VEAP: A Visualisation Engine and Analyzer for PreSS#

Natalie Culligan
Department of Computer Science,
Maynooth,
Co. Kildare,
Ireland
natalie.culligan@nuim.ie

Keith Quille
Department of Computer Science,
Maynooth,
Co Kildare,
Ireland
kquille@stconlethscc.ie

Susan Bergin
Department of Computer Science,
Maynooth,
Co. Kildare,
Ireland
sbergin@cs.nuim.ie

ABSTRACT

Computer science courses have been shown to have a low rate of student retention. There are many possible reasons for this, and our research group have had considerable success in pinpointing the factors that influence outcome when learning to program. The earlier we are able to make these predictions, the earlier a teacher can intervene and provide help to an at-risk student, before they fail and/or drop out. `PreSS (Predict Student Success)` is a semi-automated machine learning system developed between 2002 and 2006 that can predict the performance of students on an introductory programming module with 80% accuracy, after minimal programming exposure. Between 2013 and 2015, a fully automated web-based system was developed, known as `PreSS#`, that replicates the original system but provides: a streamlined user interface; an easy acquisition process; automatic modeling; and reporting. Currently, the reporting component of `PreSS#` outputs a value that indicates if the student is a "weak" or "strong" programmer, along with a measure of confidence in the prediction. This paper will discuss the development of `VEAP: a Visualisation Engine and Analyser for PreSS#`. This software provides a comprehensive data visualisation and user interface, that will allow teachers to view data gathered and processed about institutions, classes and individual students, and provides access to further user-defined analysis, to allow a teacher to view how an intervention could influence a student's predicted outcome.

CCS Concepts

Social and professional topics→ Professional Topics → Computing Education

Keywords

Computer science; Education; Data Visualization ; Educational Tools

1. INTRODUCTION

There is a significant student retention problem in third level computer science courses, and many institutions report a high failure and drop-out rate [13, 14]. Studies show that intervention and feedback for students who are either struggling with the material, or who find the material too easy and are in danger of disengaging, can be successful in reducing the drop-out rate if the students are identified as at-risk early in the module [9, 10]. Thus, projects to successfully identify the students that are in danger of dropping out, so that interventions can be made, are vital. There are many systems for predicting student success, but arguably, the most successful system is `PreSS` [2]: a semi-automatic machine learning system developed between 2002 and 2006 that predicts how well a student is likely to perform on an introductory programming module with more than 80% accuracy, after only 2-4 hours of exposure to basic programming concepts. Conversely, `PreSS#` [1] is a fully automated web-based system, built between 2013 and 2015, that accurately replicates the `PreSS` system. `PreSS#` provides an easy-to-use system that a student can log into and complete basic survey type questions, after minimal exposure to programming concepts. The system predicts if the student is likely to perform well and is a "strong programmer", or if they are likely to struggle and are a "weak programmer". The issue addressed in this paper is: how do we best deliver this information to the teachers in order to ensure that the data and functions provided by `PreSS#` are used to their full potential.

To this end we have developed `VEAP, the Visualisation Engine and Analyser for PreSS#. VEAP` is a teacher focused piece of software with three main functions:

- Generating visualisations of the data produced and gathered by `PreSS#` on the user's students
- Navigating and displaying the data of individual students
- Running further analysis on student results

This system allows teachers to identify students at risk of failure or disengagement and allows for interventions early on in a module, before the student has failed or disengaged. Further development of `VEAP` could provide educational institutions with the ability to help at-risk students, and to lower the drop-out rate in computer science courses.

2. Related Research

To date there appears to be no system that can predict student outcome with the accuracy of `PreSS#` after such minimal exposure to programming concepts. `VEAP` is the only visualisation engine and user interface created for use with the `PreSS#` system. Extensive research has been conducted on the factors that contribute to student success and predicting student outcome from these factors. Some of the larger studies are outlined below.

Bergin and Reilly [12] examined fifteen attributes that may influence student success in an introductory programming module. The goal of the study was to find if there were factors

Koli Calling 2016, November 24-27, 2016, Koli, Finland
© 2016 ACM. ISBN 978-1-4503-4770-9/16/11…$15.00
DOI: http://dx.doi.org/10.1145/2999541.2999553

that could be used to predict a student's likely performance, and to allow for informed and personalised interventions. A questionnaire and a custom-made cognitive test were used to gather data from students on: prior academic results; comfort level in the module; programming skill; self-perception; and some specific cognitive skills. They found the most important factors in success in a programming module to be the student's perception of their own understanding of the module. Other factors that were also found to be important were comfort level in the module, as well as high school maths and science scores, with some gender differences identified.

Doyle, *et al* [5] examined factors that influence student success in an introductory programming module, and found that: gender; programming language; and student attitude towards technology, did not affect student success, but that: mathematics exam scores; previous programming experience; and enrolling in an optional laboratory sessions, were all associated with student success.

Rountree, *et al* [6] investigated the factors that influenced student success using a survey collecting information about: age; gender; mathematical background; as well as expectations on the difficulty of the course. Like the `PreSS` study, it was found that mathematical ability and confidence to be important factors, but that a student's expectation of a high grade was the strongest indicator of success.

Byrne and Lyons [7] examined how: gender; prior computing experience; learning style; and academic performance, relate to results in a first year programming module. Like the `PreSS` study, they found a relationship between mathematics and final results, no significant link between gender and final results, and a possible link between learning style and student success.

Wilson and Shrock [8] investigated 12 factors and how they contributed to student success in an introductory computer science course. Their study concluded that comfort level in the class was the biggest indicator of success in the course and, like `PreSS`, found that mathematical ability was an important factor in predicting success, while computer game-playing had a negative effect on student success.

Self-esteem is a serious problem in computer science education. Eliasson *et. al* [5] found that students' confidence in their programming abilities seemed to significantly drop after completing a module, signifying that self-esteem is an important area of investigation for future study into computer science course retention.

2.1 PreSS and PreSS#

`VEAP` is an interface for `PreSS#` that is based on `PreSS` [2], a semi-automated prediction system. It is state-of-the art: no other system is able to predict introductory programming success with the level of accuracy that `PreSS#` has achieved, to the best of our knowledge. The underlying model has been continuously tested over the past ten years with multiple institutions and consistently achieves this high level of accuracy, irrespective of language or student cohort. The closest any other system has achieved is around 40%-50% [3]. `PreSS#`'s predictive performance is based on three student factors to determine the student's outcome. They are:

- Mathematical ability
- Time spent playing computer games
- Programming self-esteem

Mathematical ability is defined by the score received in the last mathematics exam taken by the student, which is converted into a numerical value between 1 and 12.

Hours spent playing computer games is defined as the number of hours a student spends per week during term time. The value for this is a number between 0 and 10, with 0 indicating 5 hours or less of game playing, and 10 indicating a very high amount of game playing, more than 45 hours a week.

Programming self-esteem measures how confident a student is in their programming ability. To calculate this value, `PreSS#` records the answer to 10 questions about the student's perception of their own programming ability. These questions are modified Rosenberg self-esteem questions [4] which have been validated for reliability and consistency [2]. These 10 answers are then reduced to a single value using Principal Component Analysis. The system uses a Bayesian learning algorithm on these three attributes to generate two values: [1] a prediction of success or failure, and [2] a measure of confidence in the prediction, between 0.1 and 1.0.

2.2 Issues with the current PreSS# system

While `PreSS#` has been very successful, it does not include a system for teachers to properly utilise the full potential of the data and functions provided by `PreSS#`. The system simply outputs a '1' or '0' to indicate if the student is a 'strong' or 'weak' programmer, along with a measure of confidence in the prediction. While these numbers are useful, there is currently no way for teachers to access this information easily and intuitively, nor is there any way to view the individual scores for each attribute. There is also no way for teachers to use the system to predict how well a student is likely to perform after a hypothetically successful intervention, even though the system has the potential to provide this function.

3. Description of the Tool

To address the issues with the current PreSS# system, `VEAP` provides a visualisation engine that allows for teachers to see, at a glance, which of their students are at risk of failure. The students can be arranged within the visualisation by two user selected parameters from the student's attributes, class, institution, gender, attributes, or outcome, thus allowing teachers to appropriately plan additional tutorials, interventions, or more advanced material.

As well as an overview of a class, the teacher also has the option to view an individual student's profile. Each student profile contains a paragraph about the student, describing their rating in each of the three attributes, and the student's prediction. The three attributes are represented by coloured bars, with an indication of how the student rated in each attribute, ranging from 0% to 100%. The teacher can manually alter these attributes, and re-run the Naive Bayes algorithm with the altered data to hypothetically test a student's predicted outcome after a successful intervention.

To fully utilise the `PreSS#` system and its rich data repository, `VEAP` was designed to include the following:

1. An overview of all the students in a class, presented in a digestible graphical form that uses a "traffic light" system to indicate which students are predicted to be strong (green), which are borderline (orange), and which are predicted to be weak (red), using the prediction and measure of confidence from `PreSS#` . This allows a teacher to see how many students are in danger of failing, at a glance.

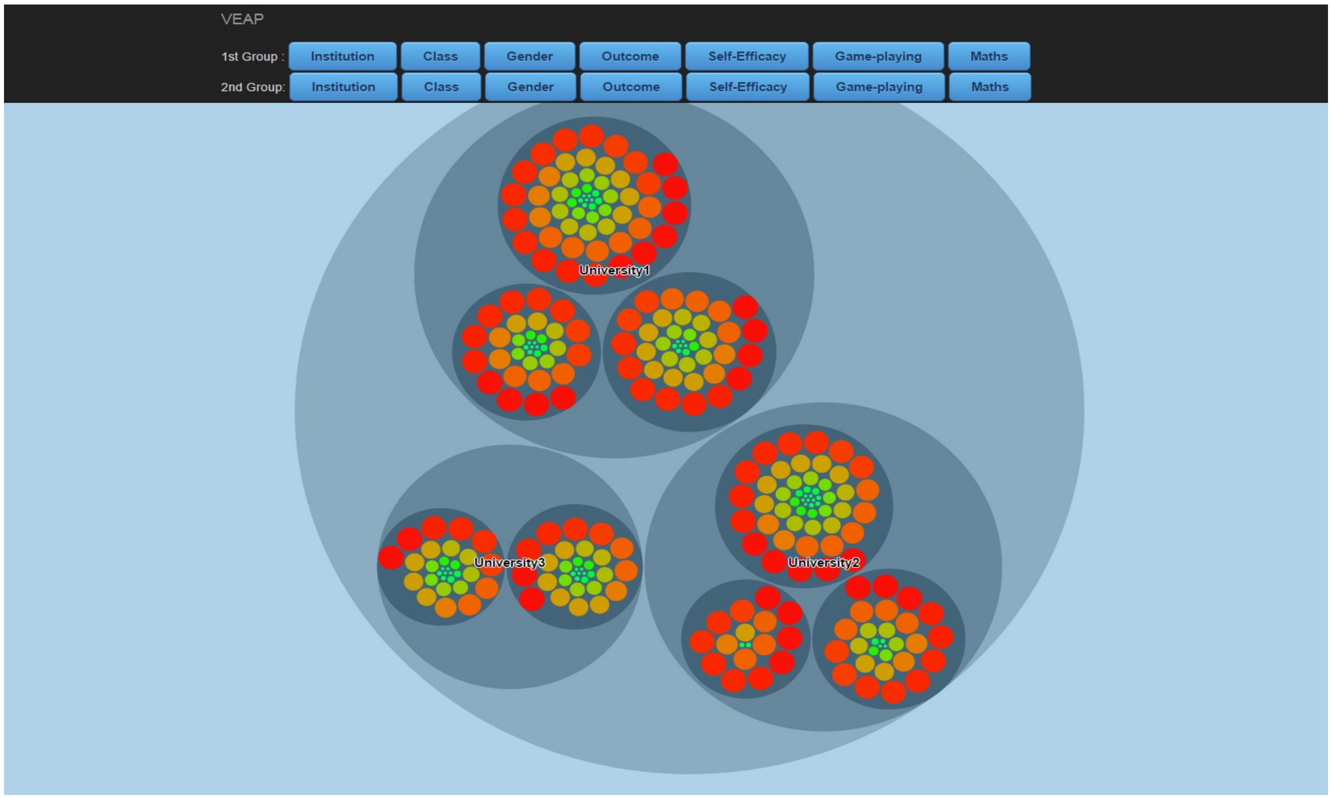
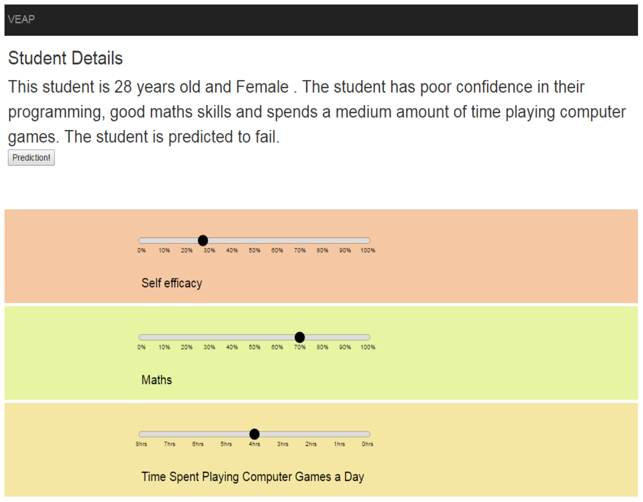


Fig 1: An example of a teacher's homepage

2. The ability to "zoom in" to selected areas of the data visualization. As introductory programming courses can often have a large number of students, and a teacher may teach several classes, so this feature is important for visualizations of very large data sets.
3. Customisation settings for the student overview, allowing the teacher to view the students arranged by one or two parameters, selecting from:
   - mathematical ability
   - time spent playing computer games
   - self-esteem
   - class
   - institution
   - gender
   - outcome
4. A student profile for every student that displays their information, including their: name; age; gender; student number; attribute ratings; and predicted outcome, as well as a visual representation of the student's attribute ratings that uses a bar chart and the "traffic light" system.
5. The ability to run further analysis on a student by manually modifying their attributes and re-calibrating by re-running the learning algorithm to indicate how an intervention may influence a student's prediction.

With these features, the teacher can arrange the students by their ratings for attributes, within a class or institution, allowing them to plan additional tutorials or more advanced materials for the appropriate groups of students. The teacher can also view an individual student's information, and use the re-calibration feature to decide how best to intervene by addressing the key attribute or attributes influencing the student's performance.



Student Details

This student is 28 years old and Female . The student has poor confidence in their programming, good maths skills and spends a medium amount of time playing computer games. The student is predicted to fail.

Fig 2 An example of a student's profile

3.1 Interface Design

The visualisation tool is designed to be a flexible and dynamic interface for the information and predictions made by `PreSS#` and for the re-calibration functionality.

Each student is represented as a circle, within one or two of the user-defined sub-groups. The students are colour coded and sized according to their predictions, so as to draw attention to the students in danger of failing. A teacher can click on any student circle to be brought to the student's profile page and view extra information about the student, and run extra analysis on their data.

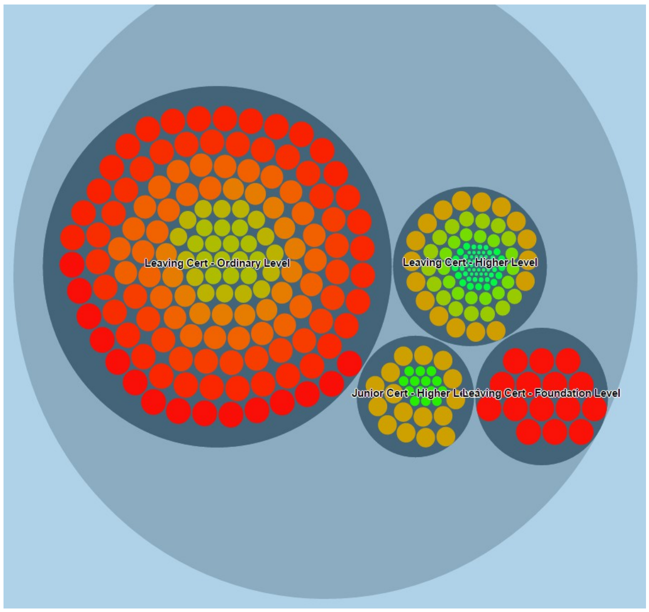


Fig 3 An example of a teachers students arranged by a single attribute: mathematical ability

On the student profile page, the teacher can view the student's attributes as slider bars, which change colour to represent the student's rating in that attribute, from red (poor), to orange (borderline), to green (good).

4. Re-calibration Tool

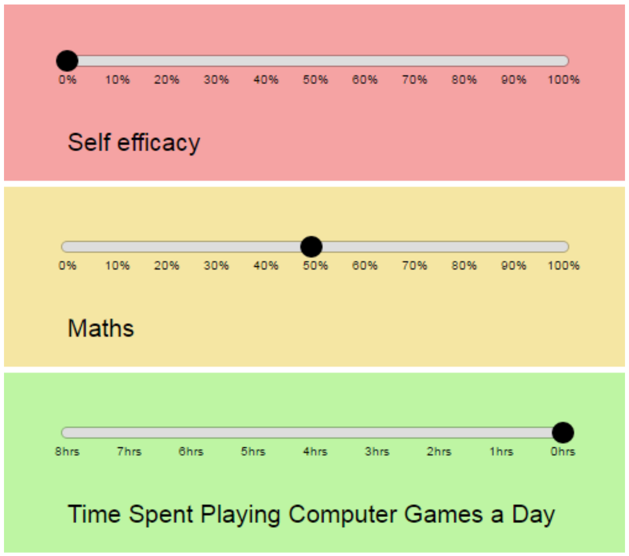


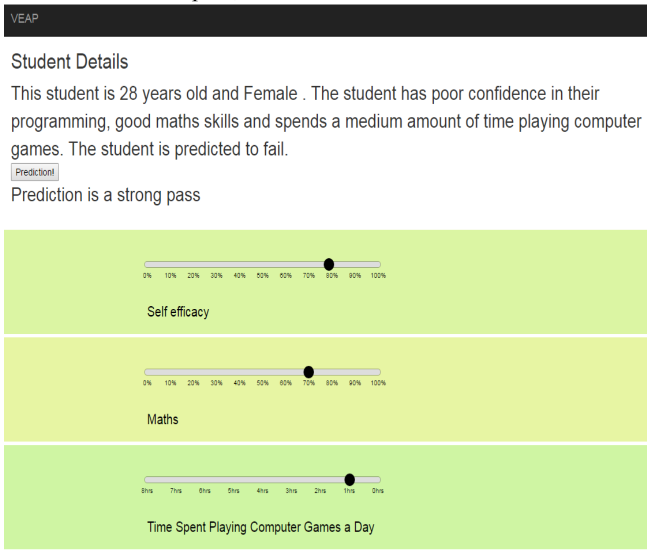
Fig 4 The slider bars showing the ratings of attributes for a student

It is a much simpler way of representing the three attributes, and is not intended to be as accurate and precise as the original `PreSS#` attribute values. Instead, it is a way to give teachers an idea of how a given student is likely to perform, and indicate what area the student is having issues with, to help the teacher to decide how best to plan an intervention for said student.

The re-calibration tool uses a simplified version of the `PreSS#` algorithm. The three attributes generated by `PreSS#` are represented by a slider, ranging from 0% to 100% for mathematics and self-esteem and from 8 hours to 0 hours (per day) for hours spent playing computer games.

Using these slider bars, the teacher can use the mouse to drag the sliders to indicate a higher or lower score for each attribute. This allows them to estimate the effect of a successful intervention on a student's prediction by raising the score for one of more ratings and clicking the "`Prediction!`" button.

Extra classes could be provided for students with a weak mathematical background, and students can be advised to cut down on their gaming time during term time, but low self-esteem is more complex, and is thus more difficult to tackle.



VEAP

Student Details

This student is 28 years old and Female . The student has poor confidence in their programming, good maths skills and spends a medium amount of time playing computer games. The student is predicted to fail.

Prediction!

Prediction is a strong pass

Fig 5 Demonstrating that the student from Fig 1.1 can raise their prediction from a fail to a strong pass by reducing their game playing and improving their self efficacy

4.1 Architecture

The visualisation engine has three main components:

- The database
- The controller
- The GUI

The system was built in C# in Visual Studio to facilitate interaction and integration with `PreSS#.` The system generates visualisations using the data from the connected database, but is not able to write to the database, ensuring the data gathered by `PreSS#` cannot be modified. All data created within the interface, such as modified attribute data, is stored as JavaScript variables or JSON data.

The controller is the computational engine behind the visualisation. It is written in C# and is responsible for: querying the database; constructing JSON files for the GUI to interpret; and running the Bayesian learning algorithm for the re-calibration function. The learning algorithm is run using the Accord.Net plugin for machine learning in .Net.

The GUI interprets teacher input and database queries to create an interactive visualisation and interface for the teacher to view the information and predictions gathered by `PreSS#`, using javascript plugin D3.

5. Discussion

The goal of `VEAP` is to enable teachers to provide a tailored educational experience for every student to help them succeed. By using `PreSS#` and delivering it to teachers, we aim to combat the worrying drop-out rates in computer science courses by pinpointing the at-risk students, and assessing how to best help these students.

The interface provides a way for a teacher to see, at a glance: how many students are in danger of failing; how many are weak at mathematics; students' programming self confidence; and time spent playing computer games. It can be used to decide what area students need help in, and teachers can use this information to plan extra tutorials or other interventions.

Interventions could consist of mathematics classes, to bring struggling students up to an adequate level of mathematical ability in comparison with other students, or simply pointing the student towards online maths courses such as KhanAcademy or Coursera, and providing them with a suggested time line for covering the necessary topics for bringing the students level of mathematics up to par. McDowell *et al* [6] suggests that pair programming increases student confidence and improves student retention. Similarly, a teacher could arrange for pair programming assignments, perhaps matching "weak" programmers with "strong" programmers.

For students who are in danger of failing due to a high amount of hours spent playing computer games, the intervention could take the form of a one-to-one talk, an email, or byat rids addressing the issue with the class.

The issue of self-esteem is a much more difficult problem. In `VEAP`, self-esteem is represented on a scale of 0-100%, but it is difficult to see how this relates to a student's confidence within a real-world setting. This is a non-trivial problem that requires research beyond the scope of this paper, and one that `VEAP` deals with quite crudely. Nonetheless, we believe that there is a real value in what `VEAP` provides: tangible evidence for a student that the only attribute preventing them from achieving a "strong" programmer prediction may be enough to boost that student's self-esteem.

6. Evaluation

Teachers with experience in `PreSS#` were asked to participate in an evaluation of the `VEAP` tool. They were given a short tutorial on how to use the tool, and were then asked to answer 4 questions:

- What did you like about the tool?
- What did you dislike about the tool?
- What would you change or add?
- Would you use the tool?

All users expressed that the ability to arrange the students by different attributes would be useful in planning interventions or extra classes. They all also stated that the re-calibration tool was something that would be useful when dealing with individual students who may be struggling. Some users felt that the tool could be more intuitive, or that it could display more information on screen. For example one user noted that the student profile page would state that a student had "good maths skills" or "poor maths skills", and wanted to know what the system considered "good" or "poor". The user suggested displaying information on the students last exam taken, as this information is in the system and may be helpful in understanding the students level of maths, and what kind of help they might benefit from.

66% of the users said they disliked the differing size of the circles that represented the individual students. They felt that, although it was helpful to see the students in danger of failing, it could lead to ignoring students who are doing well in the module, who may be in danger of disengaging. They suggested giving the user an option to invert the size of the circles, or to set them all to the same size. One user suggested that a student search function would be helpful. 100% of the participants confirmed that they would use the system.

7. Future work

`VEAP` provides an interface and visualisation and analysis engine for teachers, but it does not connect to the student outside of the initial prediction by `PreSS#`. Future work in this area could concentrate on the features requested in the evaluations, and on a student focused interface where a student could track their own progress, receive automatically generated advice on how to raise their prediction, and receive additional predictions at set points during the term.

8. References

[1] K. Quille, Susan Bergin, 2015. PreSS#, A Web-Based Educational System to Predict Programming Performance. *International Journal of Computer Science and Software Engineering*, vol. 4, no. 7,.

[2] S. Bergin, 2006. "A computational model to predict programming performance", *Ph.D, Maynooth University*,

[3] S. Bergin, 2015. Using Machine Learning Techniques to Predict Introductory Programming Performance. *International Journal of Computer Science and Software Engineering*, vol. 4, no. 12, pp. 323-328,

[4] Winch and M. Rosenberg, 1965. Society and the Adolescent Self-Image. *Social Forces*, vol. 44, no. 2, p. 255

[5] 2006. Investigating Students' Confidence In Programming And Problem Solving. *ASEE/IEEE Frontiers In Education Conference*. San Diego, CA: IEEE, Print.

[6] McDowell, Charlie *et al*. 2006. Pair Programming Improves Student Retention, Confidence, And Program Quality. *Communications of the ACM* 49.8 90-95. Web.

[7] P. Byrne and G. Lyons, 2001. The effect of student attributes on success in programming. *SIGCSE Bull* vol. 33, no. 3, pp. 49-52.

[8] B. Wilson and S. Shrock, 2001. Contributing to success in an introductory computer science course. *SIGCSE Bull.*, vol. 33, no. 1, pp. 184-188.

[9] van Schalkwyk, S. 2010. Early assessment: using a university-wide student support initiative to effect real change. *Teaching in Higher Education*, 15(3), pp.299-310.

[10] Tinto, V. 2003. *Student success and the building of involving education communities*. Higher Education Monograph Series, Syracuse University.

[11] Bergin, S. Mooney, A, Ghent, G and Quille K. Using Machine Learning Techniques to Predict Introductory Programming Performance. *International Journal of Computer Science and Software Engineering* 4, no. 12 (2015): pp323-328

[12] [Bergin, S. and Reilly, R. (2005). Programming:Factors that influence Success. *SIGCSE Bull.*, 37(1), p.411.

[13] M. Morgan, R. Flanagan, and T. Kellaghan,2001. A Study of Non-Completion in Undergraduate University Courses. *Dublin: Higher Education Authority*.

[14] J. Bennedsen and M. E. Caspersen, 2007. Failure rates in introductory programming. *SIGCSE Bull*, vol. 39, no. 2, pp. 32–36.