Chen, W. et al. (Eds.) (2016). Proceedings of the 24th International Conference on Computers in Education.
India: Asia-Pacific Society for Computers in Education

Understanding and Analyzing Students Frustration Level During Programming

Prathmi KURTIKER[a*], Ramrao WAGH[a]
[a]*Department of Computer Science and Technology , Goa University, India*
*kurtikerprathmi@gmail.com

Abstract: We present our proposed and ongoing approach to detect the frustration among novice Java learners. We use the logs from the worldwide source code repository Blackbox that captures data from BlueJ IDE environment of our learners. We describe some factors which are the cause of frustration to novice programmer and conclude that it is possible to model an individual student level of frustration in laboratory settings using derived factors; and also possible to get an average frustration across all lab sessions. The real-time capturing, analysis and detection of frustration will result in timely intervention by instructors to ensure that learners regain their interest.

Keywords: Affective state, Frustration, Integrated Development Environment, Blackbox project, Intelligent Tutoring System

1. Introduction and Related Work

It is a common situation that during first programming courses, inability to perform the given programming exercises due to various reasons leads to frustration that causes students to disengage from the programming task and hamper further learning. If this state is not detected on time and attended to it may lead to more serious consequences. Our research focuses on detecting learners' affective state of frustration while learning to program in a controlled computer laboratory set up so that real time intervention by instructors is possible especially in a large class.

Identification of a learners affective state by Facial Expressions, Paralinguistic Feature (voice), Body Language and Posture, Physiology, Brain Imaging and EEG, Text, Multimodality are widely researched (Calvo & D'Mello, 2010). But these methods are not yet feasible in real-world lab setting due to the cost involved as well as the intrusion it will result in the laboratory set up. It has been shown that in a computer lab setting where learners program on any Integrated Development Environment (IDE), it is possible to detect frustration by log data (Rodrigo & Baker, 2009; Faw, 2015). Our approach is to use logged data generated while a student works on the lab exercises using BlueJ IDE. This can be adapted to other environments with suitable modification. The prior work on coarse grain detection of frustration (Rodrigo & baker, 2009) detect an average students' frustration across all five lab exercises using compilation logs, but failed to detect individual students' frustration between each lab session. In other similar work, (Faw, 2015) captured contextual and keystroke logs of students learning in Intelligent Tutoring System (ITS). This study presented and compared two models – model 1 that uses contextual data whereas model 2 that used both contextual and keystroke data and concluded that the model 2 gives better accuracy than model 1. (Altadmri & Brown, 2015) analyzed a novice programmer mistakes in large scale data from Blackbox repository of BlueJ users all over the world and found top 18 common and frequent mistakes made by programmer.

Our approach is to use the world-wide large-scale repository but use the data captured of only our students by providing identification markers so a individual frustration level can be computed using the captured data and demographic data known to us.

2. Method and Task Design

2.1 Environment and Task Description

Our ongoing study is conducted at the Department of Computer Science and Technology of Goa University on first degree Computer Science students learning Object Oriented Programming in Semester III. The class strength is 54. There are two courses related to Object Oriented Technology , one being a theory course discussing the OO Programming and modeling (using UML) while the lab course deals with Programming in Java and use of UML for group projects. Both the courses are taught by second author of this paper. The study is conducted during the lab hours when students engage in Programming using Java. The entire class is divided in two batches. Both the batches are of approximately equal capability as alternate students have been assigned to each batch. The recording of data is done for only one batch having 23 students (16 female & 7 male). This batch is asked to program using BlueJ IDE (www.bluej.org). The other batch is using Eclipse IDE (www.eclipse.org). The reason for capturing data for only one batch is that BlueJ IDE on every computer in the laboratory is configured with participant ID which can be associated with only one user. We will use other protocols such as observation protocols with respect to Eclipse batch to compare the performance of both the batches.

The BlueJ IDE captures and sends the contextual data and source code to Blackbox repository maintained by Blackbox team. So as the students complete their exercises in BlueJ, the Blackbox data repository project captures a live interaction data about each students programming task which is perpetual (Brown, Kölling, McCall, & Utting, 2014) and can be queried.

2.2 Procedure and Feature Extraction

As the conducted study is local to our students and in computer laboratory, the demographics information about each participant and the programming lab exercise they are working on is already known. In order to find individual student frustration across all lab sessions, our first step is to create student profile and a frustration profile for each BlueJ user. Blackbox captures data about each student in their own profile, so all the data about each individual will be captured in particular profile.

Our major challenge is to find only those factors that are intrinsic to Programming that lead to frustration. There may be other factors that lead to frustration among students. The second challenge is differentiating between confusion and frustration. It is noticed that, in the beginning of any lab session, students are confused but most of them are able to find their way and achieve the desired task without causing frustration but in some cases confusion persists and lead to frustration. We are interested in finding the point when confusion and other such factors turn into frustration so some corrective action can be taken including providing manual intervention by the instructor at that point.

Till date we have captured data of this batch for all sessions related to learning various basic OO features of Java programming. The more structured data capture will be undertaken in the later part of the course where simple design problems will be given to the students and they are expected to program the solution. We plan to also observe the sessions using systematic observation protocols so as to correlate the data captured with observed data.

3. Experimental Analysis

3.1 Feature Selection

In our model, Frustration is considered as cumulative in nature. At the beginning student is not frustrated or very less frustrated as the motivation towards goal is high. The individual students' frustration across all lab sessions given by $FS = FS_i + FS_{i+1} + FS_{i+2} + \ldots + FS_{i+n}$ where i (session) range from 1 to n. Each session will be evaluated to find a frustration based on the factors listed in **Table 1** and an aggregate score of frustration across each session for each individual will be found. The factors are explained below.

F1: (Error Quotient (EQ)) that ranges from 0 to 1 is introduced by (Tabanao, Rodrigo, & Jadud, 2008) and represents the students' struggling with syntax error. Given two compile event, it checks whether the error message, the line number and the source code is the same. Where 0 means no two successive compile events have same syntax error in program and 1 means every compile event result into same syntax error all time.

F2: Every invoked compile event results into success (true) or failure (false). If the number of compile events in a sequence that are false is more, this will cause frustration.

F3: If the self-perception of student is highly positive towards the goal of completion, he/she will create more edit event followed by compile event. If the errors still persist this leads to false state which may lead to frustration.

F4: Sometimes there is frequent invocation of compile event with/without edit event in a short time span which is followed by error message. This is sure sign of frustration.

F5: The time taken between two successful compile events in sequence is important factor, if time taken is large with respect to particular task and if the task is highly important rise in frustration will be high.

These factors can be computed from the data captured in Blackbox data repository.

Table 1: Factors that Cause Frustration During Programming.

Factors	**Cause**
F1	Error quotient (EQ)
F2	The number of false compile events in sequence
F3	The number of edit event follow compile event : error / false
F4	Frequent invocation of compile event with/without edit event in short time span
F5	Time between two consecutive successful compile event

4. Conclusion and Future Direction

The main purpose of this study is to find frustration among learners of a programming language such as Java. We have outlined some of the factor which cause and indicate frustration in novice Java learners in laboratory environment. We conclude that it is possible to model frustration by carefully drawing and combining features from Blackbox data repository project. In future we would derive and prove more factors which cause frustration in novice programmer and find individual student frustration in lab settings.

Acknowledgements

The authors would like to thank all the participants who agree to take part in the study. And also thank Blackbox project team University of Kent for their support to conduct our research.

References

Calvo, R. A., & D’Mello S. (2010). Affect Detection: An Interdisciplinary review of models, methods, and their applications. *IEEE Trans. Affect. Comput.,* 1:18-37.

Rodrigo, M. M. T., & Baker, R. S. J. d. (2009). Coarse-grained detection of student frustration in an introductory programming course. *In International workshop on Computing Education Research,* pages 75-80.

Fwa, H. L. (2015). Automatic detection of frustration of novice programmers through contextual and keystroke logs. *ICCSE conference.*

Brown, N. C. C., Kölling, M., McCall, D., & Utting, I. (2014). Blackbox: A large scale repository of novice programmers activity. *In Proceedings of the 45th ACM Technical Symposium on Computer Science Education, SIGCSE,* pages 223-228.

Altadmri, A. & Brown, N. C. C. (2015). 37 Million Compilations: Investigating Novice Programming Mistakes in Large-Scale Student Data. *In Proceedings of the 46th ACM Technical Symposium on Computer Science Education – SIGCSE,* pages 522-527.

Tabanao, E. S., Rodrigo, M. M. T., & Jadud, M. C. (2008). Identifying At-Risk Novice Java programmers Through the Analysis of Online protocols. *Philippine Computing Society Congress.*