Student's Multiple Abilities and Skills Model for Online Judge Systems

1st Fabiana Zaffalon
PPG Sciences Education
Federal University of Rio Grande
Rio Grande, Brazil
fabinhazaffalon@gmail.com

2nd André Prisco
PPG Sciences Education
Federal University of Rio Grande
Rio Grande, Brazil

3rd Ricardo de Souza
PPG Computational Modeling
Federal University of Rio Grande
Rio Grande, Brazil

4th Wanderson Paes
Center of Computational Sciences
Federal University of Rio Grande
Rio Grande, Brazil

5th Paulo Evald
Center of Computational Sciences
Federal University of Rio Grande
Rio Grande, Brazil

6th Neilor Tonin
Integrated Regional University
Erechim, Brazil

7th Sam Devincenzi
Center of Computational Sciences
Federal University of Rio Grande
Rio Grande, Brazil

8th Silvia Botelho
PPG Sciences Education
Federal University of Rio Grande
Rio Grande, Brazil

***Abstract*—This article presents a multi-skills estimation model for students using Online Judge systems. It is understood that there is not only one way to solve programming problems; and, for each solution form, a skill set is needed for the solution to be successful. The proposed model is based on performance expectations and integrates the Elo model, to estimate student's abilities and problems, to the Multidimensional Item Response Theory model, which estimates the probability of success for each solution *path*. To validate the proposed model, a case study was carried out with students from the computing area, who solved problems on the *beecrowd* Online Judge platform. The proposed model was applied to the generated database. According to these results, it is observed that, in cases where the students got the solution right, more than 60% of the *paths* chosen by students are in accordance with *paths* indicated by the proposed model.**

***Index Terms*—Model, Programming, Student's abilities, Student's skills.**

I. Introduction

Nowadays, there is a growing number of Online Judge systems, which provide exercises or problems for students. These systems help students learn; because, in addition to providing an exercise database, they also provide automatic feedback to the responses submitted by students, without the need for human intervention [1]. These environments allow the recording of many submission aspects, which allows educational assessment models to be used to infer the skills worked in each solution [2]. However, as many of these systems do not allow analyzing the interaction between subject and object, the estimation of the multiple skills involved in solving problems is challenging.

This study was funded by the Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES) – Finance Code 001.

To solve programming problems, it is necessary for the student to present more than one skill. Furthermore, it is possible for students to solve the same problem in different ways, according to their abilities, because programming problems can be solved in different ways (paths). It is highlighted that each problem has more than one way (path) to be solved and each path is composed of a set of skills needed for the solution. Besides, programming skills related to the programming language, students also need to have specific skills for problem solution, such as: logical reasoning, text interpretation, mathematical knowledge, specific knowledge of the problem area, among others [3].

There are massive online systems that make programming problems available to their users [4]. These platforms are known as *Online Judge*, because when receiving the source-code with problem solution, they evaluate it automatically, based on input and generated data, providing immediate feedback to the platform user [5]. In the context of massive online education, most Online Judge systems have access only to the final result of the interaction between a subject and an object. Thereby, it is not possible to identify the paths adopted and the skills involved in problem solution. In this way, it is important to adopt techniques for representing and evaluating the skills of both students and problems, guided by performance expectation-based methods.

The principle of performance expectation-based methods is updating the student's skills and problem difficulty after each subject-object interaction. In general, the models estimate the expectation of a student getting a certain problem right. When this expectation is met, there is a low increase in student's skills, and when there is a break in expectation, skills are

updated with a higher value [6].

The Item Response Theory (IRT) and Elo model are expectation-based models. The IRT aims to represent the probability of a person correctly answering an item based on its parameters (difficulty, discrimination and chance) and the person's abilities. The relation between the probability of a student giving a certain answer to an item and his latent traits (proficiencies or skills in the assessed area of knowledge), is expressed in such a way that the greater the individual's ability, the greater the probability of getting the correct answer [7]. Furthermore, IRT has models aimed at multiple skill items, called Multidimensional Item Response Theory (MIRT). Multidimensional models are classified as Compensatory, when a low skill can be compensated for by a higher skill; or Non-Compensatory, when all skills are needed [8], [9].

Elo is a statistical ranking system that can calculate relative skill level values for competitors or machines in competitive games. Its objective is to classify players through their game history. For the education area, the model was adapted to relate the student to the problem with which the student is interacting, while the original model deals with the relation between two players [10].

Aiming to contribute to Online Judge system improvement, a new model is proposed to estimate the multiple student abilities who solve programming problems in massive online learning environments. This model can be used in the Online Judge systems to allow it to provide better feedback to the students, facilitating its skill development. The model is based on performance expectations and integrates MIRT into the Elo model. In general, the proposed model takes into account the multiple paths for problem solution, identifying the most likely path to be taken by each student, through the expectation of success of each path and student's abilities. With this information, the model estimates, individually, the multiple student skills involved in the solution.

This work is organized as follows: Section II gives a general overview of the abilities required for programming problem solutions. Next, Section III and IV present the fundamentals of Online Judges and performance expectation-based methods, respectively. The proposed method is introduced in Section V, followed by a case study in Section VI. Finally, Section VII draws the conclusions.

II. Abilities of Programming Students

To solve a programming problem, it is necessary to interpret the information contained in the problem statement and have specific knowledge to plan a solution strategy. It is noteworthy that the same problem can be solved in several ways that require one or more strategies. Strategies involve skills such as reasoning and deductive logical thinking, in addition to allowing the student to exercise reasoning gradually [11]. There are many skills involved in solving a programming problem, with mastery of programming language syntax being just one of the skills [12]. Among them, the following skills stand out: mathematical ability, logical reasoning, text interpretation, analogical reasoning, conditional reasoning, procedural reasoning, temporal reasoning, among others [3], [11].

Advances in computing allow researchers from all areas to envision new strategies for solving problems. Computing has made it possible to solve advanced problems with practical applications. These advances drive the incorporation of computational support to problem solutions [13], such as Computational Thinking (CT) [14]. CT is a problem-solving method that decomposes a complex problem into minor problems to facilitate the abstraction and identification processes, as it is in computer science [14], which can help in solving problems in the most diverse areas.

Computational Thinking elements, related to Computer Science, can be identified in algorithm source-codes [13], [15]. They are:

- Data analysis: solution with basic statistical calculations [16]. This element is easily recognizable through the use of arithmetic operators in the algorithm [15].
- Data representation: solution using data structures [16]. This element is identified through the application of identifiers, arrays, pointers or data structures in the algorithm [15].
- Abstraction: solution using a set of commands that are repeated for the same purpose: use of conditionals, loops, recursion, etc. [16]. This element is recognizable by the presence of recursive calls, conditional structures and loop commands [15].

III. Online Judge

Online Judge are platforms that provide programming exercises for computer students, as well as programming competitors. These platforms are able to present an automatic evaluation of programming language source-codes. Its objective is to provide a fast and reliable evaluation, based on the algorithms submitted as solutions to the database. Basically, these systems have a standard for data entry. After data processing, the results are formatted into a standardized output. Thus, it is possible that automatic evaluation occurs through a tool that generates the input data and another tool that obtains, verifies and compares with the output data [17].

In Brazil, a widely used *Online Judge* system is *beecrowd*[1], which provides programming problems, where users can choose both the problems to be solved and the programming language used to solve the chosen problem [5], [18].

Beecrowd platform problems are organized by following categories:

- *Beginner*: intended for those who are starting to program;
- *Ad-Hoc*: problems involving simulation, dates, among others;
- *Strings*: problems that handle strings;
- *Structures and Libraries*: problems with stacks, queues, maps, sorting, among others;

[1]https://www.beecrowd.com.br/judge

- *Mathematics*: involves numerical systems, prime numbers, among others;
- *Paradigms*: problems covering dynamic programming, binary search, among others;
- *Graphs*: problems involving knowledge of tree implementation, maximum flow, among others;
- *Computational Geometry*: problems involving polygons, lines, among others;
- *SQL*: issues covering query languages.

After submitting the problem solution, if there are no errors during the compilation or execution of the submitted source-code, the *beecrowd* platform performs the automatic evaluation. It has a template for each problem that is compared with the output data of the user program, informing if the submitted solution is completely correct or if there is some error percentage. The user receives a *feedback* and can rewrite the code to submit it again. This process can be repeated until the source-code is completely correct.

beecrowd has 7 feedbacks, according to the solution [19]:

- *Accepted*: the problem was accepted without any errors;
- *Closed*: there was a problem connecting to the server; therefore, the submission was not received;
- *Compilation error*: some code structure was written wrong;
- *Presentation error*: the output data is not formatted as described in the problem statement;
- *Runtime error*: error in program execution flow;
- *Time limit exceeded*: execution time exceeded that stipulated in the problem statement;
- *Wrong answer*: program executed normally; however, it presented incorrect outputs.

IV. Performance expectation-based models

In education, expectation-based models calculate the chances of the student having a positive interaction with the learning object. In the case of problem solution, the models estimate the student's expectation to correctly develop the problem solution [20].

The principle of these models is to update the skills according to expectations and the obtained results. When a student with high ability solves a problem considered easy, it is understood that there is no break in expectation. In this case, skill upgrades will be lower than when expectation breaks occur [6]. Among these models, the Item Response Theory, IRT [8], and the Elo Rating System, or simply Elo [10], stand out.

A. Item Response Theory

IRT is a methodological framework for modeling response data from assessments in education, composed by a set of mathematical models that estimate the probability of a student successfully settling a question. The models estimate the abilities, θ, of the students through the answers given to the set of items in a certain area of knowledge to be evaluated. Skill estimation is related to the student's probability of getting the problem right, considering one or more parameters. Thus, the greater the student's ability, the greater the chance of a successful response [9].

The IRT has more than one model to estimate some skill. The one-dimensional models are applied in tests where there is a predominance of only one skill. On the other hand, in tests involving more than one skill, multidimensional models, MIRT, are more adequate [9]. Besides, IRT Multidimensional models can be compensatory and non-compensatory. In a compensatory model, shown in (1), a low skill can be compensated for by the lowest skill [21], it being possible that a student gets a solution to the problem.

$$P_{ij} = P(Y_{ij} = 1) = \frac{e^{(\sum_{m=1}^{M} \alpha_{jm}(\theta_{im} - \beta_j))}}{1 + e^{(\sum_{m=1}^{M} \alpha_{jm}(\theta_{im} - \beta_j))'}}, \quad (1)$$

where Y_{ij} is the individual response i to the item j; $P(Y_{ij} = 1)$ is the probability of correct answer; α_{jm} is the item discrimination parameter j in the dimension m; θ_{im} is the ability of the individual i in the dimension m; and, β_j is a scalar that indicates the item's difficulty j.

The non-compensatory MIRT model, shown in (2), assumes that the student must have each of the relevant skills to correctly answer an item. Therefore, when there is a low skill, the success probability is decreased. Each skill is treated as an independent probability [22].

$$P_{ij} = P(Y_{ij} = 1) = \prod_{m=1}^{M} \frac{e^{(\alpha_{jm}(\theta_{im} - \beta_j))}}{1 + e^{(\alpha_{jm}(\theta_{im} - \beta_j))'}}. \quad (2)$$

Research indicates that the MIRT model is better suited to reality, since, in education, issues require more than one skill from students [23]. Naturally, one-dimensional models do not apply in many practical situations. As an example, consider a mathematical test, where text interpretation is required before requiring mathematical development. In this case, it is a two-dimensional test, as it requires two skills [21].

B. Elo Rating System

The Elo classification is a method for calculating the relative skill levels of players, initially utilized in the chess game [10]. In the adaption of the Elo system to education, it is considered that a student is a player and the problem (a programming problem in this case) is the opponent [6]. Skills are updated at the end of each interaction between a subject and a object, according to the expectation of success and the observed result [6].

The Elo system classifies players based on results and success expectations [6]. The logical function of student's success probability estimation is given by

$$P(R_{ij} = 1) = \frac{1}{1 + 10^{\frac{\theta_j - \theta_i}{400}}}, \quad (3)$$

where $R = \{0,1\}$ is the result set of a game: 1 (win) and 0 (lose). Given a game between player i and player j, with Elo θ_i and θ_j, respectively, when the game ends, new Elos are calculated according to the result expectations, the previous

Elos and a constant k. The higher the k, the greater the Elo change [6]. The Elo is calculated as follows,

$$\theta_i = \theta_i + k(R_{ij} - P(R_{ij} = 1)). \quad (4)$$

Elo is considered a simple model in relation to its use in online environments and easy to implement in education systems. However, it is constrained to tracking only one skill [6].

V. Student's Multiple Abilities and Skills Model

The proposed skill estimation system, the Student's Multiple Abilities and Skills (SMAS) model, has as its main objective to estimate the multiple abilities of students in massive online systems. SMAS is based on performance expectation models: it integrates the MIRT model, estimating the probability of the student getting the programming problem right, with the Elo model, used for identification of the student's multiple skills in each problem solution.

The proposed model is suitable for massive programming learning systems, as programming problems have several ways to be solved, and each way involves certain skills of the student. For formality, each solution form is called *paths*. Each solution path is comprised of essential skills to solve the problem correctly. Furthermore, it is assumed that, in each *path*, the abilities are not compensating for each other. In this way, students need to have all the skills to develop a correct solution. It is understood that there is compensation between the solution *paths*, because even if the student has low skills, if the chosen path does not require these skills, it is possible that the student will be successful in the solution.

Figure 1 presents the paths and ability relations of the SMAS model. The model is represented by a graph, where each node stands for a skill, and each undirected edge represents the relationship between the skills. Given a problem j, a subgraph represents a solution *path* (s_n), where skills are not compensatory. Therefore, the SMAS model assumes that there is compensation between subgraphs, but does not allow compensation within a subgraph. Note that, in Fig. 1, a problem j can be solved using 4 skills: A, B, C and D. To solve this problem, there are 2 possible *paths* (s_1 and s_2): for the *path* s_1, the skills A, B and C are needed; and for path s_2, skills A and D are required. The student will solve the problem using either the *path* s_1 or the s_2, according to his abilities. In this problem, the skill E is not needed for its solution.

Figure 2 shows, in general, the methodology for applying the SMAS model.

Thus, each problem j has a set of *paths* to its solution, $j = \{s_1, s_2, .., s_n\}$, each set s is made up of n skills that have difficulty β and relevance α, $s = \{(\beta_1, \alpha_1), (\beta_2, \alpha_2), ..., (\beta_n, \alpha_n)\}$. Relevance is how important a skill is for the correct solution to the problem, given by a real value between 0 and 1. A relevance equal to 0 means that the skill is not needed; and, a relevance equal to 1, indicates that the skill is indispensable.

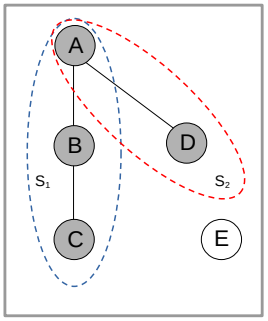

Fig. 1. Example of a problem with two paths to the solution

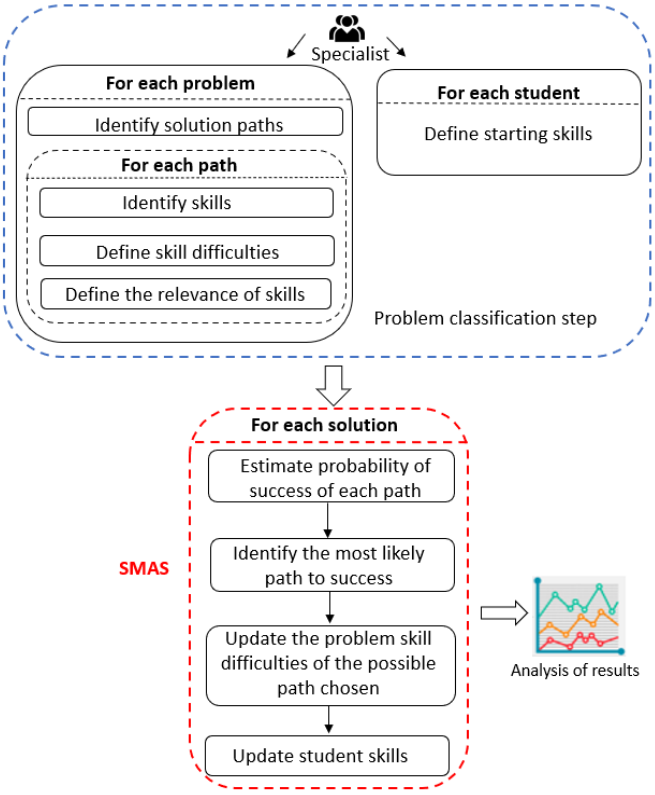

Fig. 2. Methodology of SMAS model

Whereas all skills are essential in every path s; then, for each *path*, the probability of success is calculated, using the non-compensatory MIRT, as shown in (5). As the model terms are multiplicative, the conditional probability of a correct answer is limited by the smallest of the obtained probabilities [24].

$$P_{is} = \prod_{n \in s}^{N} \frac{e^{(\alpha_{ns}(\theta_{in} - \beta_{ns}))}}{1 + e^{(\alpha_{ns}(\theta_{in} - \beta_{ns}))'}}, \quad (5)$$

where P_{is} is the probability that the student i gets the problem right with the *path* s; α_{ns} is the relevance of skill n in *path* s; β_{ns} is the difficulty of skill n on *path* s; and θ_{in} is the n skill of the student i.

According to Rossler [26], the tendency of individuals is to perform tasks by means that require less energy, time and thought. Therefore, it is understood that the student will choose the *path* that presents the highest probability of correct answers (the *path* considered easiest for a given problem). Thus, the probability of the question being correct assumes the highest probability among the *paths* ($P_{ij} = P_{is}$), as well as the problem parameters: difficulties and relevance of *path* skills with greater probability.

Once the probability of question correctness has been determined and with knowledge of a problem answer, the Elo model updates problem difficulty parameters and student's abilities [25], considering the difference between the observed performance Y_{ij} (response) and the expected performance P_{ij} (success probability).

$$\begin{aligned} \theta_{in(t)} &= \theta_{in(t-1)} + \alpha_{jn} K\{Y_{ij} - P_{ij}\}, \\ \beta_{jn(t)} &= \beta_{jn(t-1)} - \alpha_{jn} K\{Y_{ij} - P_{ij}\}, \end{aligned} \tag{6}$$

where β_{jn} and α_{jn} are the difficulty and relevance of skill n of problem j, respectively, both from the *path* with the highest probability of success; K is a constant equal to 32[2], which defines how much the estimate can be affected by the difference between the current and the expected response; Y_{ij} is student i's answer to problem j, being 1 for correct answer or 0 for error.

VI. Case Study

To validate the SMAS model, a case study was carried out, over a period of 2 weeks, in 3 classes of computing courses: 29 students from 2 technical-level classes and 8 higher-level course students, totaling 37 case study participants.

A set of 200 problems from *beecrowd* repository has been analyzed by an expert [26], who investigated all possible *paths* to solve each problem. For each *path*, a specialist identified the skills involved in the problem solution and associated a difficulty value and a relevance value, both varying between 0 and 1, to each problem. Information on the methodology for analyzing the problems can be consulted at [26].

Virtual classrooms were created on the *beecrowd* platform and a list of 30 exercises was registered. As most students are beginners at programming and have never used this platform, registered exercises are the simplest problems on the analyzed list. All problems were classified as *Beginner* in *beecrowd*, being they just structured problems, which require only simple mathematical calculations. Overall, not all exercises require the use of conditional and loop commands; and, some problems require the use of a one-dimensional array. Besides, all problems have more than one *path* of solution.

The skills involved in the problems are:

- Data analysis: identified regarding the use of arithmetic operators;
- Data representation: use of identifiers, arrays, data structures and pointers;
- Abstraction: use of logical operators, selection commands, repetition and recursion.

In a virtual room on *beecrowd*, the teacher has access to the student's submission history, as well as the source-code of all submitted solutions. This made it possible to compare and analyze *paths* chosen by students, as well as the *paths* indicated by the model for each solution, which is one of the case study objectives. Thus, source-codes were analyzed on the platform and data were tabulated. For each submission, students were related to the problems, received feedback, *path* chosen by the student, as well as submission date and time.

The student's initial skills were defined with a value equal to 1100. After collecting the data, they were organized in chronological order of submission. Next, the SMAS model was applied to verify the *paths* indicated by the model and analyze student's abilities.

A. Results and Discussion

During the case study period, 982 submissions were made, where 157 submitted solutions were correct and 825 submitted solutions were incorrect. So, there were 408 submissions with feedback *Wrong Answer*, 356 submissions with *Runtime Error* and 61 submissions with *Presentation Error*. Figure 3 shows the percentages of each feedback.

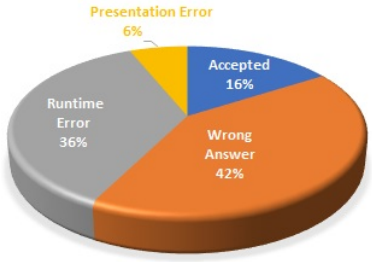

Fig. 3. Feedback percentage

The reason for the high number of incorrect submissions may be associated with the lack of experience from students. Analyzing the source-codes, it is possible to observe that many solutions were correct, but they were not accepted by *beecrowd* because the output data was not formatted as requested.

Observing only the submissions that received *Accepted* feedback (total of 157 submissions), in 100 submissions, the *paths* indicated by SMAS model were compatible with those chosen by students; and 57 submissions did not match. Figure 4 shows these percentages.

When analyzing only the wrong answers, it is observed that in 430 submissions the chosen paths did not match those indicated by the model, while 394 did, as shown in Fig. 5.

Note that paths indicated by the SMAS model vary along the submissions, even when a student submits a solution to the same problem several times. This occurred because the problem's skill difficulties changed according to the other student submissions, which is a characteristic of the Elo model, where problem parameters are modified after each interaction

[2]empirically adopted value, proposed in [6]

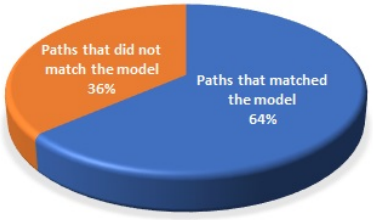

Fig. 4. Percentages of *paths* that matched and did not match student choices in correct submissions

Fig. 5. Percentages of *paths* that matched and did not match student choices in incorrect submissions

with the students. In general, the problem difficulty increases with each submission of an incorrect result, or decreases otherwise.

In many problems, there is not a big discrepancy in the relevance of abilities in each *path*, which makes the success probabilities in each *path* very close. Therefore, any of the *paths* indicated by the model require skills compatible with the students. Regarding the evolution of student's skills, the model allows individual analysis of each one. The graph of Fig. 6 shows the performance and evolution of a student (student 1) who performed 26 submissions to 4 problems. Among the submissions, 5 solutions were correct (feedback *Accepted*) and 21 solutions were incorrect or had an error.

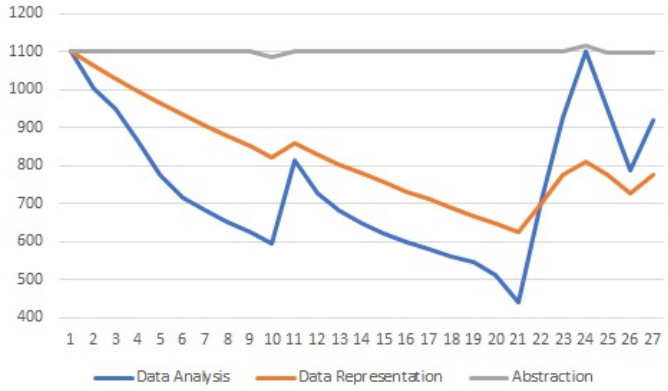

Fig. 6. Abilities of student 1

It is possible to observe from the graph that student 1 submitted solutions to problems that did not require, or required very little, the skill *Abstraction*. As the number of submissions considered wrong was greater than the correct ones, it implies a decrease in all student abilities.

Student 2 made 38 submissions to 5 problems, where there were 5 correct answers and 33 errors. The skills of this student are shown in the graph in Fig. 7. Among the submissions, in 26, the student chose the path indicated by the model. Again, the student had his abilities diminished due to the high number of wrong submissions.

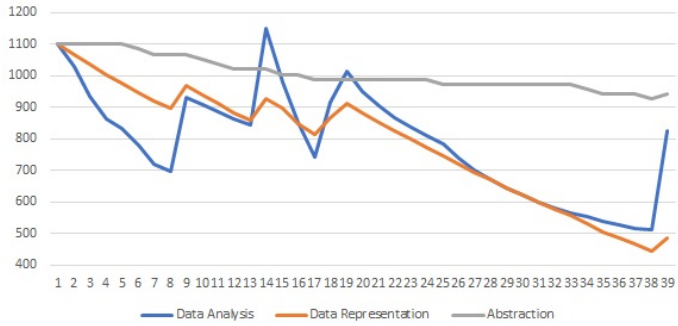

Fig. 7. Abilities of student 2

In the solved problems, it is observed that the relevance of *Data Analysis* and *Data Representation* is higher than the relevance of *Abstraction*. It can be observed in the graphs generated for the 2 discussed students.

Student 3, with 3 incorrect and 4 correct submissions, had a gain in all skills. Among the 7 submissions, in 5 exercises, this student chose the same *path* indicated by the SMAS model. The progress of the student's abilities can be seen in Fig. 8.

Fig. 8. Abilities of student 3

VII. Conclusion

In this paper, a new model to estimate the multiple abilities of students who solve programming exercises, named Student's Multiple Abilities and Skill model, or simply SMAS model, was proposed. The assumption underlying this model is that programming problems have more than one way to be

solved (*paths*), and each *path* involves multiple skills. Thus, students solve problems whose *path* has the highest probability of being correct. To corroborate this model, a case study was applied to 37 students on the computer course, from high school and university level. A virtual classroom was created on *beecrowd* Online Judge, using the problems provided by this platform. These problems were analyzed by an expert, who identified the *paths* for solving each problem, as well as the skills associated with their difficulties and relevance. The results obtained from the analysis of the student submissions were tabulated in chronological order of submission, where the proposed model was applied.

The results obtained indicate that it is possible to identify the solution *paths* adopted by students, through the success probability estimation of each *path*, which varies from student to student, according to their abilities. In addition to the *paths*, it is possible to analyze the student's skill evolution, as well as the skills most required in problem solution. It was observed that many students would greatly diminish their abilities, because they submitted too many incorrect solutions, which was not surprising, as they were beginners at programming and had no experience with submissions in Online Judge systems. Besides, Online Judge systems are very rigorous with data input and output standards, considering as error any answer that does not meet this standard, even if the logic of the solution is correct. One solution, when you have access to submission feedback, is to treat differently the update of error abilities, when feedback is distinct from *Wrong Answer*. In future work, some adjustments will be made to the SMAS model to deal with this update that penalizes errors. In other words, the skills will be updated according to feedback, not considering only errors or successes.

REFERENCES

[1] S. Wasik, M. Antczak, J. Badura, A. Laskowski, and T. Sternal, "A survey on online judge systems and their applications," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–34, 2018.

[2] W. X. Zhao, W. Zhang, Y. He, X. Xie, and J.-R. Wen, "Automatically learning topics and difficulty levels of problems in online judge systems," *ACM Transactions on Information Systems (TOIS)*, vol. 36, no. 3, pp. 1–33, 2018.

[3] R. D. Pea and D. M. Kurland, "On the cognitive effects of learning computer programming," *New Ideas in Psychology*, vol. 2, no. 2, pp. 137–168, 1984.

[4] S. Wasik, M. Antczak, J. Badura, A. Laskowski, and T. Sternal, "A survey on online judge systems and their applications," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–34, 2018.

[5] C. V. Giordano, L. N. de Lira, C. Langhi, and M. D. Feitosa, "Tecnologia de apoio ao ensino e aprendizagem de programação em graduações tecnológicas profissionais: Juiz on-line," *Boletim Técnico do Senac*, vol. 47, no. 2, 2021.

[6] R. Pelánek, "Applications of the elo rating system in adaptive educational systems," *Computers & Education*, vol. 98, pp. 169–179, 2016.

[7] D. F. de Andrade, H. R. Tavares, and R. da Cunha Valle, "Teoria da Resposta ao Item: conceitos e aplicações," *ABE, Sã Paulo*, 2000.

[8] F. B. Baker, *The basics of item response theory*. ERIC, 2001.

[9] D. F. de Andrade, H. R. Tavares, and R. da Cunha Valle, *Teoria da Resposta ao Item: conceitos e aplicações*. São Paulo: ABE - Associação Brasileira de Estatística, 2000.

[10] A. E. Elo, *The rating of chessplayers, past and present*. Arco Pub., 1978.

[11] G. Falckembach and F. Araujo, "Aprendizagem de algoritmos: dificuldades na resolução de problemas," *Proceedings Sulcomp*, vol. 2, 2013.

[12] A. Gomes, C. Areias, J. Henriques, and A. J. Mendes, "Aprendizagem de programação de computadores: dificuldades e ferramentas de suporte," *Revista Portuguesa de Pedagogia*, pp. 161–179, 2008.

[13] V. Barr and C. Stephenson, "Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community?" *Acm Inroads*, vol. 2, no. 1, pp. 48–54, 2011.

[14] J. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, 2006.

[15] R. L. de Souza, F. Z. Ferreira, and S. S. da Costa Botelho, "Proposta para avaliação de códigos fonte com tf-idf," in *Anais do XXXI Simpósio Brasileiro de Informática na Educação*. SBC, 2020, pp. 112–121.

[16] V. Barr and C. Stephenson, "Bringing computational thinking to k-12: what is involved and what is the role of the computer science education community?" *Acm Inroads*, vol. 2, no. 1, pp. 48–54, 2011.

[17] J. O. Chaves, A. Castro, R. Lima, M. V. Lima, and K. H. A. Ferreira, "Uma ferramenta baseada em juízes online para apoio às atividades de programação de computadores no moodle," *Renote*, vol. 11, no. 3, 2013.

[18] R. E. Francisco, A. P. L. Ambrósio, C. X. P. Junior, and M. A. Fernandes, "Juiz online no ensino de cs1-lições aprendidas e proposta de uma ferramenta," *Revista Brasileira de Informática na Educação*, vol. 26, no. 03, p. 163, 2018.

[19] J. L. Bez, N. A. Tonin, and P. R. Rodegheri, "URI Online Judge Academic: A tool for algorithms and programming classes," in *2014 9th International Conference on Computer Science Education*, 2014, pp. 149–152. [Online]. Available: https://www.urionlinejudge.com.br/

[20] A. P. Vargas, R. Santos, M. Neves, D. Teixeira, and S. S. da Costa Botelho, "Sistema de recomendação baseado no elo para problemas de pré-cálculo: Um experimento com calouros universitários," in *Anais do XXXI Simpósio Brasileiro de Informática na Educação*. SBC, 2020, pp. 1213–1222.

[21] R. T. Nojosa, "Teoria da Resposta ao Item (TRI): modelos multidimensionais," *Estudos em Avaliação Educacional*, vol. 1, no. 25, pp. 123–166, 2002.

[22] M. D. Reckase, "Multidimensional Item Response Theory," *Handbook of statistics*, vol. 26, pp. 607–642, 2006.

[23] L. Pasquali, *TRI–Teoria de Resposta ao Item: Teoria, procedimentos e aplicações*. Editora Appris, 2018.

[24] R. T. Nojosa, "Modelos multidimensionais para a teoria de resposta ao item," *Pernambuco, UFPE, Master Dissertation*, 2001.

[25] A. Prisco, R. Santos, S. Botelho, N. Tonin, and J. Bez, "A multidimensional Elo model for matching learning objects," in *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2018, pp. 1–9.

[26] W. d. O. Paes, "Habilidades necessárias para resolução de problemas de programação aplicados em sistemas de recomendação," 2022.