Enhancing Student Engagement via Reduction of Frustration with Programming Assignments using Machine Learning

Mario Garcia Valdez[1], Amaury Hernandez Aguila[1], Juan-J. Merelo[2] and Alejandra Mancilla Soto[1]
[1]*Dept. of Graduate Studies, Instituto Tecnologico de Tijuana, Mexico*
[2]*Geneura Team and CITIC, University of Granada, Spain*

Keywords: Affective Computing, Neural Networks, Learning Analytics.

Abstract: Learning to program is often regarded as a difficult task. When selecting an appropriate programming exercise, experienced instructors gauge a studentś affective state and skills to then assign an activity with the appropriate level of difficulty. This work is focused on the prediction of the affective states of programmers with different levels of expertise when learning a new programming language. For this, an interactive web-based programming platform is proposed. The platform is designed to collect data from the studentsínteraction for data analysis. Current work is focused on the prediction of affective states using non-obtrusive sensors. Specifically, the aim of this research is to evaluate the use of keyboard and mouse dynamics as an appropriate sensory input for an affective recognition system. The proposed method uses feature vectors obtained by mining data generated from both keyboard and mouse dynamics of students as they work in basic Python programming assignments, which were used to train different classification algorithms to classify learners into five different affective states: boredom, frustration, distraction, relaxation and engagement. Accuracy achieved was around 75% with J48 obtaining the best results, proving that data gathered from non-obtrusive sensors can successfully be used as another input to classification models in order to predict an individualś affective states.

1 INTRODUCTION

Introductory programming courses are generally regarded as difficult (Robins et al., 2003; Lahtinen et al., 2005) and there is a common conception that they often have a high failure rate (Bennedsen and Caspersen, 2007). Jenkins (Jenkins, 2001) argues that there are multiple factors involved in this lack of success:

- There is a deep relation with the expectations, attitudes, and previous experiences of the teaching staff and students.
- The nature of the subject that involves the learning of new abstract constructs, syntax and tools.
- Groups are often heterogeneous and thus it is difficult to design courses that are beneficial for everyone. Many students begin to program when they are at their first year of university which means they are tackling a totally new topic that does not respond to their habitual study approaches.

On the other hand, Dijkstra (Dijkstra et al., 1989) argues that the subject of programming is very problem-solving intensive, which means that a high precision is required because even the slightest perturbation can render a program totally worthless. These difficulties often frustrate students. Jenkins (Jenkins, 2001; Jenkins, 2002) notes that many students expect the course to be difficult and come with the notion that they will have to struggle, others may have a stereotyped image of a programmer, these beliefs can negatively affect their initial motivation. During the course, novice programmers experience many emotions, for instance frustration and confusion when they cannot find a bug in a program, but also joy when they successfully run a challenging program for the first time. They can also become bored if they found the exercises too repetitive or too easy. Learning to program is a difficult task, where emotions play a significant role. Research has gone far to understand the role of emotion in learning, both in the process and the outcome; for instance, in e-learning (Kort et al., 2001; Rossin et al., 2009) and in programming (Rodrigo et al., 2009; Jenkins, 2001; Bosch et al., 2013; Khan et al., 2007).

In these works the emotion of flow/engagement is regarded as the ideal affective state in which students tend to be most capable of acquiring meaningful information through the learning process. Flow is defined by Csikszentmihalyi (Csikszentmihalyi, 1990) as a mental state in which a person performing an activity is fully immersed in a feeling of energized focus, full involvement, and enjoyment. Engagement is also a positive affect where students are thought to be more involved behaviorally, intellectually, and emotionally in their learning tasks (Bangert-Drowns and Pyke, 2002).

In order to promote engagement, instruction is designed with the objective of assigning learning activities that result challenging to students, but not much as to frustrate them. For this, experienced instructors gauge a studentś affective state and then assign an activity with the appropriate level of difficulty. However, recognition of emotions is a difficult task, even for humans. Instructors often use their social intelligence to recognize students ́ affective states. In class a good instructor habitually reads the faces of students in the classroom to see if they are confused, bored or engaged; then decides what to do next. Interactive Learning Environments (ILEs) can be enhanced if they can offer an automatic perception of emotions. The recognition and simulation of human affects are becoming important fields of study, and many researchers have demonstrated that affect-aware computers can provide a better performance in assisting humans (Picard et al., 2001). When ILEs embrace these methodologies and techniques a problem arises. Sensors must be used to gather data, in order to perform the recognition of usersáffective states.

In programming courses, students solve programming problems by programming their proposed solution in a particular language; in order to do this, they must write the code, some times compile it, and finally run it to see how it works. While doing this, data can be collected, all the dynamics of actually writing the code in the keyboard, errors, corrections, elapsed time, number of attempts, compiling errors, exceptions, among others. Data mining this could bring us a better understanding of the learning process, and also a better model to both the selection of the appropriate content and to evaluate the learnerś competencies. In this work, a method for the recognition of affective states through the mining of keystroke and mouse dynamics data is proposed. The hypothesis is that students can vary the dynamic of keystrokes according to their affective state when programming. A correlation of the userś interactions (keyboard and mouse) with an emotional state, has been established in the preliminary work by Zimmermann et al. (Zimmermann et al., 2003), but the work experiments where conducted by asking users to write a predefined message, not programming execercises. In order to test the correlation of keyboard and mouse dynamics and an affective state when programming, an interactive web-based programming platform is proposed. The platform is designed to collect keyboard and mouse data from the studentsínteraction for data analysis. The proposed method has three main components: a JavaScript library to capture keyboard and mouse dynamics from a browser-based editor, a pre-processing step whose output is a feature vector that is sent to the third component, a classification algorithm. An experiment was conducted where students solved a series of programming exercises using the web based learning environment. Using only the data from the studentś keyboard and mouse dynamics six affective states where recognized for each of the attempts at solving the exercises. Results obtained from the experiment are promising. The affective states recognized include the most common in novice programmers according to the study of Bosch, D'Mello & Mills (Bixler and D ́Mello, 2013): boredom, engagement/flow, confusion and frustration. For this experiment four binary classifiers were compared: J-48 decision trees, k-nearest neighbour classifier, feed forward neural network, and a naïve Bayes. The output of each classifier determined if the student experienced the affective state during the exercise. In order to determine what affective states a student was experiencing, an Experience Sampling Method (ESM) was used by Csikszentmihalyi & Larson (Kubey et al., 1996). After the students successfully solve a programming exercise, they are presented with an ESM survey that asks what they were feeling during their solving of the exercise.

The structure of this work is organized as follows: next Section presents a series of works related to the proposed method in this paper; Section 3 describes the proposed method for the recognition of affective states in a web based learning environment for the teaching of programming languages; next in Section 4 we explain the experimental evaluation of the proposed method and results. Finally, we draw some conclusions about our methodology and experiments.

2 RELATED WORK

Affect recognition is an active field of research. Many methods have been proposed, some require the intervention of the user to fill up questionnaires or forms; these selections remain static until the user changes the values. These methods are easy to implement, but cannot detect dynamic changes. A more dynamic approach requires the use of sensors to capture affective states as they change. The context, the environment and the learning activity determine what kind of sensors can be used. The most common learning environment is the classroom, a physical space with context to facilitate learning. But learning is possible in a wide variety of settings, such as outside-of-school locations and outdoor environments these are sometimes referred as ubiquitous learning environments, an example of such environments is the work of Yang (Yang, 2006) in a context-aware environment, but missing affective information. There are also virtual learning environments such as the one proposed by Dillenburg (Dillenbourg et al., 2002) where learners can have avatars, and virtual places where they can play roles and socialize.

In the general context of learning there have been some approaches for affect recognition. The work of Kapoor y Picard (Kapoor and Picard, 2005) uses a multi modal approach using sensory information from facial expressions and postural shifts of the learner combined with information about the learnerś activity on the computer; the learning activity was solving a puzzle on a computer. They report using a multi-modal Gaussian Process approach achieving an accuracy of over 86%. Sidney, et al., (Sidney et al., 2005) enhances the intelligent tutor system AutoTutor with affective recognition using non-intrusive sensory information from facial expressions, gross body movements, and conversational cues from logs. The subject material of that system consisted in lectures of computer literacy.

Elliott, Rickel y Lester (Elliott et al., 1999; D´Mello et al., 2008) propose the integration of an affective reasoner with an agent in a virtual environment. In this work an agent called Steve responds emotionally to other agents and interactive users. The agent simulates his emotions through different multimedia modes including facial expressions and speech. Affective reasoning agents were used for training, by putting students in work related situations. Agents could react to the behavior of students, for instance if a student was being careless in a task and was in a dangerous situation, Steve would show distress or fear. Also in a virtual environment the work of McQuiggan, Robison & Lester (McQuiggan et al., 2010) extends this line of research by investigating the affective transitions that occur throughout narrative-centered learning experiences. The analysis of affective state transitions in this work replicated the findings by DMello et al. (D´Mello et al., 2008) and Baker et al. (Rodrigo et al., 2009) where also engagement/flow dominated self-reported affect. For outdoor environments Shen, Wang & Shen (Shen et al., 2009) augment a pervasive e-learning platform with affective recognition. The results about emotion recognition from physiological signals achieved a best-case accuracy (86.3%) for four types of learning emotions.

Bosch, D'Mello & Mills (Bosch et al., 2013) analyzed the relationship between affective states and performance of novice programmers learning the basic Python. The results of their study indicated that the most common emotions students experienced were engaged (23%), confusion (22%), frustration (14%), and boredom (12%). It was useful to consider these results, as it presented evidence of what affective states need to be targeted in order to obtain less biased data. Similar to the previous work, Rodrigo et al., (Rodrigo et al., 2009) observed which affective states and behaviors relate to studentś achievement within a basic computer science course. The authors found that confusion, boredom and engagement in IDE-related (on-task) conversation are associated with lower achievement. Measuring mood is important, since it may have an impact on programmerś performance according to Khan, Hierons y Brinkman (Khan et al., 2007). It may be possible to detect moods on the basis of information regarding the programmerś use of the keyboard and mouse, and to integrate them into development environments that can improve programmer performance.

There has been very little research reported on the effectiveness of the use of keyboard and mouse dynamics as a sensory channel for affective recognition, and the few have not been focused on programming. The preliminary work by Zimmerman et al. (Zimmermann et al., 2003) describes a method to correlate users interactions (keyboard and mouse) with an emotional state, measuring different physiological parameters. The work of Vizer, Zhou & Sears (Vizer et al., 2009) also uses sensory data based on the time elapsed between each key press, the task given to users was to write a free-text and used linguistic features in order to recognize both physical and emotional stress. The results show a classification accuracy of 62.5% for physical stress and 75% from emotional stress; authors argue that these results are comparable with other approaches in affective computing.

They also stress that their methods must be validated further in other contexts. Moods may have an impact on programmerś performance according to Khan, Hierons y Brinkman (Khan et al., 2007). It may be possible to detect moods on the basis of information regarding the programmerś use of the keyboard and mouse, and to integrate them into development environments that can improve programmer performance. They briefly describe a further experiment that could use keyboard and mouse but only as future work. There are other studies about the behaviour of programmers, which are not directly concerned with affective states but are nevertheless important to their performance. Eye tracking in computing education is proposed in the work Busjahn et al., (Busjahn et al., 2014), and it is also used for assessing learners comprehension of C++ and Python programs in Turner et al., (Turner et al., 2014). Blikstein (Blikstein, 2011) proposed an automated technique to assess, analyse and visualize students learning computer programming. Blikstein employs different quantitative techniques to extract students behaviours and categorize them in terms of programming experience.

Although the use of KD is found in several research works as a biometric measure, its use for identifying affective states is rare in comparison. Epp, Lippold & Mandryk (Epp et al., 2011) effectively used KD in conjunction with decision-tree classifiers for the identification of 15 affective states. Although their work was based on fixed text, their technique to extract a feature vector was an inspiration for the proposed method in this work. As for free-text KD, Bixler y DḾello (Bixler and D´Mello, 2013) present a method for the identification of boredom and engagement based on several classification models. A similar situation exists in the case of Mouse Dynamics (MD), which is used mainly for authentication; however, Salmeron-Majadas, Santos and Boticario (Salmeron-Majadas et al., 2014) use both MD and KD to predict four affective states using five different classification algorithms. Bakhtiyari y Husain (Bakhtiyari and Husain, 2014) discuss a method based on fuzzy models for the recognition of emotions through KD, MD and touch-screen interactions. For a broad review of emotion recognition methods based on KD and MD, the work by Kolakowska (Kołakowska, 2013) is recommended.

3 PROPOSED METHOD

The goal of this work is to propose an affective recognition method based on the sensory data provided by the keyboard and mouse dynamics generated by a learner as he/she types a programming exercise. A brief explanation of the whole process is explained next, details come later. The process starts when the learner begins to type a program in a browser-based editor. As she types or moves the mouse the dynamics are recorded. When the learner submits the code to evaluation, the request includes the sensory data along with the code and information about the session. In the server, the code is evaluated in an external virtual machine that provides a sand box to prevent malicious or erroneous code to halt the server. When the result is ready, it is recorded along with the sensory data and sent to the preprocessing module. The output is a feature vector, ready for classification. A previously trained classifier is responsible for the classification, which outputs the predicted affective state. The method does not consider other sensory data, but it could be integrated with other sensory inputs in a multi-modal approach. Each of the steps is explained in detail next.

While a student is trying to solve an exercise, a script coded in JavaScript is running in the background, which captures every keystroke, mouse movement and mouse button press. Each capture of these events records a time-stamp in milliseconds (using the method getTime() of JavaScriptś built-in class Date) that describes when the event occurred. If the event is a keystroke, the script captures what key was specifically pressed, and what type of event occurred, it can be either a key- down or a key-up event. If it is an event related to a mouse button press, the key code of that button is recorded, as well as the type of event occurred again key-down or key-up. Finally, if the event was a mouse movement, the mouse coordinates inside of the web browser are recorded. The script monitors the mouse position every 100 milliseconds, and only if the position has changed, it records the new position. Each time a learner tries to evaluate a program, all the data generated is sent to the server along with the code. When the result of the execution is returned, all records are cleared and the process starts again. There is no problem if the user leaves for a long period of time, because no event will be triggered. If a user copies and then pastes the code from another source, this will be recorded. There is a limitation, only the browser-editor must be used; this could be a problem for more advanced programmers needing specialized editors. On the other hand a browser-based editor with the corresponding remote execution, does not require the installation of interpreters or compilers in learners computer. The code could even be written in a mobile device or any web-enabled device. Programming exercises are evaluated using unit tests; the results of the evaluation are shown to users. If all tests the program is considered to be correct. The source code for the Protoboard web based learning environment including the KD and MD functions are open source and available as Github repositories at https://github.com/amherag/keyboard-mouse-dynamics; the code for the sandbox is also available in https://github.com/mariosky/sandbox.

The raw data obtained from the script needs to be preprocessed to obtain a feature vector. Basically, this pre-processing consists in measuring the delays between key- down, key-up or mouse-move events triggered during an exercise. To calculate the average and standard deviations of these key presses, the delays between a key-down and a key-up event of the left button clicks are used. In addition to these averages and standard deviations of the delays between keystrokes and mouse button presses, the average and standard deviations of the number of total events contained in a digraph and a trigraph are calculated. These features are proposed and explained by Epp, Lippold y Mandryk (Epp et al., 2011). Most of the times, a digraph should contain four events, while a trigraph six. However, sometimes an individual can start a digraph or a trigraph before ending the previous one. These additional features represent these particular cases, and could be meaningful for the estimation of a learnerś affective states. Regarding the mouse movements, the average and standard deviation of the duration of each mouse movement, and the averages and standard deviations of the movements in the axes X and Y are also calculated. Lastly, a final feature is added to preprocessing of the data. The web tutorial recorded and included in the feature vector how many attempts a student required before successfully solving an exercise. The final feature vector consists of 39 features; these are based on the work of Epp, Lippold & Mandryk (Epp et al., 2011).

Once feature vectors are obtained from an experiment, the generated dataset is normally used as training data for a classifier. Researchers of affective recognition have used a wide variety of classification algorithms. As a proof of concept four well known classification algorithms where compared: k-Nearest Neighbors (k-NN) algorithm, a feed forward neural network trained with back- propagation, a naïve Bayes classifier and finally a decision trees algorithm for rational data (J-48) (Tan et al., 2006). Experiments and results will be presented next.

4 EXPERIMENT AND RESULTS

The aim of this research is to evaluate the use of keyboard and mouse dynamics as an appropriate sensory input for an affective recognition system in a learning environment for programmers, who interactively write short programs. An experimental approach was adopted with this aim: Sensory and quantitative data was collected from learners as they were enrolled in a basic course of Python programming. This data was then pre-processed using the method described earlier in Section 3, and together with the quantitative data obtained from users, classifiers were trained and

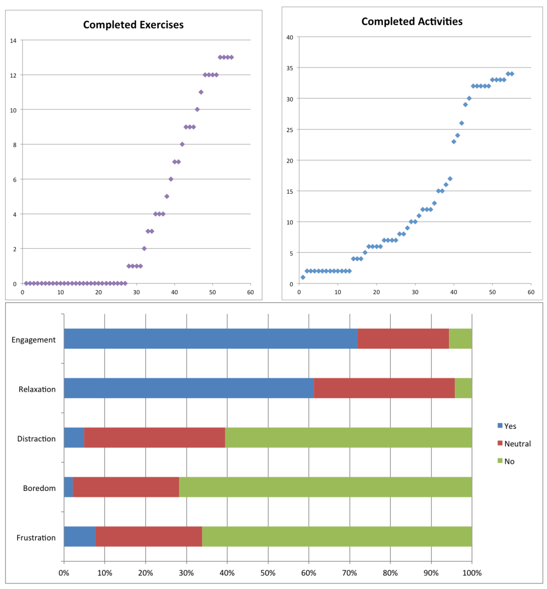

Figure 1: Completed activities and exercises by learner in ascending order (left) and distribution of emotions reported (right).

validated. The results were then compared and we arrived to some conclusions regarding the effectiveness of the method. The goal given to subjects was to solve as many exercises as they could in a period of two weeks. There was neither time limit nor a minimum amount of time required for a participant while trying to solve the exercises or complete the tutorial. The participants were able to stop and resume their interaction with the system at any time.

A tutorial was developed to obtain the necessary data using the proposed platform Protoboard, a web-based environment, whose latest version has been released under an open source license, and can be found online at https://github.com/mariosky/protoboard. Users log in to Protoboard by creating an account or by using their account credentials from the social network Facebook. The web tutorial begins with three introductory videos that explain the fundamentals of programming in Python, and how to use the editor to execute their code. What follows after these videos, are 13 programming exercises that students need to solve in sequential order. Exercises are of incremental difficulty. For this experiment Protoboard was configured to allow unlimited attempts to solve each exercise.

A total of 55 volunteers, with no previous experience in Python, where recruited as a response to three announcements in a special interest group of programming students in a social network, but the fact that they needed Facebook kept some prospects from volunteering; a subject reported that he normally creates temporary email accounts to try new web services; however, the use of Facebook gave researchers access to public information about the subjects. Their ages were in the range of 18 to 30 years. Although no experience in software programming was needed, as the web tutorialś course is of a very basic level, participants had different levels of experience, from 0 (8 persons) to more than 5 (22).

In order to determine what affective states a student was experiencing, the Experience Sampling Method (ESM) (Kubey et al., 1996) was used. After the students successfully solve a programming exercise, they were presented with an ESM survey that asks what they were feeling during their solving of the exercise. A very brief description is given about what to do in this survey, followed by statements the students need to answer according to how they were feeling. As an example, the statement I was feeling frustrated is presented, and a student needs to answer either Strongly agree, Agree, Neutral, Disagree, and Strongly Disagree.

After the two weeks of the experiment, only four learners completed all of the programming exercises and 22 did not completed any. Out of the total activities available (videos, survey and questionnaires) only two completed all. Figure 1 shows the number of exercises and activities completed by each learner.

The participantsínteraction generated a total of 142 feature vectors one for each successfully completed exercise. The affective states reported by learners after completing each exercise was grouped in three classes: Yes, Neutral and No. The class distribution of the emotions reported is shown in Figure 6. This results show that the most common emotions were flow/engagement (72%) and relaxation (61%) while few learners reported distraction (5%), frustration (8%) and boredom (2%). This distribution is different from what was reported by DḾello et al. (Bixler and D´Mello, 2013) and Rodrigo et al. (Rodrigo et al., 2009). Although Flow/Engagement was also the predominant class, the distribution is skewed to the first two. There are some possible reasons for this; first the majority of students had prior experience in programming, so learning new one is not that problematic, a second reason could be the freedom users had to abandon the activities, perhaps frustrated or bored learners simply quit the tutorial. A high number of learners (49%) did not completed any exercise and the once who completed more where also the more experienced.

As part of the data mining process a canonical genetic algorithm was used for feature selection with the following parameters: population size 700, 30 generations with tournament selection, tournament size was 25% of population size. A uniform crossover with 0.5 probability and mutation of 1/39, and a minimum feature size of 5. In the end, the subset for the frustration classifier includes 11 features, and the subset for the boredom classifier consists of 13 features.

The selected classifiers where implemented using Rapid Miner, with the following parameters: For J48 decision tree algorithm a confidence threshold for pruning of 0.25, with a minimum of 2 instances per leaf, and 3 folds for reduced error pruning. The feed-forward neural network used a back-propagation learning algorithm with two hidden layer of 20 neurons each, with a learning rate and momentum of 0.6, and was trained for 200 generations.

Table 1 shows the performance of each of the four classifiers together with the κ coefficients. A 10-Fold cross validation was used. The accuracies and κ coefficients obtained are close to what is usually obtained in fixed-text methods, for example, the results presented in Epp, Lippold y Mandryk (Epp et al., 2011). In a fixed-text method, subjects are asked to write specific texts. The results obtained with this method are satisfactory, considering that learners were writing a program, a task comparable with free-text writing, where there is no restriction on what has to be written. In the case of the κ coefficients, it is usual to see values below 0.2 in methods involving free-text. In this case, some values of κ were close to 0.5, is important to consider the values of κ in these results because the class distribution is not uniformly distributed. As it is observed in many works in affective recognition, decision trees normally produce competitive accuracy. In this case the J-48 classifier obtained an accuracy of 80.48%, which is marginally better than the others, and an accepted κ statistic for the kind of problem. The artificial neural network (ANN) gives the highest accuracy in the classification of frustration.

5 CONCLUSION

The aim of this research was to evaluate the use of keyboard and mouse dynamics as an appropriate sensory input for an affective recognition system. The context of use is a learning environment for programmers, in particular for learning assignments consisting in writing short programs interactively. An experimental approach was adopted with this objective, by effectively using the method with real subjects using the environment.

The proposed method in this work obtained satisfactory results. It is usual for a classification method based on free-text to obtain an accuracy and κ measure below their counterparts of fixed-text; however,

Table 1: Performance of every classifier with 10-fold cross validation. Accuracy (κ).

Affect	Naïve Bayes	J-48	k-NN	ANN
Flow/engaged	79.00% (0.50)	80.48% (0.51)	76.14% (0.43)	78.43% (0.45)
Relaxation	71.10% (0.40)	72.57% (0.45)	69.14% (0.37)	71.24% (0.34)
Distraction	70.33% (0.43)	71.00% (0.43)	74.00% (0.50)	72.52% (0.39)
Frustration	74.71% (0.49)	73.86% (0.45)	72.62% (0.42)	78.00% (0.50)
Boredom	74.71% (0.47)	84.57% (0.59)	83.81% (0.58)	76.19% (0.45)

in this work the results obtained were similar to those works applied to fixed-text dynamics. A possible explanation of this would be the addition of the mouse dynamics features and the additional preprocessing performed on the feature vectors. Out of all methods tested, a multilayer perceptron trained with backpropagation, the one tagged ANN in RapidMiner, gave the best classification results.

While these are promising results further experiments are needed. Other experiments should focus on novice programmers, but that is left as future work.

ACKNOWLEDGMENT

This work has been supported in part by: de Ministerio español de Economía y Competitividad under project TIN2014-56494-C4-3-P (UGR-EPHEMECH) and by CONACYT PEI Project No. 220590.

REFERENCES

Bakhtiyari, K. and Husain, H. (2014). Fuzzy model on human emotions recognition. *arXiv preprint arXiv:1407.1474*.

Bangert-Drowns, R. L. and Pyke, C. (2002). Teacher ratings of student engagement with educational software: An exploratory study. *Educational Technology Research and Development*, 50(2):23–37.

Bennedsen, J. and Caspersen, M. E. (2007). Failure rates in introductory programming. *ACM SIGCSE Bulletin*, 39(2):32–36.

Bixler, R. and D´Mello, S. (2013). Detecting boredom and engagement during writing with keystroke analysis, task appraisals, and stable traits. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, pages 225–234. ACM.

Blikstein, P. (2011). Using learning analytics to assess students' behavior in open-ended programming tasks. In *Proceedings of the 1st international conference on learning analytics and knowledge*, pages 110–116. ACM.

Bosch, N., D´Mello, S., and Mills, C. (2013). What emotions do novices experience during their first computer programming learning session? In *International Conference on Artificial Intelligence in Education*, pages 11–20. Springer.

Busjahn, T., Schulte, C., Sharif, B., Begel, A., Hansen, M., Bednarik, R., Orlov, P., Ihantola, P., Shchekotova, G., Antropova, M., et al. (2014). Eye tracking in computing education. In *Proceedings of the tenth annual conference on International computing education research*, pages 3–10. ACM.

Csikszentmihalyi, M. (1990). Flow: The psychology of optimal performance. *NY: Cambridge UniversityPress*.

Dijkstra, E. W. et al. (1989). On the cruelty of really teaching computing science. *Communications of the ACM*, 32(12):1398–1404.

Dillenbourg, P., Schneider, D., and Synteta, P. (2002). Virtual learning environments. In *3rd Hellenic Conference" Information & Communication Technologies in Education"*, pages 3–18. Kastaniotis Editions, Greece.

D´Mello, S., Jackson, T., Craig, S., Morgan, B., Chipman, P., White, H., Person, N., Kort, B., el Kaliouby, R., Picard, R., et al. (2008). Autotutor detects and responds to learners affective and cognitive states. In *Workshop on emotional and cognitive issues at the international conference on intelligent tutoring systems*, pages 306–308.

Elliott, C., Rickel, J., and Lester, J. (1999). Lifelike pedagogical agents and affective computing: An exploratory synthesis. In *Artificial intelligence today*, pages 195–211. Springer.

Epp, C., Lippold, M., and Mandryk, R. L. (2011). Identifying emotional states using keystroke dynamics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 715–724. ACM.

Jenkins, T. (2001). The motivation of students of programming. In *ACM SIGCSE Bulletin*, volume 33, pages 53–56. ACM.

Jenkins, T. (2002). On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, volume 4, pages 53–58. Citeseer.

Kapoor, A. and Picard, R. W. (2005). Multimodal affect recognition in learning environments. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 677–682. ACM.

Khan, I. A., Hierons, R. M., and Brinkman, W. P. (2007). Mood independent programming. In *Proceedings of the 14th European conference on Cognitive ergonomics: Invent! Explore!*, pages 269–272. ACM.

Kołakowska, A. (2013). A review of emotion recognition methods based on keystroke dynamics and mouse movements. In *2013 6th International Conference* *on Human System Interactions (HSI)*, pages 548–555. IEEE.

Kort, B., Reilly, R., and Picard, R. W. (2001). An affective model of interplay between emotions and learning: Reengineering educational pedagogy – building a learning companion. In *icalt*, volume 1, pages 43–47.

Kubey, R., Larson, R., and Csikszentmihalyi, M. (1996). Experience sampling method applications to communication research questions. *Journal of communication*, 46(2):99–120.

Lahtinen, E., Ala-Mutka, K., and Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. In *ACM SIGCSE Bulletin*, volume 37, pages 14–18. ACM.

McQuiggan, S. W., Robison, J. L., and Lester, J. C. (2010). Affective transitions in narrative-centered learning environments. *Educational Technology & Society*, 13(1):40–53.

Picard, R. W., Vyzas, E., and Healey, J. (2001). Toward machine emotional intelligence: Analysis of affective physiological state. *IEEE transactions on pattern analysis and machine intelligence*, 23(10):1175–1191.

Robins, A., Rountree, J., and Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer science education*, 13(2):137–172.

Rodrigo, M. M. T., Baker, R. S., Jadud, M. C., Amarra, A. C. M., Dy, T., Espejo-Lahoz, M. B. V., Lim, S. A. L., Pascua, S. A., Sugay, J. O., and Tabanao, E. S. (2009). Affective and behavioral predictors of novice programmer achievement. In *ACM SIGCSE Bulletin*, volume 41, pages 156–160. ACM.

Rossin, D., Ro, Y. K., Klein, B. D., and Guo, Y. M. (2009). The effects of flow on learning outcomes in an online information management course. *Journal of Information Systems Education*, 20(1):87.

Salmeron-Majadas, S., Santos, O. C., and Boticario, J. G. (2014). Exploring indicators from keyboard and mouse interactions to predict the user affective state. In *Educational Data Mining 2014*.

Shen, L., Wang, M., and Shen, R. (2009). Affective e-learning: Using" emotional" data to improve learning in pervasive learning environment. *Educational Technology & Society*, 12(2):176–189.

Sidney, K. D., Craig, S. D., Gholson, B., Franklin, S., Picard, R., and Graesser, A. C. (2005). Integrating affect sensors in an intelligent tutoring system. In *Affective Interactions: The Computer in the Affective Loop Workshop at*, pages 7–13.

Tan, P.-N. et al. (2006). *Introduction to data mining*. Pearson Education India.

Turner, R., Falcone, M., Sharif, B., and Lazar, A. (2014). An eye-tracking study assessing the comprehension of c++ and python source code. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 231–234. ACM.

Vizer, L. M., Zhou, L., and Sears, A. (2009). Automated stress detection using keystroke and linguistic features: An exploratory study. *International Journal of Human-Computer Studies*, 67(10):870–886.

Yang, S. J. (2006). Context aware ubiquitous learning environments for peer-to-peer collaborative learning. *Educational Technology & Society*, 9(1):188–201.

Zimmermann, P., Guttormsen, S., Danuser, B., and Gomez, P. (2003). Affective computing – a rationale for measuring mood with mouse and keyboard. *International journal of occupational safety and ergonomics*, 9(4):539–551.