Monitoring the Progress of Programming Students Supported by a Digital Teaching Assistant

Nuno Gil Fonseca[1,2(✉)], Luís Macedo[1], and António José Mendes[1]

[1] Department of Informatics Engineering, Center for Informatics and Systems of the University of Coimbra, University of Coimbra, Coimbra, Portugal
{nunogil,macedo,toze}@dei.uc.pt

[2] Instituto Politécnico de Coimbra - Escola Superior de Tecnologia e Gestão de Oliveira do Hospital, Oliveira do Hospital, Portugal

Abstract. Several studies have shown that there is an important link between continual monitoring by the teachers and the students' performance. Unfortunately, the teachers cannot be continuously looking for what the students are doing. To overcome this situation, we propose the use of *CodeInsights*, a tool capable of capturing, in an autonomous, transparent and unobtrusive manner, information about the students' performance and then, based on teacher's expectations, notify them about possible deviations in the specific context of programming courses. The decision on whether the system should or should not notify the teacher is supported by an artificial cognitive selective attention mechanism. Although *CodeInsights,* provided with the described mechanism, hasn't been fully tested in a real case scenario, we present some specific examples of how it can be used to assist teachers.

Keywords: Intelligent systems in education · Digital teaching assistant · Programming · Selective attention

1 Introduction

Now more than ever, programming is becoming more and more part of the student's life, even on early stages of their lives [1]. However, programming is not a trivial task, and the students experience a series of difficulties along the way [2]. Due to factors like inhibition, the students may be facing difficulties, and the teacher might be totally unaware of them. As those difficulties can be relevant to their progress, their non-clarification may lead to drastic consequences in terms of the learning process. Furthermore, when the students are capable of expressing their problems to the teachers, it is always difficult trying to understand what the student already knows and has done to solve the problem in hand. This lack of information sometimes leads the teachers to solve the problem themselves without explaining its origin to the students.

Finding ways to overcome these barriers between teachers and students, which includes the problems faced by the teachers in becoming aware whether or not the students are facing difficulties, as well as their nature, has given rise to research on

DOI: 10.1007/978-3-319-65340-2_7

teaching methodologies. As stated by Bloom [3], the most efficient teaching-learning method is individualized tutoring, since the teacher is more aware of the students' difficulties and capabilities, and can more easily adapt the teaching pace, as well as the materials and assignments, proposed to the students. Also, Raabe and Silva [4] claim that students' performance is highly coupled with the capacity that the teachers have to detect their difficulties and react within a useful time. In the specific context of face-to-face learning with a large number of students, this proximity relationship between teachers and students may be hard to achieve.

Finding strategies that increase this proximity relationship has driven researchers to make use of technology. In fact, strategies that include technology in education have provided a promising solution to overcome this problem. One of those strategies includes using agent-based technology. For instance, according to Jafari [5], there are three groups of intelligent agents for teaching and learning applications: digital classmate, digital teaching assistant, and digital secretary. Among these groups, especially related to our interests is the group of digital teaching assistants. In this category, we find those systems whose main purpose is to autonomously collect and analyze data about the student's work, and display it to the teacher in the most suitable way. Eventually, the systems may also generate notifications to the teacher whenever a deviation from the usual path is found. As examples of this type of systems, we may find the *Logic-ITA* system [6], the *AVA* system proposed by Raabe and Silva [4], the teaching assistant agent proposed by Choy et al. [7] or the *AmI-RIA*: Real-Time Teacher Assistant proposed by Mathioudakis et al. [8]. Particularly related to the context of programming, we may find the *ELJT* (E-Learning Java Trainer) [9], the *Retina* system proposed by Murphy et al. [10], the *TestMyCode* system [11] or *ClockIt* [12].

A significant limitation of some of these systems is that they may be highly coupled to a specific set of assignments, or the students may need to use a specific development methodology (e.g. the Test Driven Methodology must be utilized with the TestMyCode tool). Also relevant is the fact that some tools were designed to analyze the code snapshots[1] *a posteriori*, and not in real time, which in our opinion is a significant drawback.

In most cases, these systems are only used to collect and process data coming from the students and make the resulting information available to the teacher. The teacher should then browse through the data available and try to make sense of it, which obviously is very time-consuming. For this purpose, systems should include some notification system that will alert the teachers about specific aspects of the student's performance (e.g. unfinished assignments, plagiarism of code). However, if not carefully designed, systems could rapidly start to notify the teachers about everything, which will lead to a scenario where the overload of information will make the system impossible to use. To avoid this information overload scenario, systems may include selective attention mechanisms, capable of identifying the most relevant information to display to the users.

Although there has been much research in this field, there is not a general theory/model of selective attention. In spite of this, a number of models of selective

[1] A code snapshot is a copy of the source code written by the student to solve a designted assignment at a given moment in time.

attention have been proposed in Cognitive Science (e.g., [13–15]). Particularly related to these models is the issue of measuring the value of information. A considerable amount of literature has been published on these measures, especially from the fields of active learning and experimental design. Most of those measures rely on assessing the utility or the "informativeness" of information (see [16] for more details).

A number of models of selective attention have been proposed in Cognitive Science (e.g., [17]). Particularly related to these models is the issue of measuring the value of information. A considerable amount of literature has been published on these measures, especially from the fields of active learning and experimental design. Most of those measures rely on assessing the utility or the informativeness of information (e.g., [18–21]).

In this paper, we adopt the model of selective attention proposed by Macedo [16]. We integrate it into one digital teaching assistant tool to decide if a certain notification should or nor be generated. This tool should be capable of autonomously capturing information about the students' performance and, based on the teacher's beliefs, notify them about possible deviations whenever needed. This way, the teacher can be more aware of the students' performance and be capable of making more educated decisions and at the same time avoid an overabundance of information.

The next section presents Macedo's model of selective attention while Sect. 3 describes how this model is incorporated in our system, called *CodeInsights* [22]. Finally, we discuss our approach and present conclusions.

2 Background Macedo's Model of Selective Attention

The artificial selective attention mechanism that we are using is an implementation of the model proposed by Macedo [16] which has already been used and validated in several different scenarios. This artificial selective attention mechanism is comprised of three distinct components: (i) Surprise Value of Information, (ii) Uncertainty-based Value of Information and (iii) Motive Congruence/Incongruence-based Value of Information.

According to the "Principle of Selective Attention" stated by Macedo, "*A resource-bounded rational agent should focus its attention only on the relevant and interesting information*". Macedo has defined three different real numbers α, β, and γ as levels above which the absolute values of motive congruency (*MC*), surprise (*S*) or information gain (*IG*), respectively, should be such that the information can be considered valuable or interesting according to the following algorithm.

```
IF($\boldsymbol{MC} > \boldsymbol{\alpha}$ AND($\boldsymbol{S} > \boldsymbol{\beta}$ OR $\boldsymbol{IG} > \boldsymbol{\gamma}$)):
    generate notification
ELSE:
    discard notification
```

The initial values for the thresholds are automatically defined based on previous experiments, but the teachers can change them at any time according to their needs.

The Uncertainty-based Value of Information component consists basically of determining the gain of information, *IG*, about the acquisition of new information in an experiment. It does not matter what event in fact happened. The value for *IG* can be determined as described in (1), where *m* stands for the number of mutually exclusive events of experiment *E,* and H_{prior} and H_{post} are, respectively, the entropies before and after the acquisition of new data that might change the probability distribution of the event(s) of interest.

$$IG(E) = \frac{H_{prior}(E) - H_{post}(E)}{\log(m)}$$
$$IG(E) = \frac{-\sum_{i=}^{m} P_{prior}(E_i) \times \left(\log P_{prior}(E_i)\right) - \left(-\sum_{i=}^{m} P_{post}(E_i) \times \left(\log\left(P_{post}(E_i)\right)\right)\right)}{\log(m)} \tag{1}$$

In our particular case, H_{post} is equal to zero, because we are assuming that the experiment has ended (i.e. no more attempts will be received after the deadline has passed) and therefore all the $P_{post}(E_i)$ but one are zero, this one having the value unity. Thus, only when we are confident of the outcome does H_{post} vanish. Otherwise, it is positive. So the value for *IG* in these cases is obtained by (2):

$$IG(E) = \frac{-\sum_{i=}^{m} P_{prior}(E_i) \times \left(\log\left(P_{prior}(E_i)\right)\right)}{\log(m)} \tag{2}$$

Next, we will use the Surprise Value of Information component, which is used to determine the extent of the surprise felt by the agent (representing the teachers) when a certain event occurs. According to Macedo et al., the intensity of surprise about an event E_g, from a set of mutually exclusive events E_1, E_2, ..., E_m, is a nonlinear function of the difference, or contrast, between its probability and the probability of the most expected event E_h in the set of mutually exclusive events E_1, E_2, ..., E_m. So, the value of surprise *S* of some event E_g, $S(E_g)$, is given by (3);

$$S(E_g) = log\left(1 + P(E_h) - P(E_g)\right) \tag{3}$$

For this purpose, we assume that the set of mutually exclusive events has at least one event whose occurrence is unsurprising (has a higher probability to occur), namely, E_h.

Finally, the Congruence/Incongruence-based Value of Information component is used to assign different degrees of desirability (satisfaction) for receiving notifications about the occurrence of each event (i.e. the teacher may have more satisfaction with the occurrence of some events than with the occurrence of others). Instead of probabilities or expectations, this component takes as input "desires" that can be satisfied or frustrated independently of the probability of occurrence for that event. The "desires" are expressed as a value between −1 and 1. It is important to notice that the negative value for *D* is used to represent "unhappy desires" (i.e. the teacher is not "happy" with that, but still wants to be notified). In the end, the motive congruence value for the outcome of an event – *MC* – is equal to $|D|$.

As mentioned previously, the system will only generate notifications about the occurrence of those events for which $MC > \alpha$ AND $(S > \beta$ OR $IG > \gamma)$. The "OR" between surprise and information gain can easily be explained by looking at Fig. 1. This figure depicts the evolution of the values of S and IG (YY axis) using different values for the probability of the occurrence of a certain event (XX axis), assuming that there are only two possibilities.

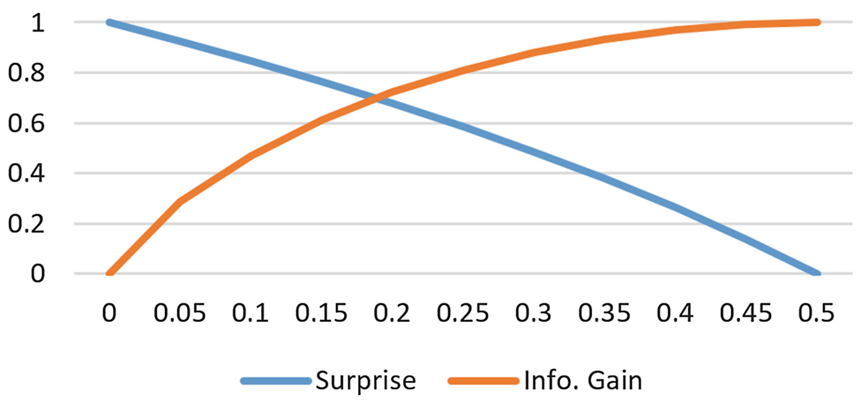


Fig. 1. The distribution of the values of surprise and information gain using different values for the probability of the occurrence of a certain event

As can be seen in Fig. 1, the value for the gain of information increases as the probability of the occurrence of a certain event increases up to 0.5, which means that the occurrence of an event in a very homogenous scenario is more likely to reduce uncertainty than otherwise. In contrast, the value of surprise decreases as the difference between the probability of occurrence of the two events also decreases. Because of this contrast, the OR is used to accommodate those scenarios where the occurrence of an event may not represent a huge gain of information, but will definitively be surprising, and the reverse scenario as well.

3 Using the Selective Attention Mechanism in *CodeInsights*

Our main goal is to develop a tool capable of automatically collecting and analyzing data coming from the students in the form of code snapshots (a copy of their code) and provide the teachers information about how their students are performing.

The code snapshots are automatically sent to a central server each time the students run the developed code, regardless of where they are (at the classes, at home, etc.). When received, the snapshots are automatically processed, and various pieces of information about the students' performance is extracted and immediately made available to the teachers in the form of aggregated statistical data and data visualizations, as well as more individualized information about each student.

In Fig. 2 is shown a partial view of the main dashboard of *CodeInsights* from which the teacher has access to data and visualizations about a series of aspects related to the student's performance such as the more/less popular assignments, daily/hourly distribution of the attempts performed by the students, the most common compilation errors, the amount of attempts made during the classes period (and outside that period), and several other.

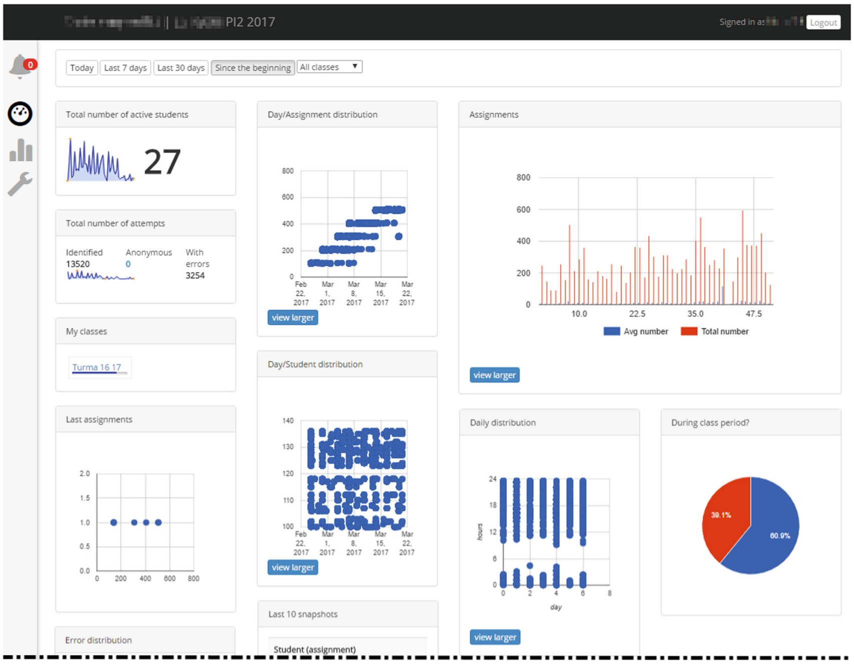


Fig. 2. Partial view of the main dashboard of *CodeInsights* which contains a series of data visualizations about the students' performance and behavior

Based on this information, the teacher could "manually" identify some deviations from the normal learning path and provide feedback or take the appropriate measure, via the most suitable means: verbally, directly in *CodeInsights*, via email, Moodle or any other existing platform. However, the teachers most likely don't have time to be constantly looking at the charts or browsing through the data and snapshots within *CodeInsights*. So, for this purpose, we have developed an automatic notification system, which will notify the teacher about a series of aspects concerning each student's performance such as unusual number of lines of code, compilation errors, attempts per assignment, assignments not even attempted, or a number of "unfinished" assignments. In reality, *CodeInsights* is capable of generating notifications about virtually any aspect that can be observed from the data.

Since there are plenty of situations where the system may automatically generate notifications to the teachers about different aspects of a student's performance, it is possible that the system may generate such an overload of information that the teacher may be unable to process it all in useful time. Because of this potential overload, we have decided to incorporate into our system the cognitive mechanism of selective attention previously presented.

Let's, for instance, assume that the teacher is interested in being notified when the percentage of students that have attempted to solve a given assignment is unusual. Additionally, let's consider that we have two assignments A1 and A2 with the probability distribution according to the teachers beliefs/expectations of student completions depicted in Tables 1 and 2.

Table 1. Example of teacher's beliefs for assignment A1

A1		
% of students		Probability
A	<= 20%	0.10
B	]21–79]%	0.75
C	>= 80%	0.15

Table 2. Example of teachers's beliefs for assignment A2

A2		
% of students		Probability
A	<= 20%	0.60
B	]21–79]%	0.25
C	>= 80%	0.15

Estimates for these probabilities can be calculated using aggregated data from previous editions of the courses, or simply using subjective probabilities defined by the teachers based on their personal experience.

Applying the Equation nº2 to the two examples presented above, we obtain for A1 an information gain of 0.665 and 0.853 for assignment A2.

As mentioned previously, the teachers can change the value for the γ threshold (see Sect. 2). High value for γ mean that the teacher is only interested in receiving notifications about situations that represent a high gain of information (i.e. that will contribute to decreasing the uncertainty). The exact opposite happens otherwise – a very low value for γ will mean that even if that event does not help to reduce the uncertainty, the teacher still wants to be informed about its occurrence. This latter scenario should be avoided, because it may result in receiving huge number of notifications.

Assuming that the system is initiated with 0.65 as the value for γ, for the particular scenarios described above, this would mean that any of the possible outcomes for the assignments A_1 and A_2 would be considered valuable. If the value for γ is defined to be 0.8, since only $IG(A_2) > 0.8$, only the occurrence of events related with the assignment A2 would be considered valuable.

Now, let's assume that after the deadline, the percentage of students that had tried to solve this assignment A_1 is 13% (A occurred), using the Equation nº3 we would have a surprise value of 0.72. If the percentage of students attempting A_1 was, for instance, 87% (occurrence of C), the value of the surprise would be 0.67.

An event is considered surprising if the value of surprise is higher than the threshold β previously mentioned (see Sect. 2). Assuming that the value for β is 0.65, the surprise value for the occurrence of these events is greater than β, so we can consider that these events are surprising enough that the teacher should be notified of their occurrence.

If, for the same assignment, the percentage of students that attempted to solve it was, for example, 55% (occurrence of B), the surprise value would be equal to zero - the occurrence of the most probable (expected) event generates no surprise at all.

As happens with the other thresholds, the value for β is defined by default to a certain value, but the teacher can increase or decrease this value at any time. Higher values of β indicate that only very unexpected events (with high surprise values) are considered interesting. If a lower value for β is used, the system will consider more events as being surprising enough. However, that could easily lead to the information overload situation that we are trying to avoid.

Finally, let's focus on the motive congruence component. In our particular scenario, let's suppose that the teacher has defined a certain desirability value, ***D***, for each event as shown in Table 3.

Table 3. Example of a table of desirabilities

A1	
% of students	D
<= 20%	−0.85
]20–79]%	0.40
>= 80%	+0.75

These desirabilities indicate that the teacher would feel very satisfied if there is a large portion of students submitting this assignment A1, and very unsatisfied if there is a low percentage of students submitting this assignment, and would feel almost indifferent if the percentage of students submitting is moderate (21-79%).

Assuming the principle that people want to know about situations that are either consistent or inconsistent with their goals, this might also mean in this particular case of the teacher, that it doesn't make sense to notify her/him when the percentage of students that attempted solving A1 is between 21 and 79%. On the other side, it makes great sense to be notified about the fact that the number of attempts for this assignment exceeds 80% or is lower than 20%.

Assuming that the teacher has defined 0.1 as the value for α, in our scenario regardless of the event that occurs, s/he will always be notified, since ***MC*** is always greater than α. In this case, the teacher could well face a situation of the overflow of undesired notifications, because the value for the threshold is set to a very low value. If the value for α is much higher, 0.8 for instance, it means that the teacher only wants to be notified about the events that really (un)satisfy him/her. In this example, the teacher will only be notified when the percentage of students is lower than 20.

It is important to reinforce that these tables of desirabilities are defined by the teachers and can be considerably different for each assignment. For instance, there may be some assignments for which the teacher may want to be notified, even if the percentage of students that attempt to solve it is entirely "normal".

To conclude, let's have a look on how the system behaves in some of the scenarios presented previously. In Tables 4, 5 and 6 are shown some examples of how the system

Table 4. Using high values for all the thresholds

Assignment	Occurred	Prob.	IG	S	D	γ	β	α	
A1	a(<20 %)	0.10	0.665	0.72	0.85	0.85	0.85	0.85	NO
A1	b(<20 %)	0.75	0.665	0	0.40	0.85	0.85	0.85	NO
A1	c(>80 %)	0.15	0.665	0.67	0.75	0.85	0.85	0.85	NO

Table 5. Using low values for all the thresholds

Assignment	Occurred	Prob.	IG	S	D	γ	β	α	
A1	a(< 20 %)	0.10	0.665	0.72	0.85	0.35	0.35	0.35	OK
A1	b(<20 %)	0.75	0.665	0	0.40	0.35	0.35	0.35	OK
A1	c (>80 %)	0.15	0.665	0.67	0.75	0.35	0.35	0.35	OK

Table 6. Using moderate values for all the thresholds

Assignment	Occurred	Prob.	IG	S	D	γ	β	α	
A1	a(<20 %)	0.10	0.665	0.72	0.85	0.70	0.65	0.80	OK
A1	b(<20 %)	0.75	0.665	0	0.10	0.70	0.65	0.80	NO
A1	c(>80 %)	0.15	0.665	0.67	0.75	0.70	0.65	0.80	NO

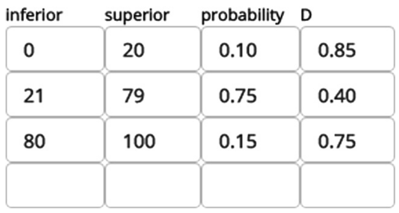
would behave in different scenarios. The heighted values for IG, S and D correspond to values below the thresholds. In the last column, the "OK/NO" indicate whether or not a notification on the occurrence of that event should be generated.

As expected, the specific values used for the thresholds are essential for the success of this artificial selective attention mechanism. Because of that, special attention should be taken when defining those values.

Specifying the expectations about a certain event in *CodeInsights* is very easy, the teacher just needs to decide if s/he wants to define expectations about an event related to a particular assignment, a particular class or a global event. The teacher can also define expectations about the occurrence of a certain event concerning a particular student. In any case, the teacher must specify the date and time when the system will check if the expectations are met or not; the type of verification to be performed; and for each possible event, the expectations interval (inferior - superior), the probability of the occurrence of that event and finally, the value of the desirability - D.

For instance, let's suppose that the teacher wants to introduce into *CodeInsights* the expectations and desirabilities shown in Table 4. To accomplish this, the teacher just needs to fill a form similar to the one depicted in Fig. 3.

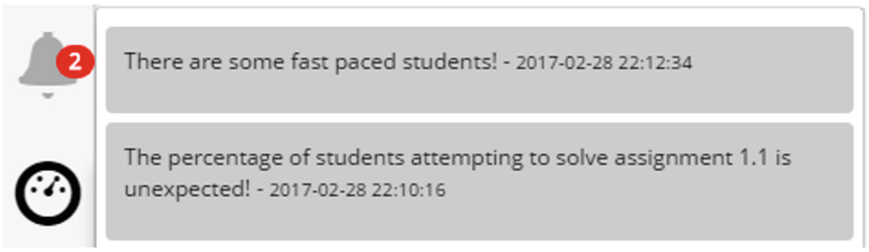
On the specified date and time, *CodeInsights* will automatically perform the necessary operations to check if the expectations about a certain event are met or not, and



inferior	superior	probability	D
0	20	0.10	0.85
21	79	0.75	0.40
80	100	0.15	0.75

Fig. 3. The form used to insert the expectations and desires chosen by the teacher

according to these expectations and desirabilities defined by the teacher, determine if a new notification should be generated or not. If so, the new notification is generated and presented immediately to the teacher if s/he is using *CodeInsights* at that moment. Otherwise, when the teacher logs in the system, a list of all the new notifications will be available, as shown in Fig. 4.



There are some fast paced students! - 2017-02-28 22:12:34

The percentage of students attempting to solve assignment 1.1 is unexpected! - 2017-02-28 22:10:16

Fig. 4. Example of the notifications list

The teacher can then explore those notifications and take the remedial measures considered necessary to overcome the problems that some students may be facing.

4 Discussion and Conclusions

In this paper, we have presented *CodeInsights*, a monitoring tool capable of collecting data from students in the form of code snapshots and, with the information that results from processing these snapshots, notify the teachers about the situations where their intervention may be crucial.

To avoid exposing the teacher to an overload of notifications, we have decided to integrate into the system a cognitive artificial selective attention mechanism capable of presenting to the teachers only the most relevant notifications, according to the teacher's expectations/beliefs.

It is important to notice that the use of the proposed system *per se* will not have any direct impact on the students' performance. The system provides data, visualizations, and notifications to the teachers, but if they do not check the system regularly and, if based on the information provided, they do not take the most appropriate measures, then the impact of using the system will be null.

Due to academic schedule constraints, we have not yet been able to perform a full evaluation of *CodeInsights* with the cognitive artificial selective attention mechanism integrated. However, preliminary tests that we have performed so far have shown us that the system is, in fact, working as supposed. The system is selecting only the most relevant notifications to present to the teacher, according to the criteria defined by the principle of selective attention.

In the current version of the system, the teacher has to set the values for the expectations, desirabilities and thresholds manually. Although it can be done in a very easy way, we plan to develop a "baseline" student model based on data from previous editions of the course. This way, the teacher would only need to specify the expectations and desirabilities for new assignments or if s/he wants to overwrite certain values (since the students are not the same, some values may not make sense for the new group of students).

CodeInsights was planned to be used in the context of face-to-face classes. Nevertheless, it will also be of great (if not greater) importance in the context of distance learning, where in most cases there is a considerable geographical and even temporal distance between teachers and students.

References

1. Fessakis, G., Gouli, E., Mavroudi, E.: Problem solving by 5–6 years old kindergarten children in a computer programming environment: a case study. Comput. Educ. **63**, 87–97 (2013)
2. Gomes, A., Mendes, A.J.: Learning to program - difficulties and solutions. Presented at the International Conference on Engineering Education September (2007)
3. Bloom, B.S.: The 2 sigma problem: the search for methods of group instruction as effective as one-to-one tutoring. Educ. Res. **13**, 4–16 (1984)
4. Raabe, A., Silva, J.: Um Ambiente para Atendimento as Dificuldades de Aprendizagem de Algoritmos. Presented at the Anais do XXV Congresso da Sociedade Brasileira de Computação (2004)
5. Jafari, A.: Conceptualizing intelligent agents for teaching and learning. Educause Q. **25**, 28–34 (2002)
6. Yacef, K., University of Sydney. School of Information Technologies: Experiment and Evaluation Results of the Logic-ITA. School of Information Technologies, University of Sydney, Sydney (2003)
7. Choy, S.-O., Ng, S.-C., Tsang, Y.-C.: Building software agents to assist teaching in distance learning environments. In: Fifth IEEE International Conference on Advanced Learning Technologies (ICALT 2005), pp. 230–232 (2005)
8. Mathioudakis, G., Leonidis, A., Korozi, M., Stephanidis, C.: Real-time teacher assistance in technologically-augmented smart classrooms. Int. J. Adv. Life Sci. **6**, 62–73 (2014)
9. Mendes, A.J., Ivanov, V., Marcelino, M.J.: A web-based system to support Java programming learning. Presented at the CompSysTech 2005 - International Conference on Computer Systems and Technologies, June 2005
10. Murphy, C., Kaiser, G., Loveland, K., Hasan, S.: Retina: helping students and instructors based on observed programming activities. In: Proceedings of the 40th ACM Technical Symposium on Computer Science Education, pp. 178–182. ACM, New York (2009)

11. Vihavainen, A., Vikberg, T., Luukkainen, M., Pärtel, M.: Scaffolding students' learning using test my code. In: Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education, pp. 117–122. ACM, New York (2013)
12. Norris, C., Barry, F., Fenwick Jr., J.B., Reid, K., Rountree, J.: ClockIt: collecting quantitative data on how beginning software developers really work. In: Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education, pp. 37–41. ACM, New York (2008)
13. Kahneman, D.: Attention and Effort. Prentice-Hall, Upper Saddle River (1973)
14. Wright, R.D., Ward, L.M.: Orienting of Attention. Oxford University Press, Oxford (2008)
15. Feldman, H., Friston, K.: Attention, Uncertainty, and Free-Energy. Front. Hum. Neurosci. **4** (2010)
16. Macedo, L.: Arguments for a computational model for forms of selective attention based on cognitive and affective feelings. In: 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction, pp. 103–108 (2013)
17. Horvitz, E., Jacobs, A., Hovel, D.: Attention-sensitive alerting. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, pp. 305–313. Morgan Kaufmann Publishers Inc., San Francisco (1999)
18. Horvitz, E., Barry, M.: Display of information for time-critical decision making. In: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, pp. 296–305. Morgan Kaufmann Publishers Inc., San Francisco (1995)
19. MacKay, D.J.C.: Information-based objective functions for active data selection. Neural Comput. **4**, 590–604 (1992)
20. Lindley, D.V.: On a measure of the information provided by an experiment. Ann. Math. Stat. **27**, 986–1005 (1956)
21. Settles, B.: Curious Machines: Active Learning with Structured Instances (2008)
22. Fonseca, N.G., Macedo, L., Mendes, A.J.: CodeInsights: monitoring programming students' progress. In: Proceedings of the 17th International Conference on Computer Systems and Technologies 2016, pp. 375–382. ACM, New York (2016)