Implementing the Adaptive Learning Techniques

Article · January 2020

DOI: 10.17323/1814-9545-2020-2-252-277

2 authors, including:

Ivan Krechetov
Tomsk State University of Control Systems and Radioelectronics
5 PUBLICATIONS **3** CITATIONS

SEE PROFILE

Implementing the Adaptive Learning Techniques

I. Krechetov, V. Romanenko

Received in December 2019

Ivan Krechetov
Head of the Laboratory of Instrumental Modelling and Learning Systems, Tomsk State University of Control Systems and Radioelectronics. Email: kia@2i.tusur.ru
Vladimir Romanenko
Candidate of Sciences in Technology, Associate Professor, Department of Automated Control Systems, Tomsk State University of Control Systems and Radioelectronics. Email: rva@2i.tusur.ru

Address: Room 607, 146 Krasnoarmeyskaya St, 634034 Tomsk, Russian Federation.

Abstract. The concept of adaptive learning emerged a few decades ago, but most theoretical findings have never been put into practice, and software solutions had no significant reach for a long time due to insufficient e-learning technology development and coverage. The recent advancements of information technology allow the elaboration of complex big data analytics and artificial intelligence solutions, in adaptive learning in particular.

This article investigates exploitation of adaptive learning technology and techniques. The solutions proposed allow mapping optimal individualized learning paths for students in online courses, using the ratio of the level of knowledge at course completion to time spent on the course as an optimality criterion. A genetic algorithm is used to solve this optimization problem. A model based on the speed of forgetting was applied to extrapolate the level of retained knowledge.

Practical implementation of the technology proposed involves a set of tools to expand the adaptive learning opportunities of distance learning systems and a module to operate the genetic algorithm. We developed a few versions of software architecture using different technologies and programming languages and either one or two servers. The solution was tested during the design of adaptive learning courses for National University of Science and Technology MISIS (NUST MISIS) and Tomsk State University of Control Systems and Radioelectronics (TUSUR).

Keywords: adaptive learning, e-learning, genetic algorithm, distance learning system.

DOI: 10.17323/1814-9545-2020-2-252-277

Massive open online courses (MOOCs) have changed the approach to basic education, erasing the boundaries of time and space, and keep transforming the contexts in which they are used. Modern trends in e-learning do not seek to replace MOOCs with newer solutions; rather, they focus on improving learning efficiency and enhancing student engagement.

Translated from Russian by I. Zhuchkova.

Adaptive learning, one of such trends, suggests making allowance for previous learning experience and monitoring the current process

and quality of knowledge building. In adaptive learning, every student follows an individual learning path adapted to their objectives and ability to perceive and process information.

The most prominent works in adaptive learning have been written by Pyotr Brusilovsky [Brusilovsky 1996; 1997; 1998; 2001], Igor Norenkov [Norenkov, Uvarov 2005; Norenkov, Sokolov 2009; Norenkov, Sokolov, Uvarov 2009], Galina Rybina [2008a; 2008b; 2010; 2011; 2014], and Leonard Rastrigin [Rastrigin, Erenshteyn 1988]; modern research in the field as well as the existing software solutions draw heavily on those works. However, adaptive learning technology had no significant spread for a long time due to low efficiency of computing tools, the state of science in the field, issues in the implementation of intelligent and expert systems, and low development and adoption of e-learning technology.

Today's level of information technology development allows finding relatively cost-effective complex solutions in AI and big data analytics, which became highly trending with the wide spread of e-learning and created a new segment in the market of educational software. Adaptive learning services and systems are one example of such solutions.

A lot of digital products available today feature elements of adaptive learning. The level and methods of adaptation may differ, being largely contingent on the context of software application and the specific aspects of a particular learning process. A substantial proportion of adaptive learning solutions are developed for adult and corporate education, foreign language services in particular. Meanwhile, the technology is poorly represented in higher education, since using it largely implies an educational reform. Some solutions do exist, however, and the producers have been trying to promote their adoption in the digital environments of higher education institutions.

Adaptive learning has mostly been adopted by American and European universities in the form of adaptive learning platforms like *Knewton*[1] or *Cerego*[2]. Courseware offered by such platforms is used as adaptive learning tools to support traditional classrooms. Those solutions for higher education have not conquered the Russian market yet, as there has been little or no localization; besides, the e-learning infrastructure of Russian universities makes integration technologically impossible.

Solutions described in this article were developed by Tomsk State University of Control Systems and Radioelectronics (TUSUR) as part of a project seeking to enable Russian universities to integrate adaptive e-learning practices into their learning processes.

[1] https://www.knewton.com
[2] https://www.cerego.com

1. Adaptive Learning Algorithm

1.1. Objective

The central idea of adaptive learning is about building optimal individual learning paths as sequences of modules. Module is a logically complete minimal unit of learning material that explains one or more terms or concepts and is connected to other units [Krechetov, Kruchinin 2017:75–80]. A module may contain text, graphics, video, audio, and any other interactive forms of content presentation.

Building a modular path is a multi-criteria optimization problem. However, given the specific aspects of higher education programs, particularly the limited course time, the ratio of the level of knowledge at course completion to the time spent on the course may be considered an optimality criterion:

$$(1) \quad F(P, t_{const}) = \frac{TM}{R(t_{const})} \rightarrow min.$$

Where P stands for the learning path (the order of learning modules), TM is the total time to be spent on all modules, and R denotes the level of retained knowledge. Since course completion period t_{const} is constant, it may be omitted when writing down the objective function (1).

The problem (1) is solved using integer programming, as the order of learning modules is basically the order of their identifiers, represented by integer values. Moreover, the problem-solving landscape is highly discretized, because by far not every order of learning modules is acceptable, which will be shown below. For this reason, problem limitations can only be written down as relations on sets, in terms of discrete mathematics. Conventional optimization algorithms do not work well with such problems, so we opt for a genetic one.

1.2. Methodological Approaches

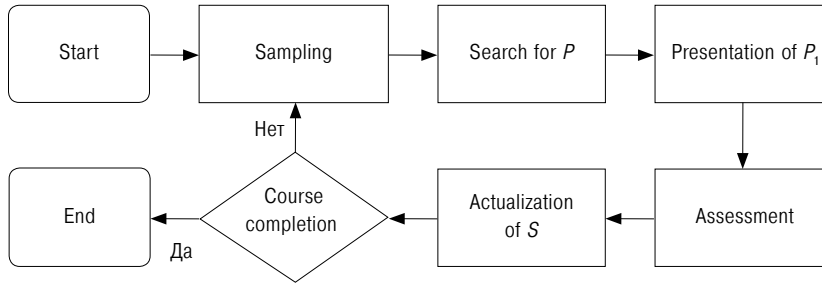
A model based on the speed of forgetting was applied to extrapolate the level of retained knowledge at course completion using intermediate test results.

There are few, if any, other reasonably substantiated models suitable for predicting the level of knowledge in future periods based on previous learning experience. Bayesian networks require substantial computing resources and do not offer better accuracy (beyond the adaptive testing objectives) [Khlopotov 2014:40–52]. Machine learning technology can accelerate decision making essentially (e. g. Snappet[3]), yet dozens of thousands completed learning paths would be required to ensure sufficient prediction accuracy for a course consisting of at least 150–200 modules. Therefore, statistical models (e. g. Bayesian network-based trust model) or forgetting curve models (e. g. [Kharitonov, Krushel 2012]) should be used at the initial stage of course integration.

Calculations for the mathematical model of forgetting used in this study, as well as the domain model, are given in Appendix.

[3] https://nl.snappet.org/

Figure 1. **Adaptive learning genetic algorithm flowchart**



1.3. Methodology and Algorithm

Figure 1 shows the general flowchart of an adaptive learning algorithm [Krechetov et al. 2018:33–40].

1.3.1. Adaptive Learning Algorithm

Let us now describe the sequence of steps.

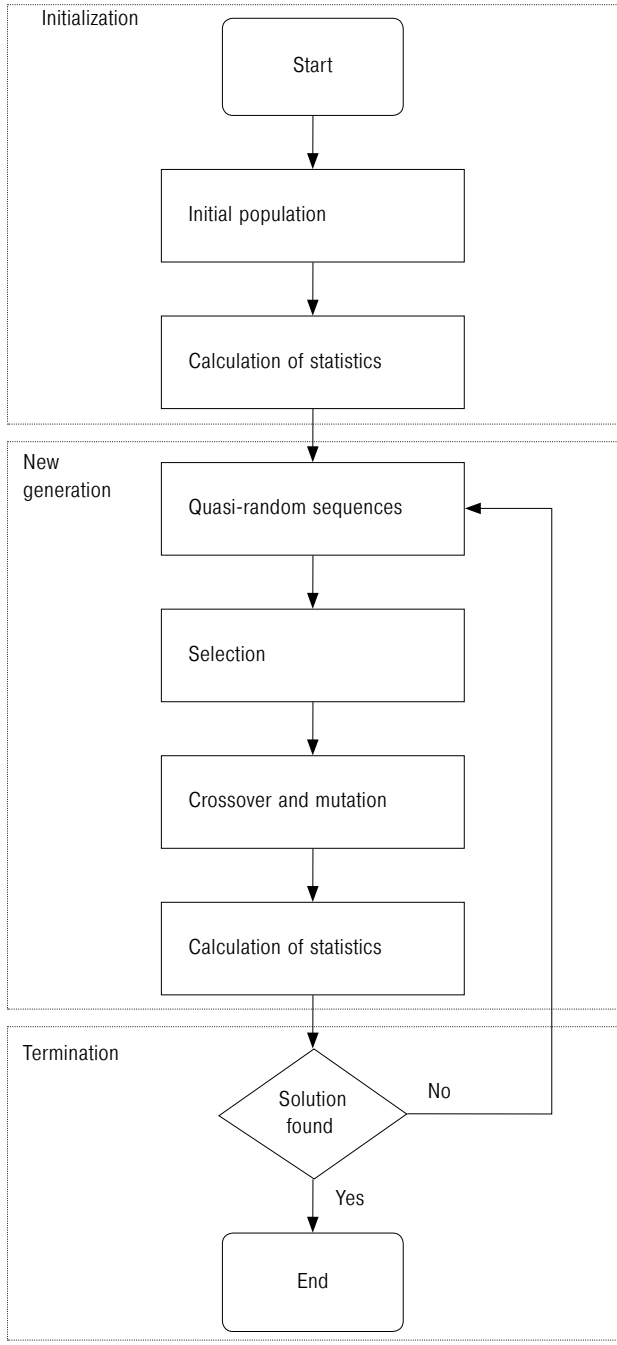
1. Sampling. Set A is sampled, which consists of modules for fostering poorly developed competencies (see formulae 11 and 12 in Appendix). Competency K_j is considered poorly developed either if it has never been learned, i. e. $HR_j = \varnothing$, or if information has been lost, i. e. the level of retained knowledge has gone below R_{norm} on the forgetting curve.
2. Search for P. A genetic algorithm described below is used for finding a learning path.
3. Presentation of P_1. A student is presented the first module of P. Learning modules are implemented in a distance learning environment.
4. Assessment. A test is developed within the framework of distance learning to assess the level of output module competencies.
5. Actualization of S. Test results are used to update the student's actual level of knowledge in history HR_j.
6. Check for course completion. A course is completed either if course time has expired, i. e. $t_{curr} \geqslant t_{const}$, or if all the competencies have been developed at a satisfactory level, i. e. $KS = K$, $KF = \varnothing$ (see formulae 9 and 10 in Appendix).

1.3.2. Genetic Algorithm Description

The fundamental principles of the genetic algorithm for generating a sequence of learning modules are provided in [Krechetov 2014:200–206]. The general flowchart is presented in Figure 2.

In a classical genetic algorithm, population is made up of individuals, each representing a candidate solution with a set of chromosomes determining its phenotype. Chromosomes usually represent arrays of bits, which are relatively easy to manipulate in the procedures of crossover, mutation, etc. When solving the problem of adap-

Figure 2. **Genetic algorithm flowchart**



tive content generation, the phenotype of an individual should define the learning path P, i.e. the order of learning modules from sample A. It is impossible to encode the phenotype as a bit array, because not every bit combination will produce an acceptable sequence of learning modules. For this reason, the implemented version of the genetic algorithm suggests that chromosomes describe ordered sequences of modules, which results in a much more complicated procedure.

1.3.3. Research and Intermediate Results

Below, we are dwelling on how genetic algorithm procedures work. Genetic algorithm parameters include the following:

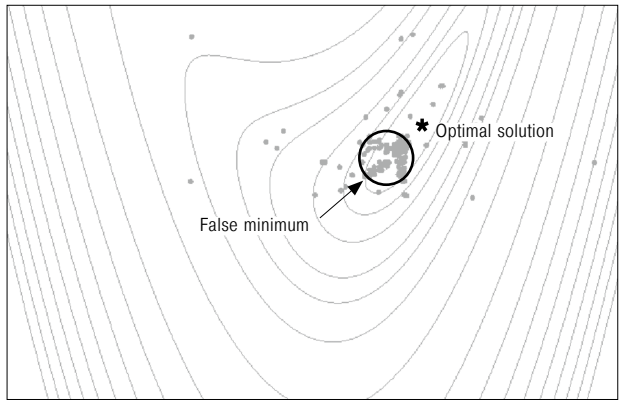
- Population size n_{pop} (set to 100);
- Maximum number of generations n_{gen} (set to 100);
- Mutation probability $p_{mutation}$ (1%);
- Crossover probability p_{cross} (90%);
- Elite individual emergence probability p_{elite} (5%).

1. Generation of the initial population. n_{pop} of individuals is generated uniformly.
2. Calculation of statistics involves two steps. First, fitness (target) function should be defined for each individual. Using the agreed notation, the previously introduced objective function (1) will look as

   $$F(P) = \frac{\sum_i TM_{Pi}}{\sum_j R_i} \to min,$$

   where $i = 1, 2, \ldots, np$ and $i = 1, 2, \ldots, m$. That is, by minimizing the objective function $F(P)$, we are trying to find a sequence of modules P with the shortest possible total course duration and at the same time to maximize the total level of retained knowledge in all competencies K_j at course completion. Second, such statistical indicators as minimum, maximum, and average objective function in the population (F_{min}, F_{max}, F_{avg}) are calculated. The individual with the lowest objective function value is selected as a near-optimal solution.
3. Quasi-random sequences. In the implemented version of genetic algorithm, crossover occurs between neighboring individuals, so quasi-random sequences are used to increase diversity in the initial population.
4. Selection. Next, neighboring individuals are compared in pairs, and the fittest one (the one with the lowest objective function value) is inherited unchanged by the next generation. Only half of the population can be generated using this method, so another iteration of quasi-random sequences and sampling is performed.
5. Crossover and mutation. The crossover procedure consists of two steps.

Figure 3. **False convergence of a genetic algorithm**



- Step 1. Identifying elite individuals. Figure 3 explains the necessity of adding them to the algorithm.

  Initially, genetic algorithms were studied to be applied to solve optimization problems, specifically the Rosenbrock function minimization problem. The Rosenbrock function is mapped as a narrow curved valley, so many optimization algorithms fail to converge to the global minimum. It turned out, eventually, that genetic algorithms often showed false convergence. A higher concentration of individuals in the valley would result in their genetic material overweighing that of the other individuals. That is, even fitter solutions (those closer to the global optimum) would become less fit as a result of crossover with "junk" DNA. A few generations later, nearly the whole population would degenerate into a false optimum.

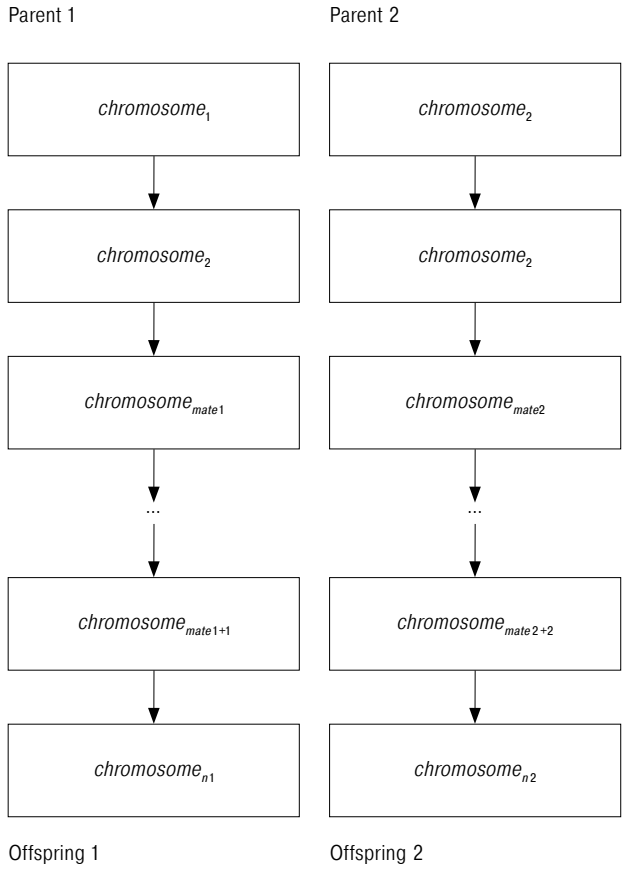
  To solve this problem, elite individuals were added to the population. Elite individuals constitute a small proportion of the population (p_{elite}) with the highest fitness function value (or lowest objective function value).

- Step 2. Randomly picking two individuals from the old population (previous generation) to generate two individuals of the new population. If at least one elite individual participates in crossover, its chromosomes are inherited by offspring unchanged, otherwise one-point crossover occurs with probability p_{cross}.

  To do this, a random crossover point $mate_1$ is chosen in the first parent.

  Next, a few attempts are made to find an acceptable crossover point $mate_2$ in the second parent (Fig. 4).

Figure 4. **Crossover process**

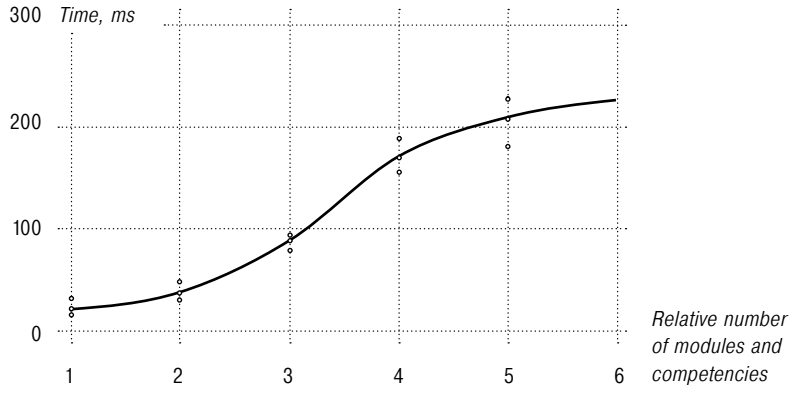


First, we pick a random point and check if the crossover points $mate_1$ and $mate_2$ are acceptable for the first and second parent chromosomes, respectively. Software generates "waves" of competencies for each of the parents, two waves of learned competencies spreading from the start of coding sequences (KS_1, KS_2) and two waves of nonlearned competencies spreading from the end of coding sequences (KF_1, KF_2):

$$KS_k = KS \cup \bigcup_{i=1}^{mate_k} KO_i,$$

$$KF_k = \left(KF \cup \bigcup_{i=mate_k+1}^{n_k} KI_i\right) - \left(KS \cup \bigcup_{i=mate_k+1}^{n_k} KO_i\right).$$

Figure 5. **Single iteration execution times**



Crossover is possible if $KS_1 \subseteq KF_2$ and $KS_2 \subseteq KF_1$.

If crossover points have been found, crossover occurs as described in the flowchart in Figure 4.

Experiments show the crossover success rate to be around 30%.

When implementing the mutation operator, either of the following two events occurs with probability $p_{mutation}$:

– Replacement of one module in a sequence with another one from the database, as long as the above conditions are met; or
– Transposition of two modules in a sequence under the same conditions.

6. Optimality assessment. The algorithm is terminated either if the generation count exceeds n_{gen}or no improvement is observed in F_{min} for ten generations.

A series of optimizations was performed, followed by an assessment of genetic algorithm execution time depending on the language used and the number of modules and competencies included in the adaptive course (Fig. 5).

The x-axis represents the relative number of modules and competencies in a section of the Informatics course (43 modules, 57 competencies). Other values were obtained by a fold increase in that number. For every value on the horizontal axis, three variables are plotted in the Y direction, representing the minimum and maximum execution time with the given number of modules and competencies as well as the median value obtained as the point with the lowest root mean square error:

$$t : \sum_{i=1}^{n} (t - t_1) \rightarrow min.$$

Some dispersion in the results is explained by using a random number generator in the algorithm procedure. The same input data may yield varying numbers of generations, crossovers, etc. Experiments show that the maximum number of generations that a genetic algorithm needs to find an optimal solution depends on the degree of diversity in the database of modules and competencies. With low diversity, an optimal solution can be achieved as soon as in the 5th generation, whereas higher levels of diversity require more iterations.

Therefore, algorithm execution time is determined by the total number of modules and competencies in a course and by module diversity. No correlation was found with the total number of modules and competencies in the database or the total number of students. Inefficiencies in server performance can only be observed if it receives multiple requests from different clients.

As seen in Figure 5, the growth rate of the function exceeds that of a linear function up to a certain point, but then it slows down because some phases in the genetic algorithm with relatively constant execution times have no noticeable impact on the total execution time anymore. This allows extrapolating efficiency of the genetic algorithm to courses with an arbitrary number of modules and competencies.

2. Implementation

2.1. Results, as Aligned with the Goals

The following project decisions were made while planning implementation of the adaptive learning model.

First of all, client-server architecture was opted for. Desktop software solutions are very rarely used in e-learning today, online systems being the primary focus of all universities investing in distance learning.

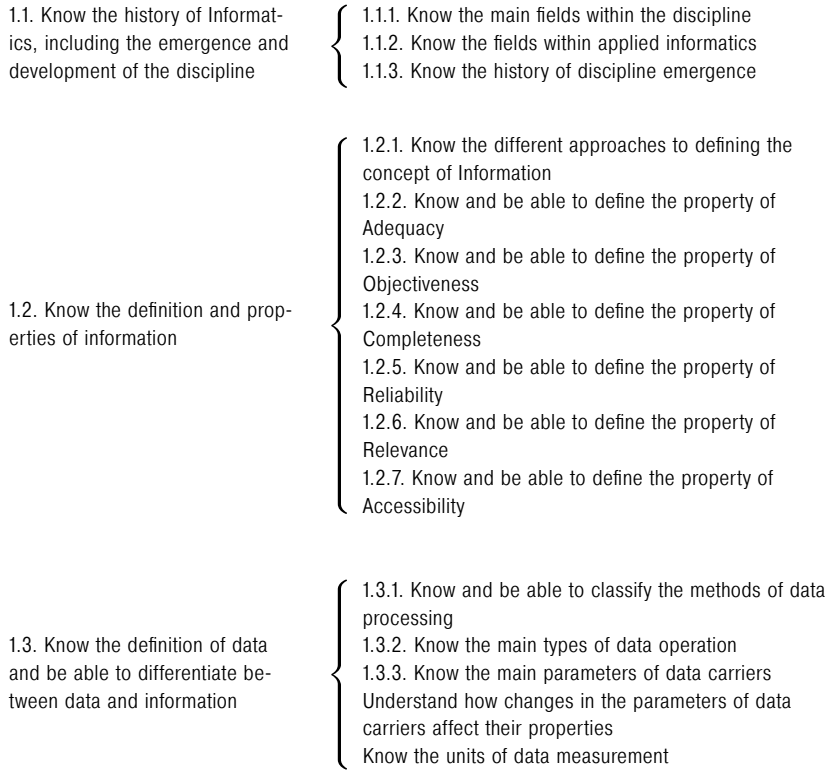
Second, it was decided to present all the tools for developing and launching adaptive learning courses as cloud services of the SaaS (Software as a Service) type to make them accessible to any university or institution concerned, regardless of the distance learning system they use, and to facilitate maintenance and support of the software component.

Third, modules and competencies were set to be universal, i. e. not assigned to any particular course. They are all contained in the same database and can be used as necessary in the development of a specific course.

Implementation involved the following steps.

1. A database of modules and competencies in the domain of informatics was developed. Originally, the modules were described in an ordinary Word document, and the competencies were represented as a mind map using FreeMind software (Fig. 6).
2. Software implementation of the previously described genetic algorithm was performed. Experiments show that scripting languag-

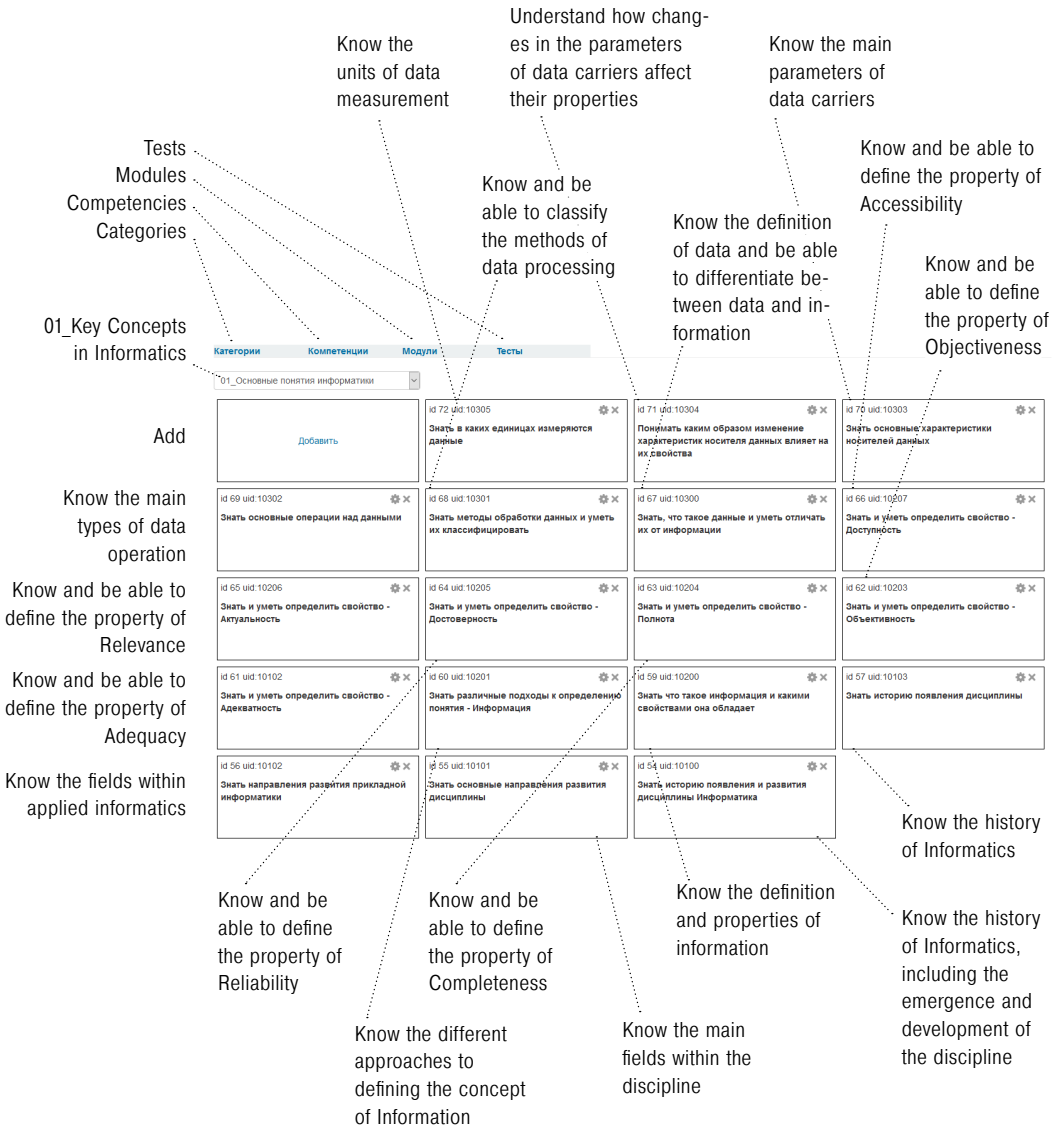
Figure 6. **A segment of the mind map of course competencies**



es like PHP are not suitable for solving this problem, so C++ and C# implementations are used in this study.

3. The database used in distance learning systems was modified to store all the data described in the domain model.
4. A few plugins were developed to create and deploy adaptive courses for the Moodle distance learning platform, (i) a local plugin containing the kernel of the solution, and an interface for database filling, (ii) a theme-type plugin to load the kernel and provide automatic navigation through the course for students, and (iii) a resource-type plugin to provide module content.
5. The pre-developed modules and competencies were transferred to the database using the new tools (Fig. 7).
6. The genetic algorithm was implemented as a SaaS solution on an individual server. The other development tools will also be available as cloud-native SaaS applications in the future.
7. The pilot course Informatics was created, and evaluation of the whole system was performed on its basis.

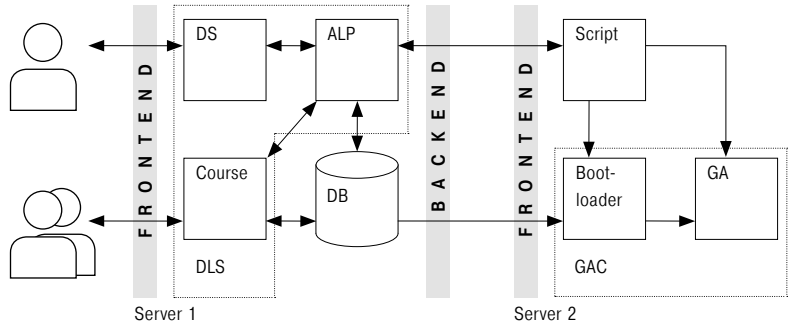
Figure 7. **Development system interface**



The overall architecture of the resulting software solution is displayed in Figure 8:

This architecture allows flexible changes to solution configuration. It was made as independent as possible from the distance learning system used. When changing the system, only the adaptive learning plugin and the genetic algorithm bootloader (in case the database structure has changed) need to be rewritten. As soon as all the tools

Figure 8. **Overall software solution architecture**



Legend:
DLS distance learning system used (Moodle), *Course* adaptive learning course; *DB* database (not included in Moodle, so displayed outside the DLS) containing all the information on course modules, tests, student profiles, etc., *ALP* plugins for *Moodle* running the adaptive learning model, *DS* development system, *GAC* component implementing the genetic algorithm, *Script* PHP script to exchange information between the servers, *Bootloader* component loading information necessary for genetic algorithm implementation (user profile, lists of modules and competencies, etc.) from the database; *GA* genetic algorithm implementation.

have been moved to the cloud, all the specified modifications will be performed on the cloud server.

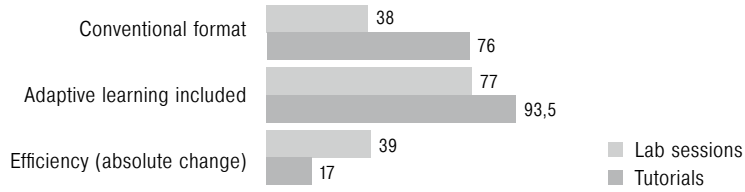
Implementation of the genetic algorithm as a cloud solution is also useful for load balancing: the algorithm occupies substantial computing resources, so using a cloud service allows keeping server response times low. However, there are no technical barriers to locating the system and all the necessary tools on the same server.

2.2. Evaluation of the Solution

In 2018, TUSUR and National University of Science and Technology MISIS (NUST MISIS) entered an agreement, under which an adaptive learning course in General Chemistry was created for NUST MISIS using the developed technology [Krechetov, Dorofeeva, Degtyarev 2018:76–86]. The course was tested in the fall semester 2018/19 by the NUST MISIS Department of General and Organic Chemistry. Unlike the online course in Informatics, this one used a blended learning model based on the flipped classroom strategy. Traditionally, the course General Chemistry featured three types of classroom activities: lectures, tutorials, and lab work. During course implementation, each of them was completed with an adaptive e-learning component.

Efficiency of an adaptive course can be evaluated using the chart in Figure 9. The y-axis represents the percentage of students who passed an assessment in their major on their first try. The sample

Figure 9. **Comparative performance of students in different groups**



consisted of groups of students of the same major in the same year, taught by the same professor.

As seen in Figure 9, students from the experimental group (which used adaptive learning) performed much better in every type of assessment than their peers in the control group with no adaptive learning component. For complete description of the evaluation results, see [Krechetov, Dorofeeva, Degtyarev 2018:76–86].

Because the integration of an adaptive course produced positive results, it was decided to continue cooperation between the universities. We are currently developing an adaptive course in Physics for NUST MISIS and an adaptive compensatory course in mathematics for NUST MISIS and TUSUR. A compensatory course in mathematics is necessary for first-year students who bring different levels of knowledge to the university and thus differ in their ability to study further mathematics. Compensatory content allows lower-performing students to catch up with the others.

We are also enhancing the data collection procedure of this adaptive learning model in order to use more student profile parameters when building individualized learning paths. Even more opportunities can be realized with big data on how students interact with content, which paths are the most effective for particular nominal categories of students, how rationally students spend the time allowed for learning, etc.

3. Conclusion

Technology proposed in this study is universal and suitable for application in a variety of contexts. It can be used to create an autonomous system of teacher-less learning. At the same time, the solution can serve as an effective student profiling tool in e-learning (distance learning), providing teachers with extensive student performance analytics and allowing them to adjust the individual learning paths. Furthermore, if e-learning tools are used to support classroom activities (most often with the help of distance learning systems), classroom performance can be tracked by the system to monitor recent student

progress, which can then be used by algorithms to actualize the distance learning paths.

References

Brusilovsky P. (2001) Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, vol. 11, no 1–2, pp. 87–110.

Brusilovsky P. (1997) Efficient Techniques for Adaptive Hypermedia. *Intelligent Hypertext: Advanced Techniques for the World Wide Web. Lecture Notes in Computer Science* (eds C. Nicholas, J. Mayfield), Berlin: Springer-Verlag, vol. 1326, pp. 12–30.

Brusilovsky P. (1996) Methods and Techniques of Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, vol. 6, no 2–3, pp. 87–129.

Brusilovsky P. (1998) Methods and Techniques of Adaptive Hypermedia. *Adaptive Hypermedia and Hypermedia*, Dordrecht: Kluwer Academic Publishers, pp. 1–43.

Buimov A., Buimov B. (2010) Veroyatnostnaya model effekta povtoreniy v obuchenii [Probabilistic Model of Recurrences]. *Proceedings of Tomsk State University of Control Systems and Radioelectronics*, no 1, pp. 236–242.

Ebbinghaus H. (1913) *Memory: A Contribution to Experimental Psychology*. Available at: https://archive.org/stream/memorycontributi00ebbiuoft (accessed 30 April 2020).

Kharitonov I., Krushel E. (2012) Prognozirovanie urovnya osvoeniya kompetentsiy vypusknikami vuza na osnove rejtingovykh otsenok abiturientov [The Prediction of the Gradu-ates Competency Mastering Level Based on the Current Raitng Estimates]. *Modern Problems of Science and Education*, no 6. Available at: http://www.scienceeducation.ru/ru/article/view?id=7531 (accessed 30 April 2020).

Khlopotov M. (2014) *Modeli i algoritmy intellektualnogo analiza obrazovatelnykh dannykh dlya podderzhki prinyatiya resheniy* [Models and Algorithms of Educational Data Intelligence Analysis to Support Decision-Making Processes] (PhD Thesis), St. Petersburg: ITMO University.

Krechetov I. (2014) Algoritm generatsii posledovatelnosti obrazovatelnykh moduley v tekhnologii polucheniya adaptivnogo obrazovatelnogo kontenta [Learning Module Sequence Generation Algorithms in Adaptive Learning Technology]. Gibridnye i sinergeticheskie intellektualnye sistemy [Hybrid and Synergetic Intelligent Systems]. Proceedings of the *2nd Pospelov International Symposium (Svetlogorsk, May 14–19, 2014)* (ed. A. Kolesnikov), pp. 200–206.

Krechetov I., Dorofeeva M., Dyagterev A. (2018) Raskryvaem potentsial adaptivnogo obucheniya: ot razrabotki do vnedreniya [Unlocking the Potential of Adaptive Learning: From Design to Implementation]. Proceedings of the *eLearning Stakeholders and Researchers Summit 2018*, Moscow: Higher School of Economics, pp. 76–85.

Krechetov I., Kruchinin V. (2017) Ob odnom algoritme adaptivnogo obucheniya na osnove krivoy zabyvaniya [About One Algorithm of Adaptive Learning Based on Forgetting Curve]. *Proceedings of Tomsk State University of Control Systems and Radioelectronics*, no 1, pp. 75–80.

Krechetov I., Romanenko V., Kruchinin V., Gorodovich A. (2018) Realizatsiya adaptivnogo obucheniya: metody i tekhnologii [Implementation of Adaptive Learning: Methods and Technologies]. *Open and Distance Education*, no 3, pp. 33–40.

Lange V. (1983) O skorosti zabyvaniya [On the Speed of Forgetting]. *Voprosy Psychologii*, no 4, pp. 142–145.

Norenkov I., Sokolov N.(2009) Sintez individualnykh marshrutov obucheniya v ontologicheskikh obuchayushchikh sistemakh [Creation of Individual Learning Routes in Ontology Education Systems]. *Information Technology*, no 3, pp. 74–77.

Norenkov I., Sokolov N., Uvarov M. (2009) Adaptivnye sredy sozdaniya obrazovatelnykh resursov [Adaptive Learning Environments]. *Nauka i obrazovanie / Science & Education*, no 3. Available at: http://technomag.bmstu.ru/doc/115688.html (accessed 30 April 2020).

Norenkov I., Uvarov M. (2005) Baza i generator obrazovatelnykh resursov [The Base and Generator of Learning Content]. *Informacionnye tehnologii / Information Technologies*, no 9, pp. 60–66.

Rastrigin L., Erenshteyn M. (1988) *Adaptivnoe obuchenie s modelyu obuchaemogo* [Adaptive Learning with the Learner Model], Riga: Zinatne.

Rybina G. (2008) Obuchayushchie integrirovannye ekspertnye sistemy: nekotorye itogi i perspektivy [Integrated Knowledge-Based Expert Systems: Some Conclusions and Outlooks]. *Iskusstvenny intellekt i prinyatie resheniy / Artificial Intelligence and Decision Making*, no 1, pp. 22–46.

Rybina G. (2008) *Teoriya i tekhnologiya postroeniya integrirovannykh ekspertnykh system* [Theory and Technology of Integrated Expert Systems Construction], Moscow: Nauchtekhlitizdat.

Rybina G. (2010) Sovremennye podkhody k realizatsii intellektualnogo kompyuternogo obucheniya na osnove razrabotki i ispolzovaniya obuchayushchikh integrirovannykh ekspertnykh sistem [Modern Approaches to Intelligent Computer-Aided Instruction Based on the Design and Implementation of Integrated Knowledge-Based Expert Systems]. *Pribory i sistemy. Upravlenie, kontrol, diagnostika / Instruments and Systems: Monitoring, Control, and Diagnostics*, no 5, pp. 10–15.

Rybina G. (2011) Intellektualnye obuchayushchie sistemy na osnove integrirovannykh ekspertnykh sistem: opyt razrabotki i ispolzovaniya [Integrated Knowledge-Based Expert Systems: Development and Implementation Practices]. *Informatsionno-izmeritelnye i upravlyayushchie sistemy / Information-Measuring and Control Systems*, no 10, pp. 4–16.

Rybina G. (2014) *Sistemy, osnovannye na znaniyakh. Integrirovannye ekspertnye sistemy* [Knowledge-Based Systems. Integrated Expert Systems], Moscow: Nauchtekhlitizdat.

Appendix

Forgetting speed. Experimental study of memory and forgetting was pioneered by German psychologist Hermann Ebbinghaus. The findings were published in his book *Memory: A Contribution to Experimental Psychology* in 1885 [Ebbinghaus 1913]. Ebbinghaus showed that the speed of forgetting could be satisfactorily approximated by the function

$$R(t) = \frac{k}{\lg t + c}, \quad (2)$$

where $k = 1.84$ and $c = 1.25$ if time t is expressed in minutes [Lange 1983:142–145]. These values were obtained for unassociated material.

Note. The level of knowledge cannot be below 0 or above 100%, so the condition $R(t) \in [0, 1]$ should be satisfied. With sufficiently high values of t, formula (2) yields positive values only, because

$$\lim_{t\to\infty}\frac{k}{\lg t + c} = 0.$$

However, low values of t render the formula ill-conditioned, so that it can produce an arbitrary output anywhere between $+\infty$ and $-\infty$. For this reason, we suggest using the following approach to solve formula (2) in practice:

$$R(t) = \begin{cases} 1, & t < 1 \\ \min\left(\dfrac{k}{\lg t + c}, 1\right), & t \geqslant 1. \end{cases} \quad (3)$$

Therefore, $R(t) = 1$ if $\lg t + c \leqslant k$, which can be interpreted as follows: newly-learned information is retained for some time in memory, higher coefficent k being associated with longer periods of retention.

With two experimental points on the forgetting curve, the coefficients k and c can be obtained:

$$\begin{cases} R_1 = \dfrac{k}{\lg t_1 + c}, \\ R_2 = \dfrac{k}{\lg t_2 + c}, \end{cases}$$

whence

$$c = \frac{R_2 \lg t_2 - R_1 \lg t_1}{R_1 - R_2}, \quad (4)$$

$$k = R_1(\lg t_1 + c) \text{ или } k = R_2(\lg t_2 + c). \quad (5)$$

Later studies conducted by Ebbinghaus to examine the process of forgetting meaningful material showed that 75% of information was retained after one day and 70% after four days. These findings are of more interest to us, as learning material is meaningful.

As there are 1,440 minutes in 24 hours, it follows from (4) and (5) that

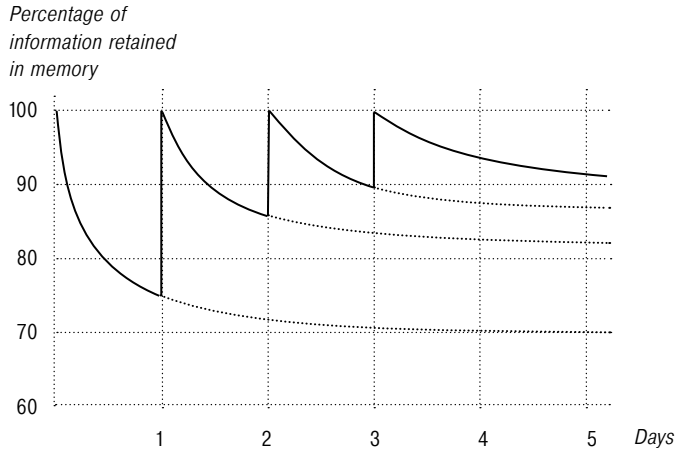
$$\frac{0{,}7 \cdot \lg 5760 - 0{,}75 \cdot \lg 1440}{0{,}75 - 0{,}7} \approx 5{,}270, \text{ and}$$

$$k = 0{,}75 \cdot (\lg 1440 + 5{,}270) \approx 6{,}322,$$

which can be taken as default coefficient values to be adjusted for the level of retained knowledge.

In this example, we examine a generalized learning process. Obviously, learning and forgetting are complex cognitive processes, and the tools available today are insufficient to predict the impact on their output precisely. However, one effect that can definitely be achieved

Figure 10. **Forgetting curve in iterative learning**



by iterative learning is that of bringing the level of student knowledge to a required level by the end of the course. In [Buymov 2010:236–242], the authors investigate the development of a repetition-based probabilistic model and derive a generalized forgetting function based on an arbitrary number of learning cycles. The function is mapped in Figure 10.

Therefore, repetitions increase the level of retained knowledge at course completion as a proportion of coefficient α:

(6) $R(t,r) = R(t) \cdot \alpha(r)$,

where r is the number of repetitions.

The type of dependence $\alpha(r)$ is unknown, plus it will be individual for every student. Suppose that it is of the form

$$\alpha(r) = \frac{1{,}5}{1 + \exp\left(-\frac{r}{1{,}5}\right)},$$

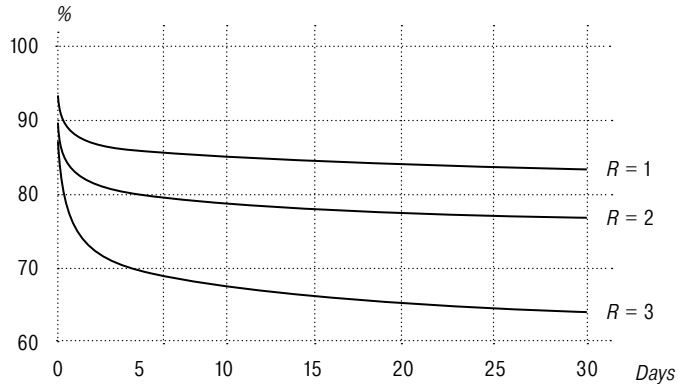
i.e. in the limit, with sufficiently many repetitions, $\alpha(r)$ will tend to 1.5 (as it has been mentioned, these assumptions will only be used to estimate the coefficients of dependence $R(t,r)$ by default; further on, they will be adjusted based on student's actual learning outcomes).

Let us define the parameters of dependence $R(t,r)$. Suppose that the level of retained knowledge is 100% after the minimum period of $t = 1$ min; then, it follows from (2) and (6) that

(7) $$R(1,r) = R(1) \cdot 1 = \frac{k(r)}{\lg 1 + c(r)} = 1 \Longrightarrow k(r) = c(r).$$

At course completion,

Figure 11. **Forgetting curve behavior depending on the *r* value**



$$R(t_{const}, r) = \frac{k(r)}{\lg t_{const} + c(r)} = R(t_{const}) \cdot \alpha(r).$$

With regard to (7), we get

$$k(r) = c(r) = \frac{R(t_{const}) \cdot \alpha(r)}{1 - R(t_{const}) \cdot \alpha(r)} \cdot \lg t_{const}.$$

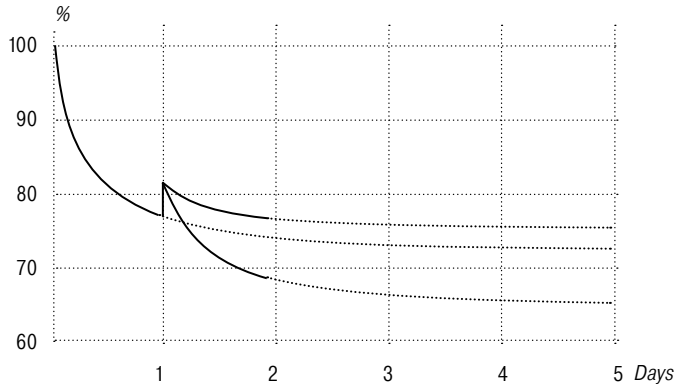
Therefore, we eventually get

$$(8) \quad R(t,r) = \begin{cases} R(t),\ r = 1, \\ \dfrac{k(r)}{\lg 1 + c(r)},\ r > 1. \end{cases}$$

When solving this formula, approach (3) should also be applied. Examples of the forgetting curve as a function of the number of repetitions *r* are displayed in Figure 11.

This formula, however, was derived from the assumption that every repetition restores the level of retained knowledge to 100%. It is not so in practice, so subsequent tests may show lower outcomes. For instance, Figure 12 displays a situation where student performance in the output test is 100% after the first iteration and 80% after the second one. In the latter point, the decrease in the level of retained knowledge $R(t, 1)$ has already slowed down, so the curve is less decreasing (the middle curve), but the new curve $R(t, 2)$ starts decreasing rapidly, so the repetition eventually results in quicker forgetting (the lower curve). For this reason, the software does not apply formula (8) to the whole amount of knowledge learned, but only to new material. For the situation in Figure 12, the predicted level of retained knowledge was 75% at the moment of repetition. The actual level turned out to be 80%, so the curve now has the formula

Figure 12. **Estimating the level of knowledge retained after repetition**



$$R = 100\% \cdot (R(t - t_1, 1) + (80\% - 75\%) \cdot (R(t - t_2, 2),$$

where t_1 and t_2 stand for the time elapsed before the first and second repetitions, respectively (upper curve). The real-life formula is even more complicated, as it has to consider all the iterations, all the individual coefficient adjustments based on students' detailed module learning history, etc.

Domain model. Domain model theory is described in [Krechetov, Kruchinin 2017:75–80]. During implementation, the model was subject to some changes [Krechetov et al. 2018:33–40]. Let the following notation be introduced to describe the adaptive learning algorithm further.

1. The set of course competencies $K = K_j$, $j = 1, 2, \ldots, m$, where m is the number of competencies (the notion of subcompetencies was introduced in [Krechetov, Kruchinin 2017:75–80], but let them be referred to as competencies in this study for brevity), and the set of important competencies (obligatory to learn) $IK \subset K$.
2. The set of course learning modules $M = M_i$, $i = 1, 2, \ldots, n$, where n is the number of modules. A module is described with four variables $M_i = (TM_i, KI_i, KO_i, RO_i)$, $i = 1, 2, \ldots, n$, where TM_i is time taken to complete the module, KI_i and KO_i are the lists of input ($KI_i = \{comp_k\}$, $comp_k \in [1, m]$, $k = 1, 2, \ldots, ni$) and output ($KO_i = \{comp_k\}, comp_k \in [1, m], k = 1, 2, \ldots, no$) competencies, and RO_i is the level of retained knowledge achieved for each output competency.
3. The set of groups of tests $T = \{T_j\} = \{\{T_{jk}\}\}$, $j = 1, 2, \ldots, m$, $k = 1, 2, \ldots, nt_j$ designed to evaluate the level of knowledge in com-

petencies K. Group T_j evaluates knowledge in competency K_j and may include one or more tests T_{jk}.

4. Course start time t_{start}, course end time t_{const}, and current point of time t_{curr}.
5. 5. Student profile $S = (HM, HT, HR, RK)$, where HM is individual module learning history, HT is individual testing history, HR is the history of changes in the level of knowledge in every competency, and RK stands for the forgetting curve coefficient in a particular competency as a function of the number of repetitions.
6. The satisfactory level of retained knowledge R_{norm}.
7. 7. The sets of well-developed (KS) and poorly developed (KF) competencies. If the level RA_{jl} of knowledge retained in competency K_j is below R_{norm}, such competency is considered poorly developed. That is, in terms of predicate calculus,

(9) $$KS = \{comp_k \mid RA_{comp_k l} \geqslant R_{norm}\},\ comp_k \in [1, m],\ k = 1, 2, \ldots, ns,$$

(10) $$KF = \{comp_k \mid RA_{comp_k l} < R_{norm}\},\ comp_k \in [1, m],\ k = 1, 2, \ldots, nf.$$

8. The set of modules $A = \{mod_k\}$, $mod_k \in [1, n]$, $k = 1, 2, \ldots, na$ that foster poorly developed competencies, i. e.

(11) $$\bigcup_{i \in A} KO_i \in KF,$$

while

(11) $$KO_{mod_k} \not\subset KS,\ k = 1, 2, \ldots, na.$$

The latter condition means that the set A excludes the modules of which all the output competencies have already been developed well enough.

9. Path P, describing an uncontroversial order of learning modules (not necessarily all of them) from the set A, $P = \{mod_k\}$, $mod_k \in [1, n]$, $k = 1, 2, \ldots, np$. Learning the modules from the set P should result in an improved level of knowledge in all the competencies for which $RA_{jl} < R_{norm}$. The uncontroversy requirement means that all the required input competencies should be developed by the time any module from P is started, i. e.

$$KS \cup \bigcup_{i < k} KO_{mod_i} \subseteq KI_{mod_k},\ k = 1, 2, \ldots, np.$$