# A TECHNOLOGY FOR SCHEDULING OF DATA EXCHANGE OVER BUS WITH CENTRALIZED CONTROL IN ONBOARD AVIONICS SYSTEMS[1]

V.V. Balashov, V.A. Balakhanov, V.A. Kostenko, R.L. Smeliansky, P.E. Shestov
Lomonosov Moscow State University, Department of Computational Mathematics and Cybernetics, Laboratory of Computer Systems
e-mail: {hbd,baldis,kost,smel,osmin}@lvk.cs.msu.su

In this paper we address the problem of automatic scheduling of data exchange over a channel with centralized control. Examples of such channels are MIL STD-1553B multiplex data bus and Fibre Channel FC-AE-1553 channel. Scheduling algorithms implementing greedy strategy and ant colony method are presented and experimentally compared. These algorithms support customization for specific requirements of the target onboard real-time avionic system. A technology for data exchange scheduling is proposed and implemented in a tool system.

## 1 INTRODUCTION

Most modern onboard real-time avionics (RTA) systems are distributed systems. They include sensors, control nodes, computational nodes, actuators, data storage and display devices connected by communication channels. An architecture based on channels with centralized control (CC channels) is widely used in RTA systems. Examples of CC channels are MIL STD-1553B, STANAG 3910, and Fibre Channel FC-AE-1553 [1]. In this paper we focus on the MIL STD-1553B bus [2].

Data exchange over a CC channel constitutes a sequence of application and service data transfers between devices attached to the channel. From scheduling point of view, each transfer is a *job*. Data exchange schedule defines start time for each job. Static scheduling strategy is typically used to schedule data exchange over CC channels in RTA systems. According to this strategy, scheduling is performed offline and the resulting schedule can not be changed in runtime. The schedule is executed by the controller, which is one of the devices attached to the channel.

In RTA systems, the number of jobs to be scheduled reaches several hundreds. Technological requirements defined by specifics of RTA system's hardware and software are applied to the schedule. These requirements define a set of constraints on the schedule. A combination of high number of jobs, high channel load, and complicated constraints makes manual schedule construction infeasible. Construction of a correct schedule, which includes all jobs and meets all constraints, is a complex task and needs automation.

In this paper we present a technology for automatic scheduling of data exchange for CC channels. The technology includes a workflow, scheduling algorithms, and a tool system. Section 2 provides a brief survey of related work in the area of interest. Section 3 presents a proposed workflow for static scheduling of data exchange and states the requirements for a tool system to support this workflow. Section 4 introduces the main principles for organization of data exchange over a CC channel and gives examples of requirements to data exchange. Section 5 describes the algorithms developed for scheduling of data exchange. Section 6 presents the tool system for automatic generation of data exchange schedules and lists its subsystems.

## 2 RELATED WORK

There are a significant number of papers that mention applications of CC channels in avionics systems. For instance, [10] describes an X-ray all-sky monitor for use on the International Space Station (ISS), which is attached to the MIL STD-1553B bus of the ISS. However, these papers provide little or no information on algorithms and tool systems used to build the data exchange schedule, or on the structure of the schedule. Technical papers that present features of bus controller interface cards also give no such information.

[1] This work is partly supported financially by the Russian Foundation for Basic Research under grants 09-07-09230-моб_з, 07-01-00237-а.

A variety of approaches have been proposed for constructing a schedule of periodic computational tasks in avionics systems (for instance [11]); these approaches cannot be directly applied to schedule data exchange over a CC channel since they do not deal with specific constraints on the schedule, including constraints presented in Section 4.3.

## 3 WORKFLOW FOR DATA EXCHANGE SCHEDULING

The proposed technology for data exchange scheduling assumes the following workflow:

1. Creation of the project: filling the database with information on structure of the onboard network and characteristics of workload for data exchange channels.
2. Automatic construction of a data exchange schedule which is *complete* (includes all data exchange jobs), and *correct* (meets all constraints, including those imposed by technological requirements); optional manual correction of the schedule.
3. In case a complete and correct schedule cannot be constructed: automatic correction of technological requirements for data exchange, so that such schedule can be constructed with updated requirements.
4. Generation of software code which defines the schedule for the devices attached to the channel.
5. Generation of reports on input data (see step 1) and constructed schedules, for inclusion in documentation on the RTA system.

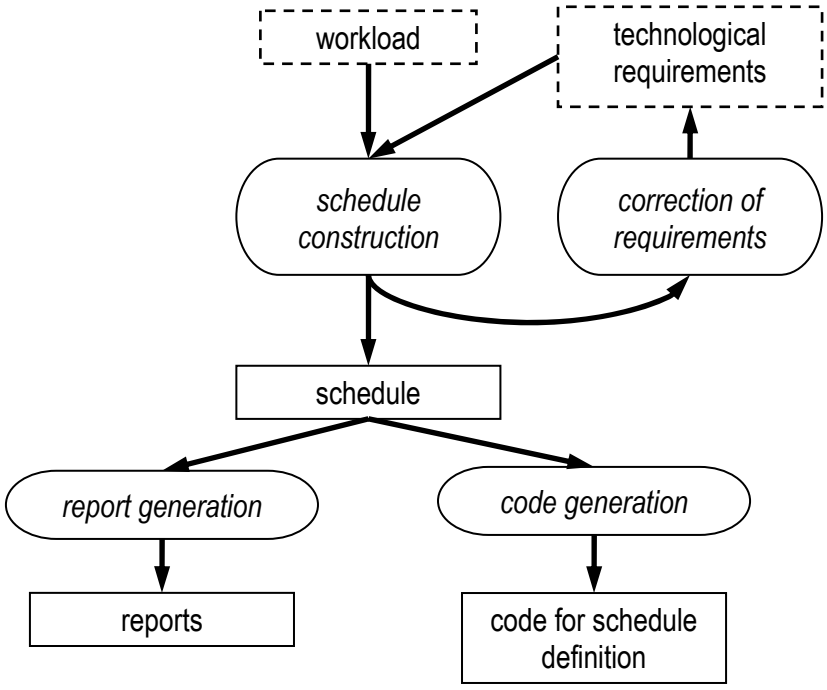



Figure 1. Exchange scheduling for a single channel

Figure 1 shows a workflow diagram for scheduling data exchange over a single channel. Main categories of data are shown as rectangles (dashed for input data, solid for output data). Workflow steps are shown as rounded rectangles.

According to the steps 1-5 listed above, a tool system for data exchange scheduling should support the following activities:

- entering and storing the input data;
- automatic construction and storing the data exchange schedules;
- automatic correction of technological requirements to data exchange (in case it is impossible to construct a correct schedule with current requirements);
- visualization and manual correction of the schedule;
- generation of schedule definition code for devices of the target RTA system;
- generation of reports on input data and constructed schedules.

Data exchange scheduling tools can be applied on different stages of RTA system development, from conceptual phase to industrial prototyping. Applying the tools on the conceptual phase enables development of consistent specifications of information interchange for different devices attached to CC channels. Applying the tools on prototyping phase automates the development of system software for data exchange control. The tools can also support upgrading the RTA system by constructing updated data exchange schedules in case new devices are connected to the channel, existing devices are replaced by new ones, or application software is modified.

To be applicable to development of RTA systems with different basic hardware and software, or different subsystems of a complex RTA system, scheduling tools must support customization according to the specifics of the RTA system. This includes support for:

- considering different technological requirements during schedule construction;
- customization of the template for generated source code;
- customization of the templates for reports;
- schedule construction for different standards of CC channels [1].

## 4 ORGANIZATION OF DATA EXCHANGE OVER A CHANNEL WITH CENTRALIZED CONTROL

### *4.1 Cyclic scheme of data exchange*

Data exchange over a CC channel constitutes a sequence of application and service data transfers between devices attached to the channel (terminal devices). CC channel can have a bus topology (MIL STD-1553B [2]) or a ring topology (FC-AE-1553 [3]). Each device can operate as a data source and/or a data receiver.

One of the terminal devices is the channel controller. It manages the data exchange and monitors the state of other terminal devices. Only the controller can initiate data exchange over the channel, other terminal devices execute the commands issued by the controller (request/response scheme). This guarantees absence of collisions. The controller operates according to the exchange schedule defined during development of the RTA system. Data exchange is performed asynchronously with execution of primary functions of the terminal devices.

An RTA system can support several modes of operation, with different corresponding sets of data transfer jobs. In this paper, data exchange scheduling problems for different modes are considered separately. For each mode, a schedule is constructed which cannot be changed in runtime. The schedule executed by the controller changes only when the RTA system operation mode is changed.

Cyclic scheme of computation is typical for modern RTA systems. According to it, each RTA system device executes its software in a loop, on each iteration of which a block of output data is prepared for sending through the channel, in form of one or several *messages*.

Cyclic scheme of data exchange corresponds to the cyclic scheme of computation. For this scheme of exchange, the workload for a CC channel is a *set of messages*, each of which is intended for periodic transfer through the channel. For each message, its duration and required frequency of transfer is given. Several exchange jobs correspond to a single message. Data to be transferred in a message is computed during the RTA system operation. Duration $l_{\text{int}}$ of the *scheduling interval*, for which the data exchange schedule must be constructed, equals to the least common multiple of the messages' periods (period is a reciprocal value of the frequency). In the RTA system runtime, after the end of the scheduling interval, the schedule is executed again. Number of data exchange jobs corresponding to a given message equals to quotient of $l_{\text{int}}$ and the message's period. Section 4.2 shows how the jobs are formed for a message and how their deadlines are defined.

For cyclic scheme of data exchange, the scheduling interval is divided into intervals of equal duration, known as *subcycles* (see Figure 2). In each subcycle, one *job chain* can be executed (job chain is a sequence of jobs executed without intermediate delays). In the beginning and the end of the subcycle there are intervals on which no job can be scheduled.

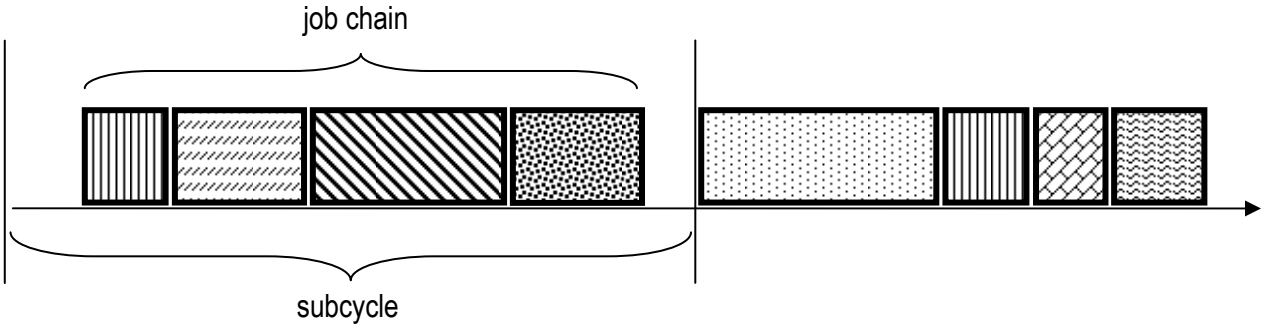



Figure 2. Cyclic scheme of data exchange

*4.2 Forming data exchange jobs for messages*

From scheduling point of view, each message is described by its duration and period (or frequency). For some messages, *phase offsets* can be defined that narrow down deadline intervals of corresponding jobs.

A set of data exchange jobs corresponds to each message. Deadline intervals for the jobs are defined by period and phase offsets of the message. Figure 3 shows deadline intervals (bold lines) for three sequential jobs of a single message, which has period $T$, phase offsets $\varphi_1$ и $\varphi_2$. Each job must be executed within its deadline interval.

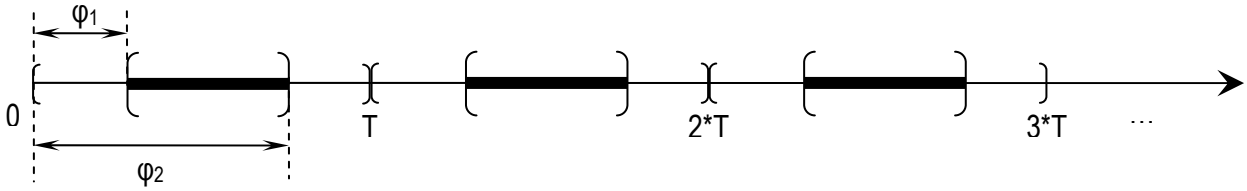



Figure 3. Deadline intervals for jobs

*4.3 Constraints on the schedule*

A correct data exchange schedule must meet a number of constraints, including:

1) general constraints:
   a) absence of collisions (execution intervals of jobs must not intersect);
   b) each job must be executed within its deadline interval;
2) constraints corresponding to technological requirements to data exchange, for instance:
   a) subcycle duration;
   b) duration of the reserved time interval in the beginning of the subcycle;
   c) duration of the reserved time interval in the end of the subcycle;
   d) maximum allowed number of jobs in a chain;
   e) constraints on ordering of jobs within a subcycle (e.g. a job of message $M_2$ must be scheduled exactly after a job of message $M_1$);

A particular set of constraints on the schedule depends on specifics of hardware and software used in the target RTA system. As a result, an important requirement to the scheduling algorithm is its ability to be customized to support different sets of constraints.

## 5 SCHEDULING ALGORITHMS

### *5.1 Greedy scheme for data exchange scheduling algorithm*

The problem of scheduling data exchange over a CC channel belongs to the NP-hard complexity class. This justifies the use of heuristic algorithms to solve the problem. Basic scheduling algorithm applied within the proposed technology is the greedy algorithm. It takes the following data as input:

- set of jobs $V = \{v_i\}_{i=1}^{N_V}$, for each of which is given:
  - duration $t_i$;
  - deadline interval $[s_i; f_i)$;
- technological requirements to data exchange.

Output of the algorithm includes:

- schedule: $S \subseteq V$, with start time assigned to each job $s \in S$;
- list (set) of excluded jobs: $U = V \setminus S$.

Let us denote by *t* the earliest point of time on which scheduling of jobs is allowed. The scheme of the greedy algorithm is as follows:

1) *t* := *0*;
2) select, according to a given deterministic criterion, a job which can be started at *t* without violation of constraints on the schedule;
   if there are no such jobs, set *t* equal to the earliest moment of time on which some job can be started (only increase of *t* is allowed), and go to step 5;
3) put the selected job on schedule, with start time *t*;
4) set *t* equal to the finish time of the scheduled job;
5) take account of constraints on the schedule defined by technological requirements to data exchange:
   5.1) update the value of *t*;
   5.2) update the deadline intervals of jobs not put on schedule;
6) put on the list *U* of excluded jobs those jobs that are not put on schedule and have deadline interval shorter than duration;
7) if there are any jobs not put on schedule and not excluded, go to step 2, else stop.

This scheme guarantees that the constructed schedule is correct, i.e. meets all constraints. However, the schedule can be incomplete (some jobs are excluded, $U \neq \varnothing$). The objective function to be maximized by the greedy algorithm is the number of scheduled jobs.

Customization of the greedy scheme to support specific technological requirements is performed by including procedures, which take account of corresponding constraints on the schedule, to steps 2 and 5. For instance, after the current job chain reaches maximum number of jobs, on step 5 variable *t* is set to start time of the next subcycle. In this way the set of supported technological requirements can be extended without modification of the greedy scheme.

Specifics of the channel workload (set of jobs) can be considered in the criterion of job selection on step 2. Examples of such criterion are *earliest deadline first* heuristic (EDF; job's priority is inversely proportional to the finish deadline of the job), and *rate monotonic* heuristic (RM; job's priority is directly proportional to frequency of the corresponding message) [5]. Computational complexity of a greedy scheduling algorithm depends on the criterion for job selection on step 2. For EDF and RM criteria, the algorithm's complexity is $O(N_V^2 * \log(N_V))$.

The scheme described above, and an algorithm based on it [4], were initially developed for solving the problem of scheduling data exchange over MIL STD-1553B bus. The same scheme can be applied to solving similar problems for other CC channels. In particular, a version of the algorithm was developed for FC-AE-1553 channel with ring architecture [9]. The main specific feature of this algorithm is support for pipelined transfer of data.

*5.2 Ant colony algorithm for data exchange scheduling*

Although the greedy scheme described above enables support for a wide variety of technological requirements to data exchange, and can be tuned for specific problems, the algorithm based on this scheme requires the criterion for job selection (see step 2) to be defined in advance. For each heuristic criterion there is a set of input data for which a correct and complete schedule exists but cannot be constructed by greedy algorithm based on this criterion. This is caused by deterministic nature of the greedy algorithm: on each iteration, the criterion deterministically selects one job to be put on schedule. Scheduling of this job on current iteration may lead to impossibility to schedule some other jobs later.

There are at least two ways to fix this disadvantage:

- apply a limited enumeration of jobs on step 2 instead of greedy heuristic or in combination with it;
- apply a nondeterministic criterion of job selection, which can select with some probability any job from the set of jobs not put on schedule.

We propose to follow the second way by using a hybrid scheme developed by V. Kokarev, which combines the greedy scheme described above with the ant colony method [6]. Ant colony method is used to find the ordering of jobs in the schedule, and steps 3-5 of the greedy scheme are invoked to put jobs on the schedule.

Ant colony algorithm (ant algorithm) searches a given graph for paths that meet certain criteria and have maximum value of the objective function. During the execution of the algorithm, partial paths (sequences of adjacent vertices) are built based on value of a heuristic criterion (local objective function, similar to a greedy heuristic criterion) and on amount of pheromone on edges ("experience" from previous iterations). Next vertex of a path is selected by the roulette scheme with probability proportional to the value of local objective function and to amount of pheromone on the edge leading to the vertex.

Ant algorithm supports automatic adaptation to specifics of the problem by marking up the graph's edges. The markup is used for construction of the solution on each iteration, and is refined while the number of iterations grows.

To apply an ant algorithm to construction of data exchange schedule, we have to convert the scheduling problem to a problem of finding a path with certain properties in a graph. Input and output data for the algorithm are the same as for the greedy scheme (see Section 5.1). Let us denote the number of subcycles in the scheduling interval as $N_{SC}$. The ant algorithm will deal with a fully connected graph, each vertex of which falls into one of the categories:

1) vertices $r_1,\ldots,r_{N_V}$, corresponding to jobs;

*2)* vertices $w_1,\ldots,w_{N_{sc}}$, corresponding to subcycles.

The ant algorithm searches this graph for paths meeting the following constraints:

- the path passes through each vertex exactly once;
- the path begins with $w_1$;
- the path can pass through $w_i$ only after passing $w_1,\ldots,w_{i-1}$.

The objective function to be maximized by the algorithm is quotient of the number of scheduled jobs and the number of jobs in the input set *V*. The schedule is build along with construction of the path. Passing through the vertex $w_i$ corresponds to starting a job chain for the *i*-th subcycle. Passing through the vertex $r_j$ corresponds to adding the job $r_j$ to the end of job chain of the current subcycle (if the resulting partial schedule is correct) or to putting $r_j$ on the list *U* of excluded jobs (otherwise). A job is put on schedule by a procedure that includes steps 3-5 of the greedy scheme for schedule construction (see Section 5.1). Local objective function guarantees that probability of selecting a vertex for a job that can be correctly scheduled (added to the current partial schedule) is higher than probability of selecting a vertex for a job that cannot be correctly scheduled.

Scheme of the proposed ant algorithm is as follows:

1) construct several paths according to the following procedure:
   1.1) set the first vertex: $w_1$;
   1.2) determine the set of vertices to each of which a pass from the current vertex is allowed according to constraints on the path;
   1.3) calculate the probability of pass to each of the vertices from this set, according to value of the local objective function and amount of pheromone on corresponding edges;
   1.4) choose the next vertex of the path by the roulette scheme, according to calculated probabilities;
   1.5) update the partial schedule:
      (a) if the chosen vertex corresponds to a job:
         (i) put the job on schedule (if this job can be correctly scheduled);
         (ii) put the job on the list *U* of excluded jobs (otherwise);
      (b) if the chosen vertex corresponds to a subcycle: switch to the next subcycle;
   1.6) if the path construction is not finished, go to step 1.2;
2) calculate the objective function for constructed paths;
3) update the amount of pheromone on the edges belonging to constructed paths, depending on values of the objective function on these paths;
4) if the termination condition is not met, go to step 2, else stop.

Termination condition for the ant algorithm is usually exceedance of given maximum number of iterations without improving the value of the objective function.

Ant algorithm's ability of automatic adaptation to a specific problem comes at the expense of higher computational complexity than that of an algorithm based on the greedy scheme. Computational complexity of constructing a single path (and corresponding schedule) is close to complexity of a greedy scheduling algorithm.

### *5.3 Experimental comparison of data exchange scheduling algorithms*

An important characteristic of a set of exchange jobs is *channel load*, expressed by the formula:

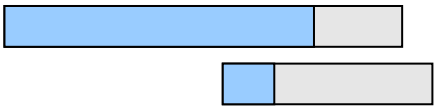
$$E = \frac{1}{l_{\text{int}}} * \sum_{i=1}^{N_V} t_i$$

Computational experiments were performed to compare the ant algorithm introduced in Section 5.2 to algorithms based on greedy scheme (see Section 5.1) with following heuristic criteria for job selection:

1) $H_1$: prefer jobs with minimum earliest finish time;
2) $H_2$: prefer jobs with minimum finish deadline (EDF).

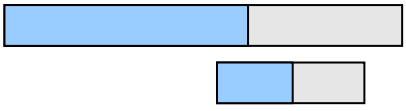
Two classes of input data were considered. First class contains pairs of jobs such that the deadline interval for the second job begins closely to the middle of deadline interval for the first job, and finishes after the end of the deadline interval of the first job; earliest finish time for the first job is greater than that for the second job. For example:

- $t_1 = 70$ , $s_1 = t$ , $f_1 = t + 90$;
- $t_2 = 10$ , $s_2 = t + 50$ , $f_2 = t + 100$.



The second class of input data contains pairs of jobs such that the deadline interval for the first job belongs to the deadline interval for the second job; earliest finish time for the first job is lesser than that for the second job. For example:

- $t_1 = 60$ , $s_1 = t$ , $f_1 = t + 100$;
- $t_2 = 20$ , $s_2 = t + 50$ , $f_2 = t + 90$.



From each class of input data, sets of jobs were considered with different values of channel load *E*. These sets of jobs correspond to sets of periodic messages. Values in tables 1 and 2 refer to the quotient of the number of jobs put on schedule and total number of jobs in the input set. Values are taken for different algorithms and different channel loads.

Table 1. Experimental results, data from class 1

| | Ant algorithm | Greedy algorithm, $H_1$ | Greedy algorithm, $H_2$ |
|---|---|---|---|
| $E$ = 0.1 | 1 | 1 | 0,6 |
| $E$ = 0.2 | 1 | 1 | 0,59 |
| $E$ = 0.3 | 0,99 | 1 | 0,57 |
| $E$ = 0.4 | 0,99 | 0,99 | 0,57 |
| $E$ = 0.5 | 0,97 | 0,96 | 0,56 |
| $E$ = 0.6 | 0,95 | 0,96 | 0,56 |
| $E$ = 0.7 | 0,94 | 0,93 | 0,55 |
| $E$ = 0.8 | 0,92 | 0,92 | 0,54 |
| $E$ = 0.9 | 0,91 | 0,9 | 0,52 |

Table 2. Experimental results, data from class 2

| | Ant algorithm | Greedy algorithm, $H_1$ | Greedy algorithm, $H_2$ |
|---|---|---|---|
| $E$ = 0.1 | 1 | 0,56 | 1 |
| $E$ = 0.2 | 0,99 | 0,55 | 1 |
| $E$ = 0.3 | 0,99 | 0,56 | 0,98 |
| $E$ = 0.4 | 0,97 | 0,55 | 0,97 |
| $E$ = 0.5 | 0,95 | 0,54 | 0,95 |
| $E$ = 0.6 | 0,94 | 0,53 | 0,93 |
| $E$ = 0.7 | 0,93 | 0,53 | 0,92 |
| $E$ = 0.8 | 0,92 | 0,51 | 0,91 |
| $E$ = 0.9 | 0,92 | 0,5 | 0,92 |

The experiments demonstrate that fraction of scheduled jobs for greedy algorithms considerably depends on the class of input data. At the same time the ant algorithm constructs almost complete schedules on different classes of data.

It must be noted that in practice of RTA system development it is essential to put *all* data exchange jobs on schedule, resulting in a complete schedule. Manual fine tuning of greedy algorithms to specifics of a class of input data enables construction of complete schedules for high channel loads (value of *E* up to 0.7). An important problem is to improve the ant algorithm, in order to enable construction of complete schedules with high channel loads, for different classes of input data.

### *5.4 Algorithm for modifying requirements to data exchange*

In case the scheduling algorithm does not construct a correct and complete schedule for a given set of jobs and technological requirements to data exchange, there are the following alternatives:

1) use another scheduling algorithm;
2) selectively exclude some jobs from the set of jobs;
3) modify the technological requirements to data exchange.

Examples of modification of technological requirements are: decreasing the duration of reserved interval in the beginning of a subcycle, or increasing the maximum allowed number of jobs in a chain. In both cases, the *value* of a requirement is changed.

An algorithm was developed for modification of technological requirements to data exchange [8]. This algorithm searches for new values of requirements for which a correct and complete schedule can be constructed by the given scheduling algorithm. The objective function to be minimized is a weighted sum of

changes of requirements' values. The algorithm for modification of requirements combines the strategy of random search with narrowing of search area, and strategies of directed search.

## 6 TOOL SYSTEM FOR DATA EXCHANGE SCHEDULING

Proposed technology for data exchange scheduling is implemented in the Scheduler CAD tool system [7] which is successfully applied to development of RTA systems. The tool system has a modular structure and includes following key components:

- project database which keeps information on the structure of onboard network, characteristics of channel workload, and constructed schedules;
- subsystem for entering and correcting the input data;
- subsystem for automatic construction of schedules;
- subsystem for automatic modification of technological requirements to data exchange;
- subsystem for visualization and manual correction of schedules;
- subsystem for report generation;
- subsystem for generation of schedule definition code for RTA system devices.

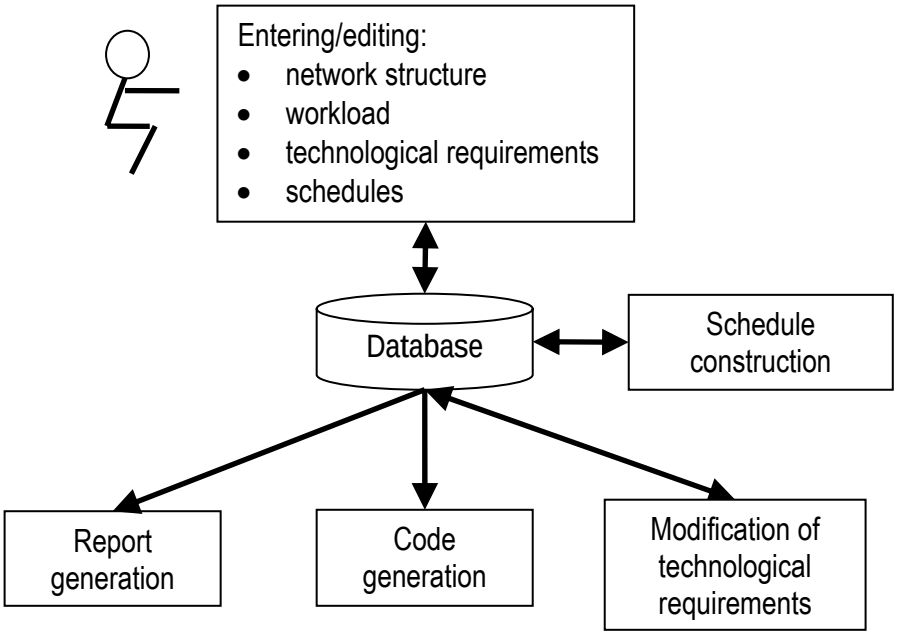



Figure 4. Structure of the tool system

## 7 CONCLUSION

In this paper we presented the technology for scheduling of data exchange through a channel with centralized control (CC channel). MIL STD-1553B was considered as an example of CC channel. One of the basic requirements to algorithms and tools for scheduling of data exchange is support for customization for specifics of a particular target RTA system. A greedy scheme of scheduling algorithm was presented that meets this requirement. A new hybrid algorithm was proposed that combines the greedy scheme and ant colony method, and enables automatic adaptation to the specifics of a particular set of input data.

The presented technology was implemented in the Scheduler CAD tool system which was successfully used for generation of data exchange schedules in RTA systems. Future work on the technology includes its extension for advanced standards of CC channels, including Fibre Channel FC-AE-1553.

REFERENCES


1. “Guide to Digital Interface Standards for Military Avionics Applications“, Technical report ASSC/110/6/2-Issue 3, Avionics Systems Standardization Committee, 2006.
2. U.S. Standard MIL STD-1553B “Aircraft internal time division command/response multiplex data bus”. U.S. Department of Defense, Washington D.C., 1978.
3. Information Technology – Fibre Channel – Avionics Environment – Upper Layer Protocol and Profile based on MIL-STD-1553B Notice 2 (FC-AE-1553). Technical Report INCITS/TR-42:2007. INCITS, 2007.
4. V.A. Kostenko, E.S. Guryanov, “An algorithm for scheduling exchanges over a bus with centralized control and an analysis of its efficiency”. Programming and Computer Software, Vol. 31, No. 6, 2005, pp. 340–346.
5. C.L. Liu, J.W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment”. Journal of the ACM.1973. 20. № 1. P. 46 – 61.
6. M. Dorigo, V. Maniezzo, A. Colorni, “Optimization by a Colony of Cooperative Agents”. IEEE Trans. on Systems, Man and Cybernetics. Art B. 1996. V.26. №1. P. 29-42.
7. V.V. Balashov, V.A. Kostenko, R.L. Smeliansky, “A Tool System for Automatic Scheduling of Data Exchange in Real-Time Distributed Avionics Systems”, in Proc. of the 2nd EUCASS European Conference for Aerospace Sciences, Brussels, Belgium, 2007.
8. V.V. Balashov. “Recommendation Generation Algorithms for Scheduling of Data Exchange through a Channel with Centralized Control”. Journal of Computer and Systems Sciences International, Vol. 46, No. 6, 2007, pp. 887–894.
9. I.A. Bychkov, V.A. Kostenko. “Problems of data exchange scheduling and network topology selection for arbitrated loop”, in Proc. of the 3rd Conference for Methods and Tools for Information Processing, Moscow, Russia, 2009 [in print].
10. H. Negoro, M. Kohama, T. Mihara et al, “MAXI software system: photon event database”. In Proc. of Astronomical Data Analysis Software and Systems XIII Conference, ASP Conference Series, Vol. 314, 2004.
11. I. Bate and P. Emberson, "Design For Flexible And Scalable Avionics Systems". In Proc. of IEEE Aerospace Conference, 2005.