Artículo de investigación

Experience in study workload formation automation process

Опыт автоматизации процессов формирования учебной нагрузки

Recibido: 15 de septiembre del 2019 Aceptado: 20 de octubre del 2019

Written by:
Maxim A. Vikulin[81]
https://www.scopus.com/authid/detail.uri?authorId=57194508522
Leonid L. Khoroshko[82]
https://www.scopus.com/authid/detail.uri?authorId=14039206400
Elibrary.ru: https://elibrary.ru/author_profile.asp?id=467626

Abstract

The scheduling of student classes is one of the most difficult and important task in the organization of the educational process in universities. A large amount of work also increases the likelihood of errors, especially if all the work is done manually. This article deals with one of the preparatory stages for drawing up the schedule, namely, the formation of the study workload. This paper describes the experience of using automation algorithms for the process of forming the study workload, covering the full cycle of operations and comprising the following: obtaining initial data and their processing, aggregation of disciplines, estimation of hours, formation of a list of teachers, as well as displaying the resulting information in the form of a computer spreadsheet.
The use of these algorithms will reduce the labor costs associated with the estimation of the study workload, reduce the likelihood of mistakes associated with erratic human nature, as well as provide an opportunity for further automation and optimization of work related to the management of the educational process.

Keywords: Academic workload, automation algorithms, educational process organization.

Аннотация

Составление расписания занятий для студентов является одной из самых сложных и важных задач в организации учебного процесса в ВУЗах. Большой объем работы также увеличивает вероятность ошибок, особенно если всё выполняется вручную. В данной статье рассматривается один из подготовительных этапов составления расписания, а именно формирование учебной нагрузки. Описывается опыт использования алгоритмов автоматизации для процесса формирования учебной нагрузки, охватывающий полный цикл операций и включающий в себя получение исходных данных и их обработку, агрегирование дисциплин, оценку часов, формирование списка преподавателей, а также отображение полученной информации в виде компьютерной таблицы.

Использование таких алгоритмов позволит снизить трудозатраты, связанные с оценкой учебной нагрузки, вероятность ошибок, связанных с человеческим фактором, а также даст возможность осуществления дальнейшей автоматизации и оптимизации работы, связанной с управлением образовательным процессом.

Ключевые слова: алгоритмы автоматизации, организация учебного процесса, учебная нагрузка.

[81] Assistant, Moscow Aviation Institute (National Research University), 125993 Volokolamskoe highway 4, Moscow, Russia,
[82] PhD, Associate Professor, Moscow Aviation Institute (National Research University), 125993 Volokolamskoe highway 4, Moscow, Russia

Introduction

Workload formation is an integral part of the preparation for scheduling, as the generated workload presents the disciplines divided into subject classes, distributed between the students' groups or subgroups, and establishes a link between each lecture and a teacher. Thus, it can be said that a correctly composed load is the key to a properly composed schedule (Kabanov, 2013).

Another important application of the workload is the estimation of the number of hours, which particular teacher spends on training activities, i.e. the direct individual workload of the teacher. This number of hours is necessary to calculate the position pay rate, assigned to the teacher, and to determine the teacher's salary consequently.

Due to the urgency of the problem, it has been decided to automatize some of the tasks to be done, which would allow some schedule preparation steps to be skipped, and others to be simplified.

Interface and incoming data

The interface, which allows load formation, contains two key fields that influence the process and result of formation. The form fields contain information about the method and objects of load estimation. "Department" field contains a complete list of all subdivisions related to educational activities. The list is subdivided into two groups: "Administration" and "Departments" (Grotsev, Tomko, 2012). Estimation can only be made within the Department framework; thus, this estimation will not be done for every administrative unit separately but for all the Sub-Departments in general.

When forming "Department" field, the identifier of the current user and the list of employee's objects of this user are taken into account. Thus, the list contains only those subdivisions that are available to the user as per his duties in the university. This fact allows access to the service not only to the management staff departments that have access to all departments, but also to the employees of Departments and the Dean's back offices, for which the actual version of the workload will be available at any time.

The second key field in the program of the workload formation is the "Formation Timeframe" The workload is formed up for two semesters. So, the "Formation Timeframe" should embrace all academic year. At this particular stage the workload estimation has been developed for the current academic year and for the next ones.

Once the form has been submitted, the workload estimation function receives the identifier of the educational unit and the semester number from which the school year begins. In this case, the semester number begins with the current academic year figures. That is, the first value of the school year in the list will always designate figure one in the function, and the second and third points will pass the values of three and five, respectively.

Workload formation algorithm

After the function has been started, the transmitted values are processed. A department controller is deployed in order to process the academic unit identifier. The same controller forms up the unit identifier function as per the same unit academic profile identifier. The function of getting the unit identifier by its training identifier is called from it. This operation is necessary for the next function, which checks the department, identifier of which was transferred, for the presence of branch units. This function returns the data comprising the objects of all the subordinated units and the initial Department. If the subdivision, the identifier of which has been assigned, has no branches, the data with a single element (object of this subdivision) will be returned.

In order to continue with the job, it will be required to receive two patches of the database: one with the Departments' identifiers and another one with their academic identifiers. All the resulting objects of the Departments are processed in the cycling stage and the necessary data is registered with their respectful array.

Depending on the value of the academic semester, the value of the $Year variable, based on the Year commencement is formed. The text form of the mentioned value, designating the Year will be displayed. Upon saving the current data the value of the Month shall verified. September or the next month is taken for instance (meaning the academic year has already started) the value of Semester will be reduced by one point.

The variable $Semester is used to verify the academic groups under training within the month

of the workload formation. For correct definition of the list of groups it is necessary to reduce the given variable at the beginning of an academic year because of automatic increase in value of a semester of all groups.

The following step generates auxiliary array for further text substitution or data processing. First, the lists with the names of the blocks of disciplines related to the educational practice and the state final certification of students are created. It is necessary for further determination of whether any discipline belongs to the specified types of work. After that, the configuration results in a general list of all blocks of disciplines as a data patch, in which the elements begin with an index corresponding to the element one, and the zero element is an empty string.

Further on the information from the database is retrieved. The list of objects of subdivisions is divided into two arrays, the values of which are the short names of Departments. In the first patch, the values of ID Field are the keys, and in the second one, Education ID Fields are the keys.

Also, all formulas for calculating the workload are selected from the database and divided into two patches: one of them gets the default formulas and the other one gets the rest. Both patches are numbered from one to nine as an index, and each element contains formulas for the corresponding type of classes.

At the final stage of the given step the $Types array is formed from nine empty patches, which is a preparation for writing all the created data on the types of works.

To form a complete list of academic groups, several actions are required. First, it is necessary to get all the objects of academic groups that are currently active. Then to check if they will be trained within the period for which the workload is calculated. Also, it is necessary to add the groups to the list previously formed on the basis of key admission indicator (KAI) (Vikulin, Khoroshko, 2017).

In order to obtain the list of academic groups under the education layout, displayed in the workload plan it is significant to initially sort out all the academic curriculum plans embracing the disciplines delivered by the corresponding Departments and previously determined at the first step of the algorithm.

Then, it is necessary to get a list of academic group identifiers by linking the curriculum to the group in the object of a student.

Having a list of group identifiers, we can get all the necessary data: name of the academic group, current semester of study, and number of students enrolled in the group. Also, for correct work of the stage of formation of streams, the list of groups is sorted by the specialty code.

The received list of academic groups is processed by the cycle and for each group the semester of studying increases by value of a variable $Semester. Thus, the group is transferred to the semester of training, which will be essential within the period for which the workload is estimated. The comment rendered to a group contains full information about the periods of training, so if the resulting value of the semester does not appear in this field, the group will not be considered in the formation of the workload.

The next step is to create database structures similar to those of academic groups in order to store the information about the applicants, the number of which is determined by the KAI.

From the received data the objects of groups are formed, the name of which contains the letter code of the group, defined by the specialty, and the Department, which will form up a particular academic group. Each formed object is added to the general groups' database.

Due to the fact that the calculation must include students' data, which are not yet available in the database (except for the total quantity defined by KAI), availability of the total list of students is also divided into two stages.

First, the database of all academic groups currently active in university is processed. The identifiers of these groups are arranged into lists and the functions of receiving students from the database are transferred. This function returns the array of all students' objects, which are listed in the received list of academic groups, with curriculum identifiers and is transferred into the database of the newly admitted students' one. This function correlates the array of all students' objects, which are listed in the received list of academic groups, with the identifiers of the curricula, assigned for these students. Then, all the newly admitted students are sorted in a cycle and for each object the $Semester field is created, where the corresponding value of the semester for an academic group is input.

Then all objects of $Groups array containing the prefix "R" in the key are sorted. For each of such elements, students' objects are created in the quantity equal to the $Count field, and containing the group key, the curriculum plan identifier and the academic semester value. Each created student object is input to the general array $Students.

The next step selects the list of disciplines delivered by the list of the Departments, created at the initial step. Apart from the general information on a discipline, the information on the form of study, taken from the corresponding curriculum plan, is added.

All the disciplines are sorted in a cycle, where for each element the file of the students studying the given discipline is formed. Formation occurs on the basis of a particular student connection with the curriculum and the semester of training: in case the identifiers of a discipline, and a student academic plan coincide and the value figure of the semester corresponds to the current or the following one within the limits of the period of workload estimation such a student shall study the given discipline (Tolstykh E.S., Tolstykh A.A., 2014).

After formation of student lists, the array of disciplines is checked for empty elements. If any are found, they are deleted. After that, the database of disciplines is key-sorted, and two lists are formed on the basis of the list of identifiers of the considered disciplines: $SubjectsData and $SubjectsDataList.

Formation of consolidated list of disciplines with identification of academic groups

The consolidated list is formed up on the basis of $SubjectsDataList database, containing the values of the academic disciplines sorted by name. Each element in this cycle contains groups learning the subject of the discipline and forming up a file.

Spotting required academic groups is performed by a random search among all elements of the $Subjects database. The list of identifiers required for displaying information about groups is formed.

The required list is searched in a cycle where for each identifier of group the file containing the information on a discipline is created: academic block, discipline title, semester of studying, classes venue, quantity of credit units, time frames of lectures, laboratory and practical classes, hours of self-study works, information on a course paper or the course project, method of supervision, discipline Department title, course identifier within the system of electronic training and the curriculum code.

In order to interpret the identifiers and numeric codes into text information to be exposed, the substitution lists obtained at the second step are used. Besides, for such elements as coursework and attestation type, numerical codes are input into the database, obtained from the language file.

For the function of aggregation of disciplines to operate on the basis of the key parameters, we use the $SubjectsData database obtained at the fifth step of the algorithm, which contains a list of disciplines sorted as per the semester of study.

First, an empty $Aggregate array is created, which is used to store the unique key to run the aggregation process. Then, all the disciplines are processed in the cycle, each of which is assigned a key consisting of academic block, course identifier within the electronic curriculum system, quantity of credit units, semester of studying, corresponding Department, classes venue, information on a course paper or the course project, time frames of lectures, laboratory and practical classes the method of supervision and form of tuition.
A large number of fields makes it possible to identify the criteria by which aggregation should take place.

The key is constructed by a concatenation of values of all entitled fields with a certain separator. It is necessary for further processing of the key by means of disassembling it into separate elements. At the same time, the separator must be selected in such a way that there is no probability that it will occur inside any other data (for example, name of discipline). As a result, separator structure was chosen as a concatenation of two symbols: signs “less” and “more”.

After the key has been created, it is checked for its existence in the $Aggregate database. In case of a negative result, an element that is an empty array is created for this key. Then the discipline identifier is added to this element. Thus, after the end of this cycle an array is created, the keys to which are the aggregation criteria, and the elements are the lists of identifiers of the disciplines that fall under this criterion.

Then on the basis of the received result another file with the same keys is formed, but elements will be the lists of the students studying disciplines meeting the criteria of aggregation. For this purpose, for each criterion the search of all identifiers of disciplines is made, and the same elements of the $Subjects array will be formed up on the fifth algorithmic step.

The obtained data with $AggregateSubjects is supposed to be implemented within the eighth and the ninth step of the algorithm. In order to achieve the goal, it will be required to sort the elements, doing which every element of the aggregation will be processed in a certain order and properly prepared. Within the framework of the sorting the criterion shall be subdivided into the data patches by the predetermined separator for those patches further use. Following that the operation on the choice of the students, engaged in the study of a given subject, and meeting the pre-set criterion and forming up the list of the identifiers of the academic groups, where the given students study, will be done. After that the data patch of $StudyGroups, containing the objects taken from $Groups, the identifiers of which shall be included into the final list will be formed.

The next step presumes the sorting of the first four types of classes (classroom training): lectures, practical and lab classes, independent work control. The "subjects" is checked on matters of the hours' allocation first. Those classes' types without the hours' allocation shall be skipped, since such classes will not be delivered and therefore cannot be included into the workload.

The calculation formula is then defined, which for the designated classes represents the maximum number of students who can attend the class. First the default formula for the designated function is taken as the basic one. After that there comes a choice of additional formulas, deriving from the second step, where every element is additionally checked whether it is capable of re-defining the current one (Kukin, 2003).

Thus, after full selection of all formulas concerning the given kind of classes, the variable $Formula will have the value of the last element for which the described condition has been executed. Such actions will ensure the correct result because the most specified formula should be applied, and also because of a certain sorting of the formulas themselves, which has the records obtained from the database, in order of the fields concretization increase.

Then there comes the sorting of all groups included into the $StudyGroups array. And in case the title of a group is not represented or the number of students in it will encounter a zero the integration of such a cycle shall be skipped.

The number of students for every academic group shall be correlated with the assigned formula, which will further determine whether the whole set of groups or separate sub-groups will be formed.

The latter part of the operation shall only be done under a provision the number of students in a group exceeds the maximum number designated for such types of classes, set forth in the variable $Formula. In order to get the subgroups, it is required to determine their quantity: the number of students is divided by the formula value and rounded off upwards. Then there comes a cycle where every subgroup ordinal number, counting down from the general figure to the figure 1 inclusive, includes the number of students in this sub-group: the number of student unassigned to any of the sub-groups (general number of students in a group in the beginning) divide by the subgroup number and rounded-off upwards. Then, the variable $GroupCount (number of unassigned students) is subtracted by the obtained result. The subgroup result information obtained is input into the corresponding subject element of the $Types array, where the academic period of studies (autumn or spring accompanied by the year indicator), the academic department, the academic block, the discipline title, the classes venue, information on the subgroup in the form of the group name with indication of the subgroup name along with the number of students in it with the number of credit hours and a separate list of the quantity of students are depicted. The latter is required for estimation accuracy.

In case the number of students in an academic group is less than the number submitted in the formula there will occur an attempt to form up the groups' course consisting of several groups (Kalyuzhny, 2015). In order to obtain the result, there comes a cycle with a pre-condition to be followed until the overall number indicator of students is less or equal to the indicator under the formula requirements. There is a conclusion line to be formed up and input into the array and the quantity of students' figure to be added to the overall number in every integration for the current group. Then the check is done on whether the next element is present in the group list. In case it has been determined the number of students is added up by the overall number to be

compared with the variable $Formula. Under a provision the mentioned indicator does not accede the formula variable data then the external cycle counter, sorting all the groups engaged in the given subject study is increasing. But under an option when the mentioned indicator accedes the allowable limit the corresponding cycle is aborted with a pre-condition and the information on the groups summoned into the groups course is input into the $Types array. Thus, the external cycle counter increase will result in the skipping the groups forming up the courses groups during the further processing, which in its turn shall avoid the data duplication.

The next stage will form up the information on the remaining types of classes, the accountable hours of which represent the estimated ones. Depending on the type of the class there goes a check on matters of its necessity to be included into the workload. Under a condition if the key-slot for a certain type of works is empty either for practical purpose or for the benefit of the final certification the academic block being beyond the allowable list shall be skipped.

Then the necessary formula is searched for, identical to that used for classroom activities. In addition, a separate formula is defined for the state final certification, which should include only the academic block.

Then all the academic groups studying this discipline are sorted, where the initial formula is calculated for each group. Before the calculation, the following data is substituted in the formula: number of students, groups and credits. The number of students and credit units varies by group and discipline, respectively, while the number of groups always equals the figure 1, as the calculation is done without aggregation of objects (that is, separately for each group).

After the estimation has been done, the corresponding information is added to the $Types data patch. In case of lecture hours availability, a percentage of them (depending on the form of academic training) a separate load of hours shall be added for consultation purposes. For the final certification, in addition to the management duties, a load calculated by a separate certification formula is added, for which data input and calculation are also performed, and then the information is added to the section "FC".

When compiling the time table it is necessary to allocate the individual workload to every lecturer, because the resulting list of the tutors has to be filled in with the ones engaged within a certain Department (Grigorash, 2013).

In order to accomplish that all the active tutors are selected from the database with corresponding information about each, taken from the table of individuals (name, surname, patronymic).

Then, for every employee there is an element from $Types array formed up with an “emp” key containing the fields like: Department, Surname, Name and Patronymic of the employee, Job position, Pay rate, Academic rank and position.

Data upload into the template

At the last step, all the generated information contained in the $Types array should be transferred to the template and downloaded to the end-user’s computer. To do this, you should first specify the name of the template into which the data will be uploaded. Depending on the number of employees received at the tenth step of the algorithm, either the template load.xlsx or - load_big.xlsx will be chosen (if there are more than forty-nine employees).

To work with a template, the “OpenTBS” template engine is deployed and the template is uploaded. Within the framework of this template engine, all fields specified in the template must be defined, so all elements of the $types array are checked, during which, if an empty element is found, it will be filled with the data containing the necessary associative keys and empty strings as elements.

Then each of the elements of the $Types array is associated with the corresponding block of the template. In addition, because each block is on a separate sheet, it switches over between them. After all the blocks are linked to the corresponding data, the current date is input within the name of the template (it is also the date of formation), and the template engine is deployed, which downloads the file to the user's computer.

Conclusion

In aggregate we have got a software product that can be used to storage and modify the information accompanied by a possibility of forming up the lecturers’ workload and its allocation among them.

In a nutshell, at this stage there is already an estimation of the load itself and its upload to a

file, which can be used by the-end user. But for correct definition of quantity of the hours allocated under a certain type of works, it is necessary to consider the standards of working time allocation affirmed and laid out in the corresponding order also (Pyankova, Glotina, Nugolnykh, 2013).

References

Grigorash O.V. (2013). Methodology of planning the academic load taking into account the results of the work of teachers. Scientific Journal of KubSAU. 92, 1-12.

Grotsev A.R., Tomko V.N. (2012). Possibilities of using HTML5 when creating interface elements for training systems. Educational Technology and Society. 3, 571-582.

Kabanov V.N. (2013). Labour Rate Setting in Higher Education Institutions. Economics of Education. 2, 42-48.

Kalyuzhny N.V. (2015). Analysis of the process of distribution of the academic load of the higher-education teaching personnel at the chairs. Science Time. 6(18), 199-202.

Kukin A.V. (2003). Mathematical model of the distribution of academic workload in a University. MsiM. 2(12), 165-170.

Pyankova N.V., Glotina I.M., Nugolnykh K.V. (2013). Prospects for solving the problem of automation of distribution and accounting of the academic load at the chairs. Perm Agrarian Bulletin. 2(2), 53-55.

Tolstykh E.S., Tolstykh A.A. (2014). Automation of scheduling in the Educational Process Management System. Territory of Science. 1, 41-48.

Vikulin M.A., Khoroshko L.L. (2017). Development of the reception check digit storage service using the MAI e-learning system. XLIII International Conference for Young Researchers. Moscow: Moscow Aviation Institute (National Research University). Available at: https://gagarin.mai.ru/files/2017/Abstracts.pdf.