KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Denis Strukov

Investigation and realization of the high school schedule optimization model in the internet environment

Magistro darbas

Darbo vadovas

prof. habil.dr. J. Mockus

Kaunas, 2010

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Denis Strukov

Optimizacinio mokyklos tvarkaraščio modelio sudarymas ir realizavimas interneto aplinkoje

Magistro darbas

Recenzentas
dr. Lina Pupeikienė
2010-05

Vadovas
prof. habil.dr. J. Mockus
2010-05

Atliko
IFM-4/1 gr. stud.
Denis Strukov
2010-05-01

Kaunas, 2010

Table of contents

Table of figures

Strukov D. Investigation and realization of the high school optimization model in the internet environment; Master's paper in science of informatics, supervisor Prof. J. Mockus; Department of Software Engineering, Faculty of Informatics, Kaunas University of Technology, 2009.

SUMMARY

Every school, college and university has to deal with the scheduling task once or more times a year. The school schedule creation on its own is a very difficult task, furthermore, to make it convenient for students as well as for teachers makes it even more complex.

The main aim of this paper is to present a flexible software system that is capable of creating a schedule for a secondary school using the latest technologies and distributed programming techniques.

The literature analyzed in the paper is related to solving the tasks of school scheduling. Contemporary solutions are reviewed. The paper also contains analysis of existing commercial and non commercial solutions. The new software system is presented. Its advantages and disadvantages are analyzed. The optimization methods that can be used in school schedule optimization are analyzed and some of them are employed in the new software prototype in order to achieve better results.

PREFACE

School schedule creation is a difficult task that every school has to deal with on a regular basis. In fact, it is relevant to any educational institution in any country. The problem of schedule creation is that there is no single and uniform way to do that. Every country, or even different institutions in the same country, may have distinct regulations, procedures and rules on how a schedule has to be created and what is the criterion that makes it a good schedule.

All secondary schools in Lithuania are very flexible, which means that the education program of an individual student depends on his personal preferences and capabilities. This set up makes it nearly impossible to divide students into uniform classes. As a result, every student will end up with an individual study plan and a personal schedule.

A school schedule has to comply with physical restrictions such as:

- a teacher can only teach one lesson at the same time,
- a student can only attend one lesson at a time,
- a room can only be used for one lesson at a time,
- etc.

It also has to satisfy certain requirements, set by the ministry of education, such as:

- a maximum number of study hours per day/week,
- no extended gaps between lessons (maximum of 2 gaps),
- etc.

In addition to the above, there are teachers who also would like to have convenient working conditions and their requirements met.

This level of complexity makes a creation of a convenient school schedule a non-deterministic polynomial-time hard problem (NP-Hard) (Mockus, 2006). There are no methods that can be used to solve this problem in polynomial time. The best option is to use different heuristic methods to produce hopefully the closes to the best possible solution.

1. PROBLEM DOMAIN ANALYSIS

1.1. PROBLEM

Scheduling is a very common and difficult problem. People come across it in a variety of fields such as industry, IT, educational, logistics and even everyday life of individual people. But having just any schedule is not good enough. Optimal scheduling can increase productivity and decrease time needed to accomplish certain series of tasks. That is the reason why the optimal scheduling problem remained popular for decades.

In this paper a specific problem of optimal school scheduling is addressed. A new scheduling platform prototype is presented which is targeting optimal school schedule creation for Lithuanian schools in a distributed computing environment.

In order to define a school schedule, some basic objects and terms have to be defined. Basic objects include:

- Subjects – the disciplines that are taught at school;
- Teachers – the employees that are able to teach certain subjects;
- Students – the consumers of information;
- Rooms – the specially equipped class rooms;
- Periods – the days of the week (weekdays);
- Time Frames –the timeslots when lessons can be taught (hours);

When these objects are combined, they make a set of more complex entities, which, in turn will be combined and bound by relationships to finally create a school schedule. The main object of a school schedule is a lesson (or an execution). A lesson is dedicated to a subject, needs a teacher, a classroom and a group of students. In other words, an execution (or a lesson) is dedicated to a task and consists of a set of resources that include a server, an execution environment and a set of jobs. The resources are the combinations of some simple objects and a task. The combination of a teacher and a task makes a server, student and a task make a job. The execution environment is a room and may or may not have a task bound to it depending if the room is specialized for an execution of a certain task or subject.

A school schedule can be defined by a set of executions (lessons) distributed in a three dimensional space with periods and time frames on X/Y axis and a Z axis reserved for

simultaneous executions. Of course all objects have to satisfy a certain amount of constraints in respect to each other and the environment they are in.

The complexity of the school schedule creation task directly depends on the amount of constraints applied to the schedule and data that is being processed. There is a basic set of constraints that cannot be ignored. This set includes such physical constraints as:

- A teacher can only teach a single lesson at a given moment of time;
- A student can only attend to a single lesson at a given moment of time;
- There is a maximum limit of hours per day (24);
- A room can be occupied by only one group at a given moment of time.

In additional to physical constraints, there are logical constraints, which must also be satisfied in order to keep the schedule valid. Such constraints include:

- Only students of the same year can study together;
- Only students of the same level (chosen by student) can study together;
- Teachers can only teach subjects and levels that they are competent of teaching;
- Once a group of students is formed for particular subject and a teacher is assigned to them, this set up has to remain stable through the entire schedule. Neither a teacher nor a student group content can change.

Due to the flexibility offered by the Lithuanian education system to the students, school schedules became very complex. The complexity of the schedule also depends on the amount of resources, such as teachers and classrooms, available to the school.

Alongside with the physical and logical constraints that are listed above, there is a list of constraints defined by the ministry of education. In addition to that, every school has its individual rules and needs that they want to meet thus creating more constraints. Most of these constraints do not affect schedules validity, but certainly influence its quality.

1.2. POSSIBLE PROBLEM SOLUTIONS

School schedule creation and optimization is considered to be a problem with a NP-Hard (Non-deterministic Polynomial time) complexity. It is possible to use simple solution search algorithms to create a small school schedule from a small data set, but the time required to solve the problem increases very quickly as the size of initial data set grows.

Since the optimal result is unknown, the only feasible way of creating and optimizing a school schedule would be using heuristics to search for a solution and hope that the solution found will be close to the best.

1.2.1. HEURISTIC OPTIMIZATION ALGORITHMS

One way of finding a school schedule of acceptable quality is to create a valid school schedule and then perform its optimization using heuristic algorithms trying to increase school schedules quality. There are a several algorithms that could be successfully applied to this particular problem. They include, but are not limited to:

- Monte-Carlo algorithm (MC)
- Simulated Annealing algorithm (SA)
- Bayesian algorithm
- Genetic Algorithm (GA)

1.2.1.1. MONTE-CARLO ALGORITHM

Monte-Carlo algorithm will repeatedly produce new valid solutions distinct from each other. Every new solution is evaluated and its quality is compared to the current best result. If the quality of a newly created solution is greater than the current best, then a new solution is recorded as the best and the algorithm continues. The algorithm will stop when a certain stop condition is detected. Stop conditions might include:

- Iteration limit is reached;
- No better results were found in the past several iterations;
- Time limit exceeded;
- Etc.

This algorithm proves to be useful if the time required to generate a new solution is small. In case if a current best solution is modified to create a new result, which might have a better quality, the time spent on modification is relatively small. This is convenient, as a greater number of iterations can be made in the shorter period of time. The down side of this approach is that the solution found will be a local optimum approximation.

1.2.1.2. SIMULATED ANNEALING ALGORITHM

In order to analyze a wider solution domain, a Simulated Annealing algorithm can be employed. It is different from the MC method in the way of how the current best result is

being chosen. The current best result is also a solution that modifications are being made to. In the case of MC, the algorithm moves on to the next result, only if it is better than the current one. The SA algorithm, with a certain probability, will move on to a worst solution and hope that the following modification will result in a much better result. The probability for a cooling schedule is calculated using formula

$$r_i = \begin{cases} e^{\frac{-h_i}{x_1 / \ln(1+x_2 N)}}, & \text{when } h_i > 0, \\ 1, & \text{otherwise} \end{cases} \quad (1)$$

Here x_1 – initial temperature, x_2 – cooling rate, N – iteration number, h – quality difference between the previous result and a newly formed one.

Simulated Annealing algorithms' performance in different problems depends on its two optimization parameters:

- Initial temperature (energy)
- Cooling rate

Optimal values for these parameters are different to each problem. They cannot be uniformly defined for school scheduling problem and will vary with each initial data set. The Bayesian algorithm can be used to search for better SA parameters and optimize the work of a SA algorithm for each school schedule individually.

1.2.1.3. BAYESIAN ALGORITHM

A significant amount of research in this field was done by professor J. Mockus in a book „A set of examples of global and discrete optimization 2“ (Mockus 2004). The main idea of the algorithm is by using Direct Bayesian Approach (DBA) (Mockus 2004) to predict the function value in the next point. It is not feasible to calculate function value at every point. That is why the next observation point is defined by minimizing the risk function R(x). The risk is an expected deviation from the global minimum (Mockus 2004). The simplified version of the risk function is

$$R(x) = \min_{1 \le i \le n} f(x_i) - \min_{1 \le i \le n} \frac{\|x - x_i\|^2}{f(x_i) - c_n} \quad (2)$$

Here $c_n = \min_i z_i - \varepsilon$; $z_i = f(x_i)$; $\varepsilon > 0$ – a correction parameter.

The goal is to minimize the risk function and calculate the actual value of the function only at the point where the prediction is best.

1.2.1.4. GENETIC ALGORITHM

Genetic Algorithms (GA) present a general and visual way of permutation and randomization (Mockus, 2006). GA is a methodology for searching a solution space in the manner similar to the natural selection procedure in biological evolution (Holand, 1975). Each candidate solution is represented by a string of symbols. The set of solutions at some state m is called the population of the m^{th} generation. The population evolves for a prescribed number M of generations. The basic structure processed by GA is the string. Strings are composed of a sequence of characters.

A simple GA consists of:

- One reproductive plan, defined as the fitness proportional reproduction;
- Two genetic operators, defined as crossover and mutation.

The probability of selection is defined in proportion to the fitness of individuals.

During the crossover operation one splits two selected strings at some random position s. Then two new strings are created, by exchanging all the characters up to the split point s. During the mutation operation one alters some string characters at random. A mutation is of l^{th} order, if one changes l elements during one mutation operation. Both the crossover and the mutation are feasible, if they satisfy all constraints. It follows from this definition that GA may be considered as a special case of the Permutation and Randomization. Thus, one may include GA into the framework of Bayesian Heuristic Approach. (Mockus 2004)

1.3. EXISTING SCHEDULING SOLUITIONS

Despite the difficulty of the school schedule creation task, most schools are still making their schedules manually. School staff spends hundreds of man hours to accomplish this difficult task. Fortunately there are software solutions on the market that are able to aid in school schedule creation. Lately more and more schools started using these software solutions. A list of most popular applications includes:

- Mimosa
- aSc TimeTables
- Rector

These are the three leading applications on the Lithuanian market.

1.3.1. MIMOSA

"Mimosa" is a scheduling and course planning software, created by Mimosa Software LTD (Finland), which may be used in any kind of school or university (Mimosa, 2010). The application interface is available in multiple languages. The structure of the application is relatively simple and intuitive to an experienced user. The package allows creation of a schedule automatically, manually or by mixing these techniques. "Mimosa" prevents the creation of conflicts and gives hints of optimal actions when a schedule is being created manually. Initial data input is fairly difficult and requires users to be trained. In order to seamlessly integrate with already in place administration solutions, additional procedures have to be implemented. "Mimosa" developers claim that their software is able to perform optimization procedures on created schedules, but exact algorithms are not published. The cost of a single user license is 500 EUR.

1.3.2. ASC TIMETABLES

The "aSc TimeTables" application, created by Slovakian company Applied Software Consultants (aSc, 2010), is similar to Mimosa software by functionality, but is much more user friendly. It also features a multilingual user interface, a convenient documentation and a user support system. Data input is intuitive and straight forward. Schedules can be created automatically or manually. A manual schedule creation or modification after an automatic creation is made easy by employing drag and drop technique. The application also prevents conflicts in the schedule while the user is making manual adjustments. During the automatic schedule creation, the application claims to minimize the number of gaps in teachers' schedule. No student schedule optimizations are performed. The "aSc TimeTables" software also allows to constrain the schedule with additional constraints such as the definition of days that a particular teacher wants to have free of lessons or the definition of classrooms, that can be used for a particular subject only and so on. The software is supported by most popular MS Windows platforms like MS Windows XP and later versions. The Applications' price ranges from 250 to 830 EUR depending on the version and other options.

1.3.3. RECTOR 3

Another popular software package developed by the Russian company - OptimeSoft, is "Rector 3" (Rector, 2010). It is widely used in Russian schools. The software is available in multiple languages. As well as the other two products described above, the application has such functionality as automatic and manual scheduling, conflict prevention, teachers'

timetables gap minimization and other features. “Rector 3” is also able to work with profiled schools, but it is mainly targeting the Russian education system and may not fit many requirements of other countries. The software is priced at 300 EUR. There is a limited trial version available for download.

1.4. SITUATION IN LITHUANIA

Statistics indicate that in 2008-2009 Lithuania had 1415 secondary schools that were providing education to 464.906 students. All of Lithuanian schools offer high flexibility. It is an enormous task to create flexible schedules manually every year. To simplify the process of scheduling, schools employ specific software. The most widely used software packages in Lithuania are “Mimosa” and “aSc Timetables”.

“aSc Timetables” is able to automatically create schedules of relatively good quality. The simple user interface allows easy adjustments to a generated schedule and prevents conflict occurrences. The data input is relatively simple and allows data import from XML. The application has good user support features. The Work flow is easy and intuitive. The downside of the application is that it does not take student gaps into account. “aSc Timetables” is more suited for the less flexible education systems. The software does not feature advanced post generation optimization algorithms.

The school schedule generated by the “Mimosa” application, most of the time, does not satisfy most of the schools and cannot be used off the shelf. That is why “Mimosa” is used a lot to just assist manual schedule creation instead of generating it automatically. The software proves to be useful in lesson conflict prevention and other constraint monitoring. The user interface is found to be non user friendly and difficult to understand. The data input proves to be complicated and time consuming. The application does not seem to use any advanced optimization algorithms.

There are many other software solutions on the market, but most of them either do not have decent automatic schedule creation features or are very complicated, non user friendly as well as expensive. Alongside with commercial solutions, there are a few open source prototype applications made by Lithuanian students while researching school schedule optimization. These applications were explicitly targeting Lithuanian schools and educational system. To my knowledge there are two base projects currently available:

- “School – Complete”
- “Schools schedule creation and optimization program”

All other applications are derivatives of the above applications.

"School – Complete" was the first attempt (2003) of Lithuanian students to create a school schedule creation application, but due to its complexity of use it was never used in real world environment. This application has two successors, which added additional functionality to the program as well as new schedule optimization procedures. The "School Schedule optimization program by Auris" program modification was the last one to use CSV (Comma separated values) file format as input. The following version called "School Schedule optimization program by Vidunas" added XML input and an optimization method based on genetic algorithm. Later research showed a lack of basic constraints, which means that the application can be used only for research purpose and not in a real school environment. The poor extensibility of the application and a big amount of bugs in the code led to creation of a new school schedule creation and optimization program.

The "Schools schedule creation and optimization program" core was programmed by me as my bachelor diploma work (2007). A web based user interface was developed. Later, the application was completed and extended by Dr. Lina Pupeikiene thus creating a new modification called "Optima". Additional constraints, optimization methods and other features were added. The "Schools schedule creation and optimization program" modification by Kristina (2009) added a Java "Web Start" interface and didactical constraints.

All of the applications, described above, lack a number of necessary constraints and thus are not able to create a valid schedule. The application code analysis revealed programming and logical errors.

1.5. SUMMARY

The school schedule creation and optimization is very difficult and time consuming task. It requires an enormous amount of time and effort to create a school schedule manually. Due to flexibility of Lithuanian education system, not many commercial software solutions may be adopted to automatically create a school schedule of desired quality. That was the reason to create a school schedule creation and optimization software specifically design to fit the requirements of a Lithuanian high school. The attempts made previously resulted in the creation of several distinct software systems described in previous chapter. None of the systems were able to create a valid school schedule. Due to the poor design and architecture of the initial software, each later modification, such as addition of new constraints or optimization method, resulted in the increase of code complexity and the decrease of its maintainability.

In order to achieve better results, a new software system is required. It must feature a better architecture and the design that allows easy future modifications without the increase of complexity. Additionally, the software must be fast. That is achieved by utilizing the power of multiple processors, computers or even computer networks.

2. SCHOOLS SCHEDULE CREATION AND OPTIMIZATION DISTRIBUTED SYSTEM PROTOTYPE

The new system is meant to replace the previous school scheduling platforms. The advanced architecture, based on a newly developed scheduling framework, will simplify further research in the schedule creation and optimization field. In order to minimize the time of experiments, the system employs a specially developed distributed computing subsystem. It also features easy addition of new and modification of existing constraints and other essentials to the research functionality.

2.1. FUNCTIONAL ANALYSIS

The developed schedule creation and optimization system prototype is a fast and expandable platform for research in the scheduling field. It can be used to create school schedules. The application features a web interface and can be accessed through internet.

In order to successfully use the system, the user must be a computer literate and be able to use the Microsoft Office "Excel" application as well as a web browser. To expand the system, one must be familiar with system architecture and be able to program in Java programming language.

2.1.1. CONCEPTUAL OVERVIEW

A general view of the system usability context is presented below (Figure 1)

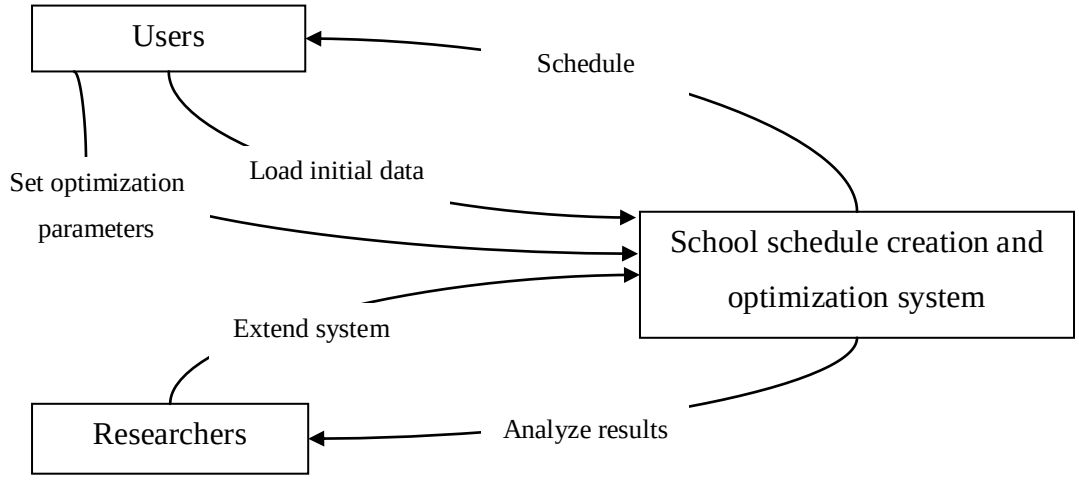

Figure 1 General view of the system usability context

From the users' perspective, a user accesses the web interface of the application and loads the initial data in the form of a MS Office Excel spreadsheet. The user also supplies the optimization parameters that will be used during the optimization. The results are presented to the user at the end of execution.

Researchers are able to make additions or modifications to the system and monitor the actual progress of the schedule optimization. The history of the optimization can also be reviewed and analyzed.

2.1.2. BUSINESS PROCESSES

On a big scale, there is a single business process in the whole application. The business process diagram presented below (Figure 2) shows three steps involved in the process:

- Upload initial data file
- Set optimization parameters
- Retrieve Schedule

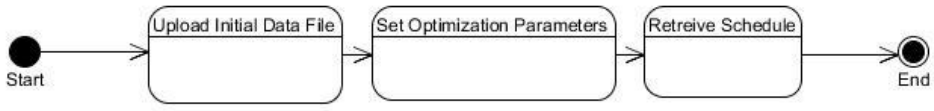

Figure 2 Business process diagram

Each step of the process may be considered to be a single user interface screen.

In the first step, the user is required to upload a specially compiled MS Office Excel file containing the initial data about subjects, teachers, their work hours, students and their study plans. Based on that information a school schedule will be created.

In the following step, the optimization parameter may be customized. In order to evaluate the quality of a schedule, penalty point values are assigned to each constraint type. The optimization method and its parameters are set. The system employs Monte-Carlo algorithm.

Lastly, the resulting schedule is presented to a user along with other information about the optimization process and schedule quality.

2.1.3. SYSTEM COMPONENTS

At the highest level, there are 4 components that make up the system:

- User interface
- Central
- Executing node
- Core application

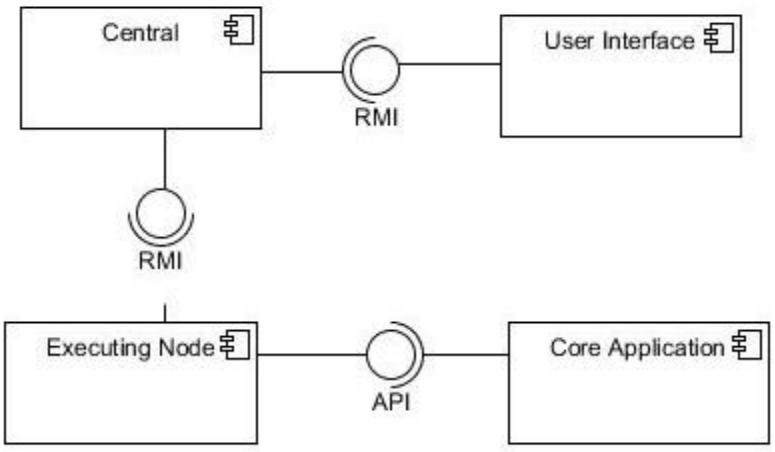

Figure 3 System components

Most of the components presented above are communicating through Java Remote Method Invocation (Java RMI). This system allows communication between distributed applications. Another interface that can be found in component diagram is Application Programming Interface (API). The API is an interface, exposed by an application, which can be directly used by other application.

A web application developed specially for this system is acting as a user interface. It provides the means for transferring the initial data file and optimization parameters to the server. It also acts as a progress monitoring tool and result visualization platform.

The central module is responsible for the work distribution and result evaluation. It is dispatching tasks and collecting results from executing nodes over the network. The results and progress information is stored available in the central module and can be requested at any time by the user through the web interface.

The executing nodes are acting as an interface to the core application. They receive information from the central module and initiate schedule creation and optimization upon request from the central. The nodes also extract information from the core application and report it to the central. Each node is deployed on a separate computer in the network. Several nodes can be deployed on one machine, depending on its processing capabilities and processor core count.

The core application is responsible for the actual creation of the school schedule. It has an API interface that enables the executing nodes to initiate procedures and extract the results. The core is the main and the largest part of the whole system.

2.1.4. CORE APPLICATION STRUCTURE

The core application is build on a specially designed scheduling framework. It extends its functionality to the requirements of a school. The module consists of two packages:

- Scheduling framework (Scheduler)
- School schedule creation platform (School Scheduler)

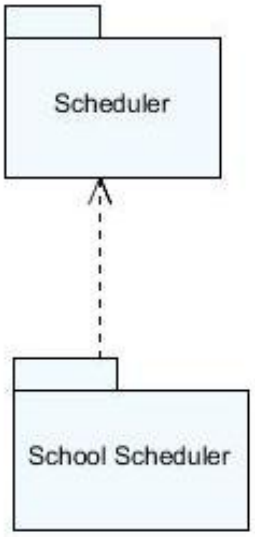

Figure 4 Package dependency diagram

The scheduling framework incorporates the schedule compilation logic. It is responsible for the collection of all required resources for a task execution to be created. In the context of school schedule creation resources are:

- Students
- Teachers

- Rooms

A single task execution can be described as a lesson of a certain subject. The scheduling framework is responsible for the creation of these task executions and their placement in the time frames within periods. Here – periods are days and a time frame is a quantum of a day.

The school schedule creation platform is an extension of the scheduler framework. All the resources described above are defined inside the school schedule creation platform. It also contains constraint definitions and all the rules of the resource interaction. The scheduling framework, by design, is unaware of these details. The framework consists of several packages:

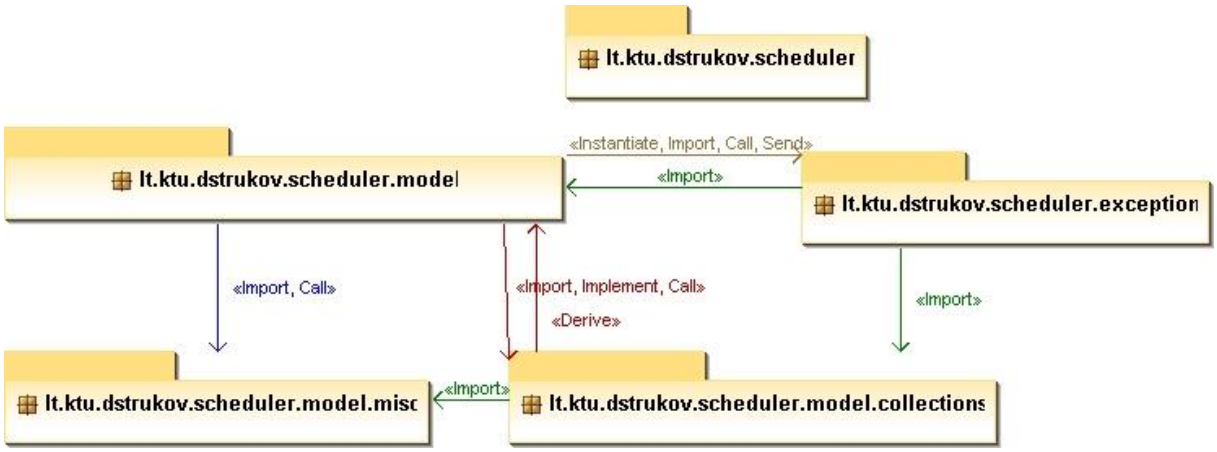

Figure 5 Package dependencies. Scheduler framework

2.1.5. SCHEDULING FRAMEWORK MODEL

The "model" package is the main package in the framework. It contains all the necessary business objects required to define a simple schedule almost in any context. The list of most important objects includes:

- Schedule – represents a schedule;
- Period – represents a day;
- Time Frame – represents a time period within a period;
- Task – represents a type of task that can be executed;
- Resource – represents a resource, required for something to be scheduled;
- Resource Owner – a resource owning object;
- Execution – represents an instance of something that can be scheduled.

All these objects either depend or consist of each other. The dependency diagram is presented below (Figure 5). Some of the helper classes were left out of the diagram for better readability. Some of them are worth of mentioning for further references:

- The "Data" class contains references to all the collections of resources, resource owners, tasks and resource requirement map.
- The "MinMaxRequirement" class, which is mapped to every resource collection, is meant to indicate the minimum and the maximum quantity requirements for each resource in order to create an execution.
- The "AbstractBase" class contains a functionality that is common to almost all objects in the framework.

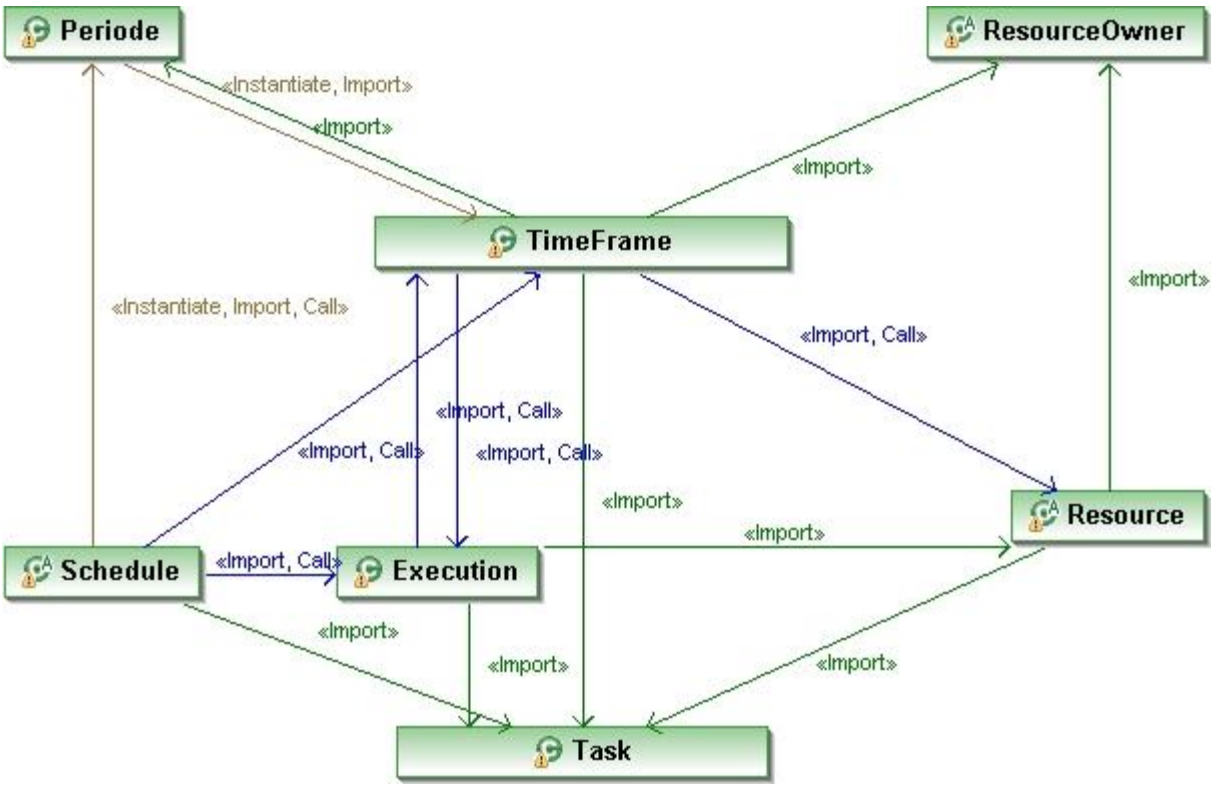

Figure 6 Scheduler framework model. Dependency diagram.

The "Schedule" object is a top level container. It contains collections of periods, time frames and executions. Each of these objects is a container on its own. A "Period" serves as a container for time frames, time frame serves as a container for execution. "Execution" in turn contains a task and a list of resources. Most of the instances hold a reference to each other for easier navigation.

When an instance of a The Schedule class is instantiated, a process of schedule creation is initiated. It is worth to note that Schedule is an abstract class, which means that some of the functionality and parameters have to be defined in a deriving system. For instance

the amount of periods and time frames that have to be instantiated and the penalty function must be defined in the deriving system.

The “Resource” and “ResourceOwner” classes complete the set of abstract classes in the model. As well as the “Schedule” class, these abstract classes require deriving system to provide specific functionality such as resource and resource owner compatibility check routines. Every resource has an owner and a task. The deriving system must specify which resource has to be exhausted before the schedule can be declared complete.

In the model there is another important package containing definitions of model object collections.

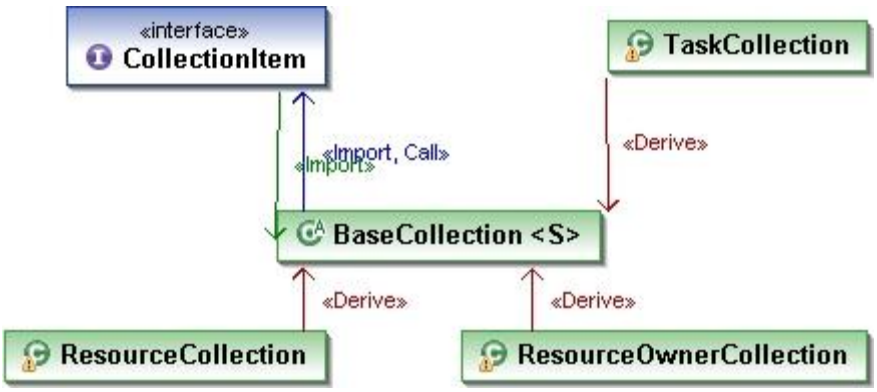

Figure 7 Model object collections. Dependency diagram.

These collections are responsible for storage of resources, resource owners and tasks. They also provide easy access to any of the model objects.

The execution creation process requires access to all the above collections. These are available through the “Data” object that stores required references. An execution object can be created in a given timeframe and for a given task. Additional information, such as resource requirements for an execution and concrete resources themselves must be provided by the deriving system.

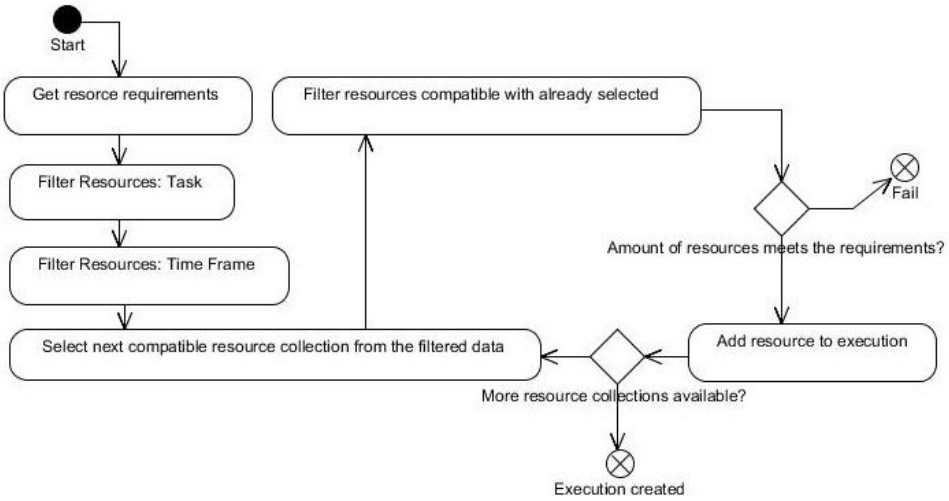

Figure 8 Execution creation process diagram

The above execution process diagram illustrates the algorithm used to create an execution. An execution is being created for a given task and a particular time frame. The first few steps filter the resources that are dedicated to the task and are available in that time frame. Then a loop is being started that picks out the required amount of resources of each type. Every resource has to be compatible with the previously selected resources. The order of resource types is defined in the deriving system. When all the resource requirements are met, execution is being created and placed in the time frame. This event registers all resources in the schedule and removes them from the list of available resources.

2.1.6. SCHOOL SCHEDULE CREATION PLATFORM MODEL

In this paper – a school schedule creation platform is discussed. By extending the framework described in the chapter above, a schedule creation platform may be created for almost any context. For the school schedule context, the list of resources, defined in the system, includes:

- Servers. A single server represents a teacher, teaching a single hour of a particular subject.
- Jobs. Here, a single job represents a student, studying a single hour of a subject
- Environments. An environment, represents a class room.

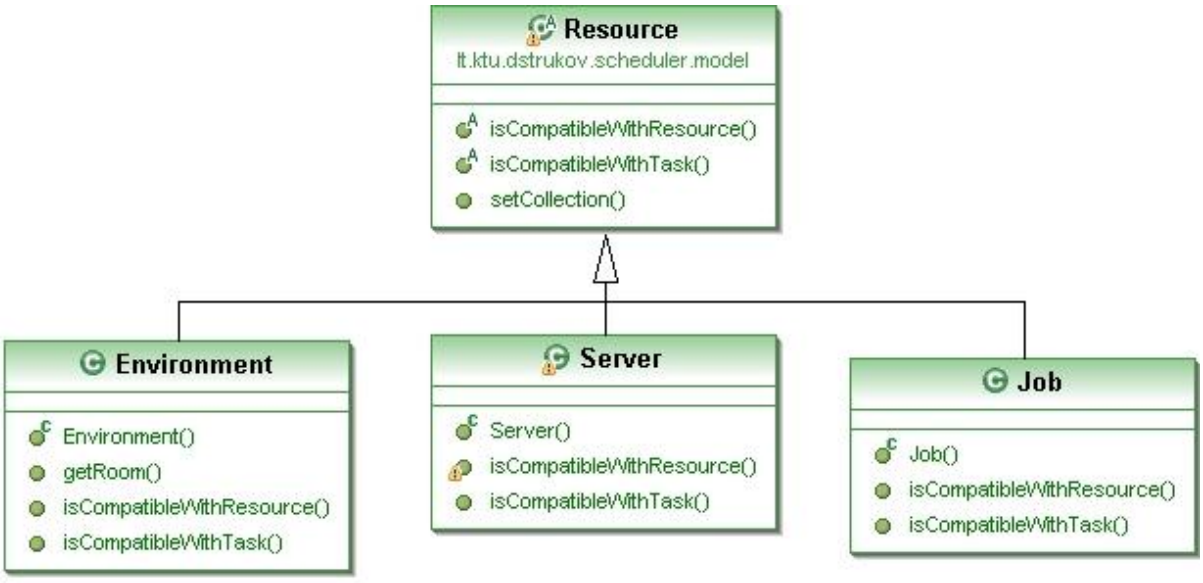

Figure 9 Inheritance diagram. School schedule resources

Every resource has its task compatibility rules and resource compatibility check algorithms. Alongside with individual functionality, every resource must have an owner. The list of resource owners includes:

- Teacher;
- Student;
- Room.

The inheritance diagram below (Figure 10) illustrates the relation of these objects to the scheduling framework.

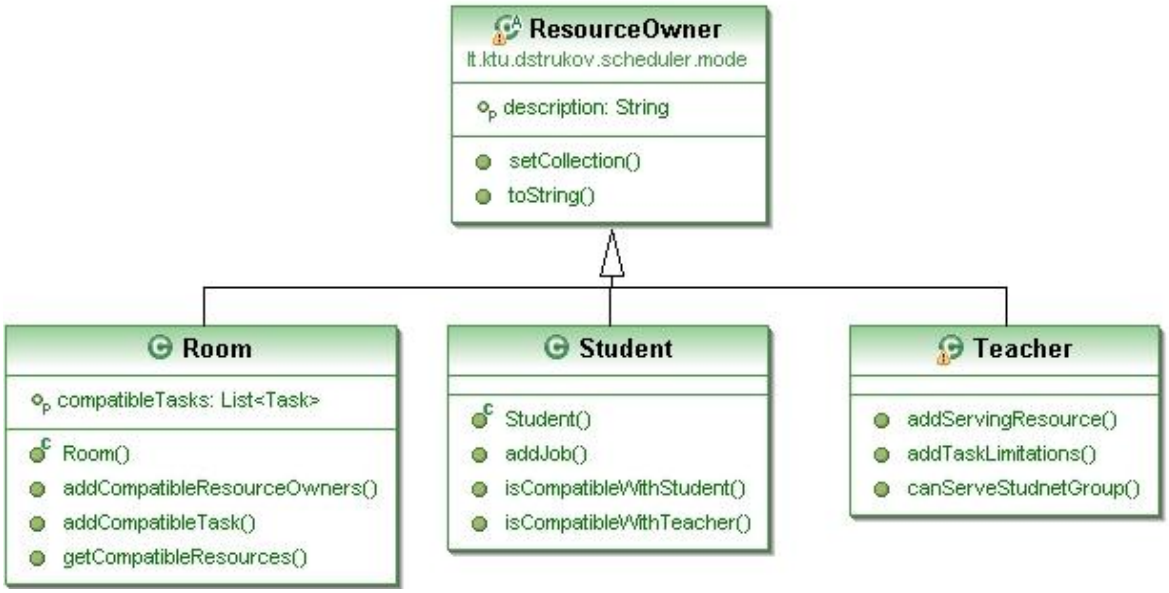

Figure 10 Inheritance diagram. Resource owners

As stated above, every resource has a resource owner. Every resource owner can have multiple resources. A resource can be defined as a single instance of a particular service provided by the resource owner. For example – a study plan of a certain student indicates that a student has to follow 5 hours of mathematics a week. With this set up, we have a single resource owner – a student, and 5 jobs – five combinations of the resource owner and a task (mathematics).

In order for the model to create a valid school schedule, all the compatibility check procedures have to be carefully implemented. Every resource must implement task compatibility and other resource compatibility check procedures.

The school schedule constraints implemented in the school schedule creation platform are individual to each resource. That means that depending on the type of the resource that the check is being executed against, different aspects are being checked. The table below illustrates the complexity of each check. The white cells represent the checks that have almost no logic behind them. The procedures represented by the blank cells marked with a minus sign will always fail the compatibility check. Those marked with the plus sign will always pass the compatibility check. The light grey cells represent the compatibility check procedures that require a single check before making a decision. The dark grey cell requires more than one check and thus is more complex.

	Server	Job	Environment
Server	-		
Job			+
Environment		+	-

Table 1 Resource compatibility checks

In the lists below, a more detail description of all compatibility constraints is presented. Each list describes a one way check against all the other resources, including itself.

The “Server” compatibility checks include:

- With another “Server” – nothing. There is no execution requiring more than one “Server” resource, thus the check procedure is not necessary.
- With a “Job” – verify that the server owning teacher can teach the job owning student.
- Environment – verify that the “Server” owning teacher is allowed to use the “Environment” owning class room.

The “Job” compatibility checks include:

- With a “Server” – in case if the teacher for a particular subject has already been assigned for the “Job” owning student, it has to be the same teacher through the entire schedule teaching that subject.
- With another “Job” – this set up requires multiple checks:
    - The other “Job” owning student must be of the same level.
    - In case if the student group for the particular subject has already been formed, the student has to stay with that group through the entire schedule.
- With an “Environment” – all jobs are compatible with all environments.

The “Environment” resource compatibility check insures that:

- With a “Server” – verify that the class room can be used by “Server” owning teacher.
- With a “Job” – nothing has to be verified. Any “Job” can be executed in the “Environment”.
- With an “Environment” – nothing to verify. There are no executions, requiring more than one environment.

2.1.7. SCHOOL SCHEDULE EVALUATION

The resources described in the previous chapter alongside with the resource compatibility constraints implemented on the scheduling framework, make up a school schedule creation platform that is able to create a valid school schedule while satisfying the basic needs of a basic school. But schools require more than just a valid schedule, they require a good schedule.

The list of factors, that influence the quality of the schedule, include:

- Gaps between the lessons in teachers’ schedule;
- Gaps between the lessons in students’ schedule;
- Violation of the maximum working hours per day for teacher;
- Violation of the maximum lessons per day for student;
- Violation of the teachers day off.

The understanding of a "good" schedule differs from school to school or even from employee to employee within the same school. For some schools, the gap between lessons in a teachers' schedule is not a big deal, but for others it a no-go.

In order to calculate the quality of a schedule, a penalty point system is employed. The more violations are found in the schedule, the more penalty points are assigned to it. The application provides the flexibility of assigning the values of the severity of a certain violation. That enables the system to distinguish between heavy and light violations. With this setup, a penalty function can be derived:

$$P = \sum_i c_i N_i \qquad (2)$$

Here the P is a variable representing the schedule quality; c_i is i-th violations occurrence count in the schedule; N_i is the *i-th* violations severity value assigned by the user to this certain violation. As we can see from the formula – the larger is the value of P, the worse is the schedule.

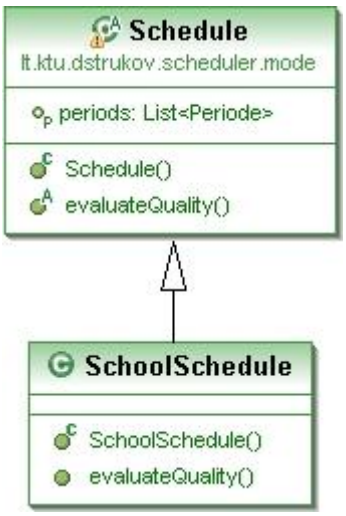

Figure 11 School schedule inheritance diagram

The procedure responsible for calculating the value of the penalty function and evaluating the quality of a schedule is implemented in the model of the school schedule creation platform.

2.2. SUMMARY

The way that a piece of software is designed defines its usability and extensibility. These two qualities are very important and have to be maintained at a very high level in order to release a software packet of good quality. During the creation of the new school schedule creation and optimization software, extensibility and usability were the main attention points.

The application design made of loosely coupled components. That keeps the maintainability of the software at the high level. It also makes possible the modification of each separate component independently.

3. SCHOOL SCHEDULE OPTIMIZATION

Since it is possible to evaluate a school schedule, it should also be possible to make it better. In order to make it better, something has to be improved or in other words changed. By attempting to eliminate the violations that the penalty function is based on, we might reduce the overall penalty count *P* (2). As we have more than one schedule quality factors, the process of school schedule optimization can reach very high complexity levels. In order to keep it simple and effective, a single line of optimization has been chosen – gaps between lessons in teachers' schedule. The modifying algorithm will target specifically this violation and try to close the gaps by moving a whole lesson from some other place in the schedule. When a lesson is being moved, its whole content must move along with it, otherwise it is likely to result in violations of mandatory constraints all over the schedule. If a whole lesson and its content are moved with an exception of the environment, there is a moderate chance of the operation to be valid. This is an important factor in a schedule optimization as the amount of constraints drastically reduces the amount of simple modifications possible. Other possible modifications, that are likely to initiate a chain reaction of modifications, are not feasible due to their complexity and the chance of final modifications to be invalid, thus scrapping all previous operations.

3.1. INITIAL DATA FILE

The initial data file was originally compiled by Dr.Lina Pupeikiene in 2006 for use with the previous versions of the "school schedule creation and optimization program". Since then, the structure of the file has undergone several improvements and modifications.

The file is basically a XML file compiled using the Microsoft Office 2003 standards. It can be created and edited using the MS Office Excel or the OpenOffice Calc applications.

There are several types of data required in order to create a school schedule:

- The data about the subjects taught in school
- The data about the teachers and their working in school
- The data about the students and their study plans
- The data about available rooms

The structure of the workbook consists of several sheets each dedicated to a concrete type of data (Figure 12). The first two sheets are fixed. Starting the third sheet and further, all the sheets are dedicated to the students. The sheet name identifies the student group. The last sheet in the workbook contains a template of the students sheet.

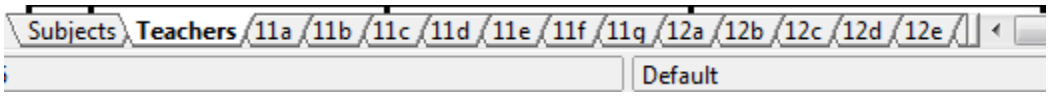

Figure 12 Workbook sheets

The first sheet is dedicated to the subjects. Here, each row contains a single subject definition.

	A	B	C	D
1	Subjects	Groups	Priority	Max. pupils
2	Ps	e1	5	30
3	T	e1	5	30
4	E	e1	5	30
5	Lk	but	5	25
6	A1	e2	3	20
7	V1	e2	3	20
8	R1	e2	3	20
9	Pr1	e2	3	20
10	A2	e3	3	20
11	V2	e3	3	20

Figure 13 Subject definition

In order to define a subject, a subject code, group, priority and a maximum amount of students has to be defined. This data has to be entered to the columns A, B, C, and D respectively (see Figure 13) for each subject.

The subject code acts like a unique identifier for a subject. It has to remain unique through the entire subject list. Same codes will have to be used in the definition of the teachers in order to define the subjects that a teacher can teach.

A subject can also have a group. The subject grouping may be used to introduce extra logic in the schedule creation. This is useful if some subjects have to be scheduled in a particular time frame or order. The subject priority caries out a similar role. It defines the order, in which the subject will be scheduled. The last property in the list defines the maximum amount of students that can attend a lesson of this subject simultaneously

The next sheet in the work book contains information about the teachers (Figure 14).

	A	B	C	D	E	F	G	H	I
1									
2	Grou	Nr.	Subject	Name	Room	Hours	Subj	Can teach groups	Days off
3	1	1	Etika; Tikyba;	Martinaitienė Laura	101; 102; 103;	30;30;	E;T;	11b; 11c; 11d; 11e; 11f; 11g; #11a; 11b; 11c; 11d; 11e; 11f; 11g;	
4		2	Etika; Tikyba;	Patamsienė Jolanta	101; 102; 103;	30;30;	E;T;	11b; 11c; 11d; 11e; 11f; 11g; #11a; 11b; 11c; 11d; 11e; 11f; 11g;	
5		3	Psicholigija	Maciulevičienė Natalija	101; 102; 103;	30;	Ps	11a; 11b; 11c; 11d; 11e; 11f; 11g;	1;2;3;
6	2	4	Leit.k.	Vasiliauskaitė Nijolė	101; 102; 103;	35;	Lk	12a; 12b; 12c; 12d; 12e; 12f;	
7		5	Leit.k.	Lasauskienė Lilija	101; 102; 103;	35;	Lk	11a; 11b; 11c; 11d; 11e; 11f; 11g; 12a; 12b; 12c; 12d; 12e; 12f;	
8		6	Leit.k.	Bundonienė Orinta	101; 102; 103;	35;	Lk	11e; 11f; 12a; 12c; 12d; 12f;	

Figure 14 Teachers data

The first four columns are pure informative and don't carry any critical information for the schedule. They may be used to increase output readability. The next four properties are closely related to each other. The information about subjects that the teacher can teach is entered in the G column. It is worth noticing that a single teacher is able to teach several subjects. In case when a teacher is teaching two subjects, the subject codes are separated with a semicolon (Figure 14, row 3, column G). For each subject, the maximum amount of hours per week has to be defined (Figure 14, column F). The multiple values are separated with semicolons. Alongside with other properties, a list of student groups (column H) and a list of rooms (column E) have to be defined for each subject that a teacher can teach. In case of multiple subjects, the lists are separated by a hash sign (#). The list of student groups defines which student groups can be taught the subject by a particular teacher. Same stands for rooms. Each subject taught by a teacher is restricted to a number of rooms which are listed in column E. The last column in the sheet (column I) defines teachers' desired days off.

Starting the third sheet and further, the workbook contains information about students and their study plans.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH
1	Nr.	Name	Dor.	ugd.		leit.k.	1 užs. kalba				2 užs. kalba				Ist	Geo	Mat	Inf	Biol	Fiz	Cher	Menai				K	Optional							Total
2			E	T	A1	V1	R1	Pr1	A2	V2	R2	Pr2								D	M	Te	S	Iot	p	K	br	ek	pb	PC				
3	1	11a Pavarde 1 Vardas 1	2			5A	5A					2B			4A	2B	3B		2B					3	2B				1				
4	2	11a Pavarde 2 Vardas 2	2			5A	4A						2B		3A	2B	5A	1B	2B					3	2B				1				
5	3	11a Pavarde 3 Vardas 3	2			5A	4A							2B	3A		3B	1B	2B			3			2B	1	1						
6	4	11a Pavarde 4 Vardas 4	2			5A	5A						2B		3A		3B	1B	2B			3			2B								
7	5	11a Pavarde 5 Vardas 4	2			5A	4A								4A		5A	1B	2B				3		2								
8	6	11a Pavarde 6 Vardas 5	2			6A	4A					2B			4A		5A	1B	2B			3			2								
9	7	11a Pavarde 7 Vardas 6	2			5A	4A						2B		3A	2B	4A		2B					3	2B								
10	8	11a Pavarde 8 Vardas 7	2			6A	4A								3A	2B	5A		2B					3	2B		1	1		1			
11	9	11a Pavarde 9 Vardas 8	2			5A	5A					2B			4A		3B		2B					4	2B								
12	10	11a Pavarde 10 Vardas 9	2			5A	4A								3A	3A	5A	1B	2B					4	2B								
13	11	11a Pavarde 11 Vardas 1	2			5A	4A								2B	3A	3B		2B						2B		1		1	1			
14	12	11a Pavarde 12 Vardas 1	2			5A	4A						2B		3A		3B	2A	2B					4	2B								
15	13	11a Pavarde 13 Vardas 1	2			5A	5A						2B		3A	3A	4A		2B				3		2B								
16	14	11a Pavarde 14 Vardas 1	2			5A	5A						2B		4A		3B	1B	2B					3	2B								
17	15	11a Pavarde 15 Vardas 1	2			5A	4A						2B		4A	3A	5A		2B					3	2B								
18	16	11a Pavarde 16 Vardas 1	2			6A	5A							2B	3A		4A	1B	2B			3			2B				1				
19	17	11a Pavarde 17 Vardas 1	2			5A	4A								3A	2B	5A	2A	2B						2B	1							
20	18	11a Pavarde 18 Vardas 1	2			5A	5A						2B		4A		5A	1B	2B					3	2B								
21	19	11a Pavarde 19 Vardas 1	2			6A	4A								4A		6A		2B					3	2B		1						
22	20	11a Pavarde 20 Vardas 1	2			5A	4A					2B			3A		5A		2B						2B		1	1		1			

Figure 15 Student data

Each sheet represents a single student group. The name of the sheet is used as a group name. It also contains the information about the level of students in that group. The group 11b

would consist of students of 11th grade. This is important, as the students from the different levels cannot be studying together.

The students' study plan is entered in a special table which is widely used in the Lithuanian schools. Each student occupies a single row in the table. The first two columns, A and B, contain information about the student. The rest of columns, except the last one, contain an individual study plan for each student. Each column represents a subject. The cell that is located at the intersection of a student row and a subject column may contain a number or a number with a letter. The number represents the amount of hours per week that the student chose to study the corresponding subject. The letter indicates the subjects' difficulty level chosen by the student. If no letter is indicated, the default difficulty level is assumed.

The order in which the subjects are arranged in this table is important. It has to correspond to the order the subjects entered in the first sheet of the workbook.

The examples of full tables with data are available in the appendix.

3.2. USER INTERFACE

The application user interface is implemented using Java Server Faces (JSF) and Asynchronous JavaScript and XML (AJAX) technologies. Such additional tools as Google Charts and DOJO framework were employed in order to not reinvent the wheel and aid the development procedure.

The user interface is implemented in a form of wizard. That means that the whole workflow is divided into several basic steps. A single screen is dedicated to each of the steps. Each screen contains user input, accompanied by detail information about the content of the screen.

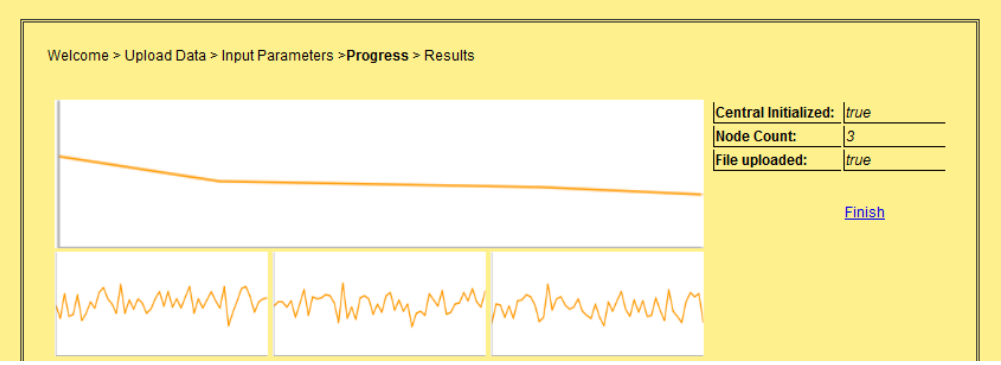

Figure 16 User interface sample

The user interface is a web application that is hosted in a JBoss server environment. JBoss is a free, java enterprise application container build around Apache Tomcat server. It

comes with a number of tools which make the application easy to develop and deploy. The JBoss server also allows RMI naming service to run within the application container. The Enterprise Java Beans (EJB3) technology is supported by the JBoss application server. That makes room for further improvements and modernizations of the application.

3.3. RESULTS

The new school schedule creation and optimization program proves to be much faster than its predecessors. The new application design allows running the optimization code on several machines simultaneously. That increases the speed of optimization by the factor *N*, where *N* is a number of processors available on the network (assuming that all processors have the same processing power).

Currently, only the Monte-Carlo local optimization algorithm is implemented in the system. This particular algorithm was chosen due to its simplicity.

The resulting schedule is presented to the user in an XML format. A single XML file is created for each student. The same is done for the teachers. A sample of a student schedule is presented below.

	A	B
1	Monday	
2	1	Lk
3	2	Mat
4	3	Ist
5	4	KK
6	5	T
7	6	A
8	Tuesday	
9	1	Mat
10	2	Lk
11	3	E
12	4	A
13	5	Ist

Figure 17 Student schedule sample

3.4. FUTURE DEVELOPMENT

The newly developed system is fresh and still needs a lot of work. The main points that need to be addressed in the future are:

- User interface;

- Optimization algorithms;
- Improvements on the distributed optimization;
- Output;
- Parallelizing application code.

The user interface has to be improved and modified. Additional interface languages should be added. The current user interface is primitive and covers only the basic application control.

The addition of the new optimization algorithms would possibly improve the quality of resulting schedule. The global optimization algorithms, such as SA, should most definitely be implemented and tested. Not only the algorithms have to be implemented, but the distributed version of each of them should be developed.

The output of the application is also on the primitive level. There are endless possibilities to expand. Such options as HTML interactive output using AJAX or even PDF reports would complement the application and increase the quality of a user experience.

The code parallelization would be a great improvement of the application. This would drastically increase the performance of the application on a multi-core system.

3.5. SUMMARY

The new application architecture provides the researchers with a steady platform to develop new ways of creating and optimizing the school schedules. The ability to run the schedule optimization on several machines simultaneously enables the user to seamlessly utilize most of the processing power available. The software can be installed as on a single multi-core system or on a local network.

User Interface is simple and easy to use. No advanced computer skills required in order to operate the software. This makes the scheduling software available to an average computer user. In that way, the scheduling system may be used as a tool to generate real school schedules in the Lithuanian high school.

Input and output uses an open XML format. The data can be viewed by MS Office Excel or a free OpenOffice Calc package. The majority of the users are already familiar with the system. That eliminates the need of additional training.

4. SUMMARY

The school schedule creation and optimization is a very complex and time consuming task. The complexity rises exponentially with the increase of the amount of initial data and the amount of constraints applied to the schedule. There are no known ways of finding the best schedule. This is the reason why heuristic algorithms are being employed in order to find the closest to the best solution.

Currently, the Lithuanian high schools are using commercial software packages in order to aid the school schedule creation. The majority of this software packages are not able to cope with the flexibility offered by the Lithuanian education system. In order to bring the effort of creating a school schedule to the minimum, specialized software is required, specially created for the Lithuanian high schools.

The analysis of the school schedule creation and optimization software packages created previously, uncovered major flaws in the schedule creation algorithms. The essential constraints were missing, thus making the creation of the valid school schedule impossible.

The new distributed scheduling software platform was created. The modern design and architecture allows the researchers to modify the application, add new constraints, create and test new optimization algorithms without overloading the application code. The ability to run the optimization of the school schedule on multiple machines simultaneously drastically decreases the time required to run the schedule optimization.

5. REFERENCES

Abramson, D. 1991. Constructing school timetables using simulated annealing: sequential and parallel algorithms, Management Science, vol. 37, no. 1, 98-113.

Action. ACTion Scheduling Software, [viewed on 2007-11-22]. Internet access: <http://www.timetable.co.za/>

Alaburdienė, R.; Dovidauskaitė, S.; Nekiūnienė, V.; Rekerta, G.; Alaburda, M. Pamokų tvarkaraščiai, [viewed on 2009-10-25]. Internet access: <http://www.soften.ktu.lt/~mockus/mimosa/index.htm>

Alexandrov, V.; Dimov, I.; Karaivanova, A.; K.Tan, C. J. 2003. Parallel Monte Carlo algorithms for information retrieval. Mathematics and Computers in Simulation, Vol 62, Issues 3–6.

Apynis, A., 2005. Optimizavimo metodai. Vilniaus universiteto leidykla, Vilnius, ISBN 9986-19-828-3.

aSc. aSc Timetables, [viewed on 2010-01-25]. Internet access: <http://help.asctimetables.com/index.php?lang_id=10>

Bilgin, B.; Ozcan, E.; Korkmaz, E. E. 2007. An Experimental Study on Hyper-Heuristics and Exam Scheduling, PATAT2006, Springer-Verlag, selected papers, LNCS 3867, 394–412.

Birbas, T.;Daskalaki, S.; Housos, E. 2009. School timetabling for quality student and teacher schedules, Journal of Scheduling, vol. 12, Issue 2, 177-197.

Burke, E.K.; Petrovic, S. 2002. Recent research directions in automated timetabling, European Journal of Operational Research 140, 266–280.

Cooper, T.B.; Kingston, J.H. 1996. The complexity of timetable construction problems. In: Proceedings of the 1st International Conference on Practice and Theory of Automated Timetabling (PATAT 1995), LNCS 1153, Springer-Verlag, 283-295.

Costa, D. 1994. A tabu search algorithm for computing an operational timetable, European Journal of Operational Research, vol. 76, 98-110.

Erben, W.; Keppler, J. 1996. A genetic algorithm solving a weekly coursetimetabling problem, in: E. Burke, P. Ross (Eds.), The Practice and Theory of Automated

Timetabling: Selected Papers from the First International Conference, Lecture Notes in Computer Science, Springer, Berlin, vol. 1153, 198–211.

Felinskas, G.; Sakalauskas, L. 2003. Pareto tipo modeliai modeliuojamojo atkaitinimo algoritmuose. Lietuvos Matematikos rinkinys, T.43, spec.nr., 573-578. (ISSN 0132-2818) (Duomenu bazse: CIS, MatSciNet, VINITI, Zentralblatt MATH).

Gaidukevičienė, R.; Kurilovas, E. 2005. Comparative study of profiled school scheduling programs in Lithuania, Informatics in education. ISSN 1648-5831. Vol. 4, Nr.1.

Glibovec, N.N.; Medvidj, S.A. 2003 Генетические алгоритмы и их исползование для решения задачи составления расписания, Кибернетика и системныйанализ, No.1.

Glover, F.; Laguna, M.; Marti, R. 2003. Scatter Search, Advances in Evolutionary Computation: Theory and Applications, A. Ghosh and S. Tsutsui (Eds.), Springer-Verlag, New York.

Hoos, H.H.; Stutzle, Th. 2004. Stochastic Local Search. Foundations and Applications. Morgan Kaufmann/Elsevier.

Laumanns, M.; Bayesian, J.O. 2004. Optimization Algorithms for Multi-objective Optimization, ISSN 0302-9743 (Print) 1611-3349 (Online)

Machado, J.M.; Shiyou, Y.; Ho, S.L.; Peihong, N. 2001. A common Tabu search algorithm for the global optimization of engineering problems, Computer methods in applied mechanins and engineering, Vol.190.

Mimosa. Mimosa Scheduling Software, [viewed on 2008-10-25]. Internet access: <http://www.mimosasoftware.com/>

Misevičius, A.; Bukšnaitis, V; Blonskis, J. 2006. Euristiniai algoritmai: tikslai, iššūkiai, metodologija, perspektyvos, Informacijos mokslai. ISSN 1392–0561.

Mockus, J. 2000. A Set of Examples of Global and Discrete Optimization: Application of Bayesian Heuristic Approach. Kluwer Academic Publishers. ISBN 0-7923-6359-0.

Mockus, J. A Set of Examples of Global and Discrete Optimization by Jonas Mockus, [viewed on 2009-01-22]. Internet access: <http://www.soften.ktu.lt/~mockus/>

Pupeikienė, L. 2006. Mokyklos profiliuotų klasių tvarkaraščių optimizavimo sistema. Informacinės technologijos 2006: konferencijos pranešimų medžiaga; Kauno technologijos universitetas. T.1. Kaunas: ISBN 9955-09-993-3.

Rector. Pamokų tvarkaraštis "Rector", [viewed on 2008-10-25]. Internet access: <http://www.rector.spb.ru/lt/index.html>

Sakalauskas, L.; Bartkutė, V.; Felinskas, G., 2006. Optimality testing in stochastic and heuristic algorithms, Technological and economic development of economy, t. XII, Nr.1, p. 4-10, ISSN 1392-8619.

Sakalauskas, L.; Felinskas, G. 2006. Tvarkaraščių su ribotais ištekliais sudarymo euristiniai algoritmai, jų tyrimas bei taikymai, Informacijos mokslai, 38 tomas, 90-103. ISSN 1392-1487.

Statistics. Statistikos departamentas, [viewed on 2009-04-21]. Internet access: <http://www.stat.gov.lt >

Žvirdauskas, D.; Adaškevičienė, V.; Tarnauskas, K.; Žvirdauskienė, R. [interatyvus]. 2006. Priemonių, skirtų vyresniųjų klasių mokinių mokymosi krūvio mažinimui, veiksmingumas. Lietuvos Respublikos švietimo ir mokslo ministerijos tyrimas, [viewed on 2009-10-15 d.]. Internet access: <http://www.smm.lt/svietimo_bukle/docs/tyrimai/Kruviu_ataskaita_0702.pdf >

6. TABLE OF TERMS

Heuristics – a technique designed to solve a problem that ignores whether the solution can be proven to be correct, but which usually produces a good solution or solves a simpler problem that contains or intersects with the solution of the more complex problem.

NP – NP is one of the most fundamental complexity classes. The abbreviation NP refers to "nondeterministic polynomial time"

Objective parameters are the parameters defined by the school. The rules defined by these parameters may be violated, but each violation will result in addition of penalty points.

Penalty points are the points assigned for each violation of objective constraints.

XML - (Extensible Markup Language) is a set of rules for encoding documents electronically. It is defined in the XML 1.0 Specification[3] produced by the W3C, and several other related specifications, all gratis open standards

Java – object orientated programming language.

Java Server Faces (JSF) – is a Java-based Web application framework intended to simplify development integration of web-based user interfaces.

Enterprise Java Beans (EJB) – is a managed, server-side component architecture for modular construction of enterprise applications.

SA - Simulated Annealing optimization method.

MC – Monte-Carlo optimization method.

GA – Genetic algorithm.

Appendix 1

Initial data file

Subjects‘ data sheet:

Subject	group	priority	MaxStud
Ps	e1	1	30
T	e1	1	30
E	e1	1	30
Lk	but	5	30
A	e2	3	20
V	e2	3	20
R	e2	3	20
Pr	e2	3	20
A	e3	3	20
V	e3	3	20
R	e3	3	20
Pr	e3	3	20
Ist	pas	4	30
Geo	pas	4	30
Mat	but	5	30
Inf	pas	4	30
Biol	pas	4	30
Fiz	pas	5	30
Chem	pas	5	30
Mu	pas	1	30
Sok	pas	1	30
Tea	pas	1	30
Da	pas	1	30
k.k.	but	3	30
lot	pas	2	30
isp	pas	2	30
gp	pas	2	30
br	pas	4	30
pb	pas	2	30
ek	pas	4	30
pc	pas	2	30

Teachers' data sheet

grup	Nr.	Subject	Name	Room	Hrs	Subj	Days off
1	1	Etika	Martinaitiene Laura	212; 213; 201;	22;	E	
	2	Etika	Patamsiene Jolanta	212; 213; 201;	16;	E	1;2;
	3	Tikyba	Maciuleviciene Natalija	213; 212; 201;	13;	T	5;
	4	Tikyba	Vasiliauskaite Nijole	213; 212; 201;	27;	T	
	5	Psicholigija	Lasauskiene Lilija	201;	1;	Ps	
2	6	Leit.k.	Bundoniene Orinta	202;	21;	Lk	
	7	Leit.k.	Kalveliene Laima	203;	21;	Lk	
	8	Leit.k.	Kucinskiene Anzelika	216; 219;	20;	Lk	
	9	Leit.k.	Miliseviciute Sigita	212; 208;	23;	Lk	5;
	10	Leit.k.	Paradauskiene Asta	201; 207;	22;	Lk	
	11	Leit.k.	Poskeviciute Dalia	212; 206;	20;	Lk	
	12	Leit.k.	Ramoskiene Aukse	213; 218;	25;	Lk	
3	13	Anglu	Kruopiene Ausra	204;	25;	A	1;
	14	Anglu	Bosulajeva Tatjana	205;	20;	A	
	15	Anglu	Gontiene Ausra	206;	24;	A	
	16	Anglu	Januskeviciute Irena	218;	25;	A	
	17	Anglu	Januskiene Birute	219;	24;	A	
	18	Vokieciu	Bieliauskiene Aldona	207; 219;	16;	V	4;5;
	19	Vokieciu	Kanapiene Zita	207; 202;	17;	V	1;5;
	20	Rusu	Mazintiene Birute	208;	27;	R	
	21	Rusu	Smitaite Ona	221;	14;	R	1;5;
	22	Rusu	Zuravliova Nina	221; 208	26;	R	
	23	Prancuzu	Bieliuniene Palma	209; 221;	16;	Pr	3;4;
	24	Prancuzu	Rudminiene Migle	213; 209;	18;	Pr	1;
4	25	Istorija	Abraitiene Danguole	210;	10;	Ist	
	26	Istorija	Jonykiene Odeta	211;	11;	Ist	
	27	Istorija	Guobiene Jurgita	221; 211;	23;	Ist	
	28	Istorija	Kavaliauskaite Regina	221; 210;	14;	Ist	1;5;
5	29	Geografija	Butkute Karina	109;	30;	Geo	
6	30	Matematika	Bukauskiene Rimante	102;	26;	Mat	
	31	Matematika	Jankauskiene Zivile	101;	22;	Mat	
	32	Matematika	Keniausiene Ada	115;	16;	Mat	4;5;
	33	Matematika	Medisauskiene Regina	113;	22;	Mat	
	34	Matematika; Informatika;	Osipaviciute Ruta	115; #214;	18;5;	Mat;Inf;	
	35	Matematika	Senulyte Sigita	113;	22;	Mat	
	36	Matematika; Informatika;	Veikutyte Ramune	101; #214;	21;5;	Mat;Inf;	
	37	Matematika; Informatika;	Velickiene Laima	101; #214;	4;5;	Mat;Inf;	1;
8	38	Biologija	Balseviciene Irma	106;	18;	Biol	
	39	Biologija	Kanciauskas Vidmantas	107;	24;	Biol	
	40	Biologija	Karoseviciene Ina	107;	9;	Biol	1;5;
	41	Geografija	Taraseviciene Rima	108;	30;	Geo	
9	42	Fizika	Adomenas Algimantas	103;	21;	Fiz	
	43	Fizika	Kavaliauskas Ovidijus	103;	24;	Fiz	
10	44	Chemija	Atstopaite Nomeda	104;	28;	Chem	
	45	Chemija	Vickackiene Lina	104;	24;	Chem	
11	46	Muzika	Blinstubiene Dalia	105;	20;	Mu	
	47	Muzika	Vaitkiene Vilija	105;	20;	Mu	5;
	48	Sokis	Aliaviene Daiva	110;	30;	Sok	
	49	Sokis	Diliene Vida	203;	20;	Sok	
	50	Teatras	Alkiene Vaida	222;	20;	Tea	4;5;
	51	Teatras	Juskeviciene Valda	222;	30;	Tea	
	52	Daile	Daliene Sigita	113;	30;	Da	
	53	Daile	Keriene Zita	115;	30;	Da	
	54	Daile	Stanionis Vytas	113; 209;	30;	Da	
12	55	Kuno kultura	Aliubavicius Rimgaudas	112;	20;	k.k.	
	56	Kuno kultura	Bundonis Arunas	112;	10;	k.k.	
	57	Kuno kultura	Gaina Giedrius	114;	26;	k.k.	
	58	Kuno kultura	Kasiulynaite Aldona	112; 114;	12;	k.k.	3;
13	59	Lotynu	Leka Nika	203;	8;	lot	1;2;
	60	Ispanu	Sinkeviciene Ausra	202;203;	10;	isp	
	61	Geopolitika	Martinkevicius Algis	203;	10;	gp	
	62	Braizyba	Ramoniene Salvinija	116; 115;	20;	br	1;
	63	PB	Dainauskas Dainius	203;	6;	pb	1;2;
	64	Ekonomika	Sinkiene Laura	203;	27;	ek	

Single group students' data sheet:

Eil.	Pavarde Vardas	Dor. ugd.			Leit.k.	1 užs.kalba				2 užs.kalba				Ist	Geo	Mat	Inf	Biol	Fiz	Chem	Menai				KK	Pasirenkami							Viso
		Ps	T	E		A	V	R	Pr	A	V	R	Pr								Mu	Sok	Tea	Da		lot	isp	gp	br	pb	ek	pc	
1	3a_Pavarde_1 Vardas_1		1		6	4					2			4	2	3	1	2	1	1		1			1				2				31
2	3a_Pavarde_2 Vardas_2		1		6	4					2			4	2	3	1	2	1	1		1			1						2		31
3	3a_Pavarde_3 Vardas_3		1		6	4						2		4	2	3	1	2	1	1		1			1						2		31
4	3a_Pavarde_4 Vardas_4		1		6	4							2	4	2	3	1	2	1	1		1			1				2				31
5	3a_Pavarde_5 Vardas_4		1		6	4						2		4	2	3	1	2	1	1		1			1				2				31
6	3a_Pavarde_6 Vardas_5		1		6	4							2	4	2	3	1	2	1	1		1			1						2		31
7	3a_Pavarde_7 Vardas_6			1	6	4					2			4	2	3	1	2	1	1		1			1						2		31
8	3a_Pavarde_8 Vardas_7			1	6	4						2		4	2	3	1	2	1	1		1			1						2		31
9	3a_Pavarde_9 Vardas_8		1		6	4							2	4	2	3	1	2	1	1		1			1				2				31
10	3a_Pavarde_10 Vardas_9			1	6	4					2			4	2	3	1	2	1	1		1			1						2		31
11	3a_Pavarde_11 Vardas_10			1	6	4					2			4	2	3	1	2	1	1		1			1						2		31
12	3a_Pavarde_12 Vardas_11			1	6	4						2		4	2	3	1	2	1	1		1			1						2		31
13	3a_Pavarde_13 Vardas_12		1		6	4							2	4	2	3	1	2	1	1		1			1						2		31
14	3a_Pavarde_14 Vardas_13		1		6	4						2		4	2	3	1	2	1	1		1			1						2		31
15	3a_Pavarde_15 Vardas_14			1	6	4					2			4	2	3	1	2	1	1		1			1				2				31
16	3a_Pavarde_16 Vardas_15			1	6	4							2	4	2	3	1	2	1	1		1			1						2		31
17	3a_Pavarde_17 Vardas_16			1	6	4						2		4	2	3	1	2	1	1		1			1				2				31
18	3a_Pavarde_18 Vardas_17			1	6	4					2			4	2	3	1	2	1	1		1			1				2				31
19	3a_Pavarde_19 Vardas_18		1		6	4							2	4	2	3	1	2	1	1		1			1						2		31
20	3a_Pavarde_20 Vardas_19		1		6	4						2		4	2	3	1	2	1	1		1			1						2		31
21	3a_Pavarde_21 Vardas_20			1	6	4					2			4	2	3	1	2	1	1		1			1				2				31
22	3a_Pavarde_22 Vardas_21		1		6	4					2			4	2	3	1	2	1	1		1			1				2				31
23	3a_Pavarde_23 Vardas_22			1	6	4							2	4	2	3	1	2	1	1		1			1						2		31
24	3a_Pavarde_24 Vardas_23		1		6	4							2	4	2	3	1	2	1	1		1			1						2		31
25	3a_Pavarde_25 Vardas_24			1	6	4					2			4	2	3	1	2	1	1		1			1				2				31