´* or *´The train operates daily except (5) and does not operate on 20 XII, 14 III and 18 IV. The train operates on 15 I and 14 V´.*

We have selected the approach of the negative exceptions are introduced as rather positive, right after the regularity description. This is based on the experience that users, having gained positive information, are likely to finish analysing the calendar text description. Therefore they might wrongly assume the train operates on the given day, although the opposite is true. That is why the negative exceptions are introduced as soon as possible and within one sentence together with the regularity description. On the contrary, in case of not being successful our mind has a tendency of continuing to explore, thus the positive exceptions may ensue in the following sentence.

When listing the exceptions, the list of individual days conforms with specific rules. Continual stretches of the days following one after another are indicated just by the first and last days. Marking a month is used only when it gets changed, and marking a year is used only in case the given day repeatedly falls into the effectiveness period.

*´The train operates on 10 XII 2008 – 12 XII, 19 – 23 II, 13, 14 V, 29 VI – 3 VII, 9 – 11 XII 2008.´*

The example above also illustrates one of the options of text-generating, in case a suitable regularity has not been found, therefore the algorithm has failed. Then we deal with listing all of the valid caledar days. Such text, however, often happens to be long and not always possible to use. That is why there is another possiblity – generating an empty chain or alerting text, e. g. *´The calendar text description is not available´.*

## 4. Determining the calendar type

The above described way of regularities searching brings good results in case the train operates with certain regularity in the whole, most frequently 1-year period of the calendars validity. It corresponds to the validity period of the tested sample calendars. Nevertheless, as long as the train operates on Mondays and Fridays only during the school year, the high number of exceptions due to the holiday periods would, from the year-round point of view, refuse this regularity. This fact has brought us to the idea of tipping-out and consequently searching certain calendar types in the given bit map. The types should identify the period within which the train operates, and search and test the regularity within this period only. Consequently, an appropriate type text, complementing the information on the found regularity, will be generated for each calendar type.

We have decided to maintain the number of types low, for lucidity´s sake, however, at the same time we have to be able to use them to describe as many bit maps as possible. That is why we have also decided to resist the possibility of using recursive types searching, which is easy to implement from the viewpoint of programming. Our algorithm is currently using 5 types described further on. The assigned bit map is tested for its applicability to individual types, and it can also fall under more of them. As long as this happens, the type with the lowest number of exceptions from the found regularity gets selected. In case of an identical number of exceptions, as simple as possible type will be used. That corresponds to the sequence we have listed them in.

The basic type is so called *year-round calendar.* In this case, the text introduced in the previous part is generated.

The following type is the calendar with one ´operates´ period. It corresponds to the calendars according to which before and/or after the train operation-days period, i. e. at the beginning and/or at the end of the validity period the train does not operate at all. The minimal scope of the period in which the train (with consequently searched regularity) operates, is set as the algorithm parameter. For this type of calendar the texts are generated as follow:

*´The train operates only from 1 XII to 31 VII on Mon, Thu, Fri´, ´The train operates only from 1 XII to 31 VII daily except (2), (6)´,* or with exceptions *´The train operates only from 1 I to 31 VIII daily except Tue, Sat and does not operate on 9 IV, 14 V and 27 V. The train operates on 14 IV, 14 VII and 15 VII´.*

A developed version of the above type is the *calendar with two ´operates´ period.* In calendars like these, instead of one there are two ´the train operates´ periods, separated from each other by the period when the train does not operate at all. The generated texts look as follow:

*´The train operates from 4 IV to 31 V and from 4 VII to 31 VII on (1), (5), (6), (7)´,*

*´The train operates from 2 V to 30 VI and from 1 IX to 31 X daily except (4)´,*

*´The train operates from 30 I to 30 IV and from 1 VII to 30 IX daily except 31 I and 15 VII´.*

Another calendar type is the *calendar with a ´does not operate period´.* In this case the bit map serves to find one period within the validity period in which the train does not operate at all. The minimal range of the ´does not operate period´ is again given by an eligible parameter. The texts are generated as follow:

*´The train, except the period from 1 IV to 2 VIII operates on Mon to Fri´, ´The train, except the period from 1 V to 31 VII operates daily except Mon, Sun.´* or *´The train except the period from 1 IV to 2 VIII*


*Fig. 2 Example of one and two ´operates´ period calendar types*


*Fig. 3 Example of ´doesn´t operates´ and ´operates daily´ period calendar types*

*operates daily except Sat, Sun, and does not operate on 19 II, 11 III and 19 VIII. The train operates on 26 X, 8 XI and 22 XI´.*

The last calendar type is the *calendar with ´operates daily´ period.* In this case the bit map needs to contain sufficiently long period in which the train operates daily. This period is introduced in an independent sentence at the end of the generated text.

*´The train operates on (1), (3). In the period from 1 VI to 31 XII the train operates daily.´ ´The train operates on (2), (3), (6) except 23 I and 16 IX. The train operates on 30 X and 13 XI. In the period from 1 VII to 31 VIII the train operates daily´.*

When testing a calendar type we came across a problem of specific extraordinary train operation days which disturbed the continuous periods when the train did not operate, and thus prevented the calendar from being allocated to otherwise unambigous type. They are e. g. ´holidays´ trains routings which are used as reinforcement before Christmas or Easter holidays. We call these days *isolated operation days* and we identify them as the days before and after which there is a sufficiently long period determined by the algorithm parameters in which the train does not operate. The days are identified and temporarily removed from the bit map even before the determination of the calendar type begins. Their maximal number is one of the algorithm parameters. The isolated days are listed in a special complementary sentence ´the train also operates on ...´at the end of the generated text.

*´The train operates in the period from 4 IV to 28 VII on (1), (5), (6), (7). The train also operates on 6 II and 18 IX.´*

## 5. Language usage

The content of the generated texts of course depends on the quality of algorithms which search the relevant calendar type, its regularity and exceptions to it. Last but not least, the language quality is important, too. We have already mentioned some of the principles, now we are going to talk about some language specifics concerning the text description of the train operation days.

First of all, the verb determining the sentence contents is important. It is convenient if the verb is able to help us distinguish between processes regularly repeating themselves and processes that happen once. In Slovak we consider using the verbs to go and to be running (to operate and to be operating) as optimal. We use the first one to describe the regularity, the second one to describe the train rides exceptions. As long as the calendar does not describe the train operation days, we recommend to select a different verb pair. Another important part of sentences is the subject. Hereby, we have been using the word train. It is easy to replace it with a more universal word (line, route) and we have created also the text version with silent subject. These are possible to use universally e. g. to describe the train rides calendar, and, at the same time, direct carriages. We can also talk about the service, its availability level etc.

Another thing we would like to point out is conscious usage of simple, unambigous and clear conjunctions and particles (except, also) the meanings of which users are able to understand and realise very well. In case of using a negative note, we avoid double negatives. Therefore, we do not introduce negative exceptions by using the word ´except´ for the second time but we use the text as follows: *´The train operates daily except (5) and does not operate on 19 V.´*

Generally, we try to generate only simple clauses or simple sentences. In the examples used hereby we have been using slightly longer text variants; it is possible to use more concise and shorter text forms, too.

One needs to realise that language formulation is an extention to the algorithms to search for the calendar type and regularity. It should present and at the same time not depreciate their results, which is a rather difficult but on the other hand manageable task. Modifying the formulations is supposed to enable us to use the algorithm core for the bit calendars text descriptions also from a different area than just the transport one, and also to use the algorithm in various languages mutations.

## 6. The algorithm implementation

The created algorithm works in accordance with the principles described in the previous chapters. First, isolated operation days are idendified in the bit map. Then they are saved into a separate data structure and removed from the bit map. Next, the bit map is tested for what calendar type it agrees with. Consequently we search the most suitable combination of sample calendars for the convenient types and the periods within which the train operates that result from them. Thus we determine the calendar regularities. On the basis of the found regularity, calendar type and isolated operation days algorithm generates a relevant text.

Implementing the algorithm we also concentrated on another important property. As we have already mentioned, transforming the calendar bit maps into its text representation is a problem which occurs practically in each information system for supporting the basic or operational planning. Thus, when designing the software architecture of the modul implementing this algorithm, we put great emphasis on universality and robustness. Our aim was to use the modul in various systems without the necessity of extensive alterations. The algorithm presupposes the existence of various binary (and, or, xor etc.) as well as unary (not) operators over the bit maps. Furthermore, the calendar bit map interface was defined as a bit values field together with the calednar validity attributes. To use the modul in arbitrary information system, the system just needs to implement the interface. The algorithm approaches the bit map through the interface, and doing that, it does not need to know the implementation details, e. g. the way of the bit map implementation into the system.

To be able to modify the bit map in a simple manner, the modul also contains a bit map editting component. It as well contains the buttons for mass introduction and changes of it departs / does not depart tokens of the whole day groups, such as week

days, month days. Last but not least it contains a place for writing out a bit map text description which is generated immediately with the bit map change. This behaviour is extremely user-friendly.

## 7. Testing and implementing the algorithm

We did the algorithm pilot testing on a set of calendars of MERITS – the whole-Europe passenger-trains database. Except the entire database we tested separately the calendar sets of the Slovak and Czech Railways trains.

Text generating is controlled by several mentioned parameters. The maximal determined number of exceptions for the testing was 15, and the maximal number of isolated days was 5. The minimal length of period to determine a calendar type was 14 days. In case the algorithm failed the generated text listed all of the calendar valid days in a form of positive exceptions.

The pilot testing brought good results. For the Slovak calendars, in 44 cases we also used the + calendar, and in 89 cases the X calendar. Regarding the similarities of the Slovak and Czech Republics, we used the + and X calendars in approximately identical numbers for the Czech Republic, too. The percentage of calendars which used the isolated days directly increased the algorithm success rate.

By means of setting the parametres in various ways it is possible to increase the algorithm success rate, as well as the quality of generated texts. The relation between the parameters values and the algorithm success rate are to be subject to thorough testing. The simplest way is increasing the number of exceptions but this, together with hightening the success rate, also makes the generated texts longer. Another possibility is to reduce the number of days necessary to accept a period when determining the calendar type. A detail analysis of bit maps which the algorithm has not been successful with will also be important. The analysis can prove it will be necessary to introduce few more calendar types. We suppose that increasing the algorithm success rate will, from a certain level on, require to consider some national specifics.

## 8. Conclusion and further development

The Slovak Railways use the algorithm basic version to generate text descriptions in the information system to search for an

Calendars of the Slovak Railways trains. 791 calendars altogether, 10 (1.26%) with isolated days.

| Calendar type | Not successful | Round-a-year | ´operates´ period | two ´operates´ period | ´does not operate´ period | ´operates daily´ period |
|---|---|---|---|---|---|---|
| Number / % | 45 / 5.69 | 396 / 50.06 | 219 / 27.69 | 26 / 3.29 | 83 / 10.49 | 22 / 2.78 |
| Aver. number of exceptions | 171 | 5 | 0 | 2 | 4 | 5 |
| Aver.number of letters in text | 245 | 61 | 61 | 92 | 87 | 101 |
| Min./max. number of letters | 42 / 538 | 13 / 144 | 45 / 121 | 66 / 131 | 48 / 154 | 67 / 164 |

Calendars of the Czech Railways trains. 1352 calendars altogether, 14 (1.04%) with isolated days:

| Calendar type | Not successful | Round-a-year | ´operates´ period | two ´operates´ period | ´does not operate´ period | ´operates daily´ period |
|---|---|---|---|---|---|---|
| Number / % | 155 / 11.46 | 631 / 46.67 | 341 / 25.22 | 65 / 4.81 | 96 / 7.10 | 64 / 4.73 |
| Aver.number of exceptions | 183 | 5 | 1 | 2 | 3 | 6 |
| Aver.number of letters in text | 295 | 59 | 68 | 92 | 85 | 116 |
| Min./max. number of letters | 45 / 585 | 13 / 146 | 45 / 155 | 65 / 159 | 48 / 142 | 63 / 174 |

Calendars of the complete MERITS database. 21 389 calendars altogether, 642 of them (3.00%) calendars with isolated days:

| Calendar type | Not successful | Round-a-year | ´operates´ period | two ´operates´ period | ´does not operate´ period | ´operates daily´ period |
|---|---|---|---|---|---|---|
| Number / % | 2282 / 10.67 | 7157 / 33.46 | 7240 / 33.85 | 3606 / 16.86 | 794 / 3.71 | 310 /1.45 |
| Aver.number of exceptions | 161 | 6 | 3 | 2 | 5 | 7 |
| Aver.number of letters in text | 223 | 61 | 80 | 92 | 95 | 113 |
| Min./max. number of letters | 36 / 770 | 13 / 158 | 45 / 176 | 62 / 196 | 46 / 174 | 62 / 204 |

optimal conection. The system called VIS has also been created by our department. The algorithm results have been satisfactory, and having implemented it, we gain priceless notices and impulses.

Now we are preparing new and full-scale version of the algorithm for VIS system. Further on, we suppose the algorithm implementation mainly with the ZONA and SENA information systems for timetable construction, to generate the calendar description texts for passenger timetables.

We are rather surprised that, with respect to importance of the problem, we don´t know about any similar algorithm, result or research. Thus we will welcome any contact and possibility to meet another approach and compare the results.

**Acknowledgement**

This work was supported by project "Centre of Excellence for Systems and Services of Intelligent Transport" ITMS 26220120028.

## References

[1] RUZBARSKY, J.: *Algoritmization of timetable creation,* INFOTRANS 2002, ISBN 80-7194-419-X

[2] GABOR, M.: *Data stream and modification in the timetable construction and other aids of the train traffic diagram,* INFOTRANS 2005, ISBN 80-7194-792-X

[3] SOTEK, K., BACHRATY, H.: *New trends in creation of railway timetable,* Conference EURNEX-ZEL 2008, ISBN 978-80-8070-853-5.