Continuous Assessment in Agile Learning using Visualizations and Clustering of Activity Data to Analyze Student Behavior

by

Suhas Xavier

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved May 2016 by the
Graduate Supervisory Committee:

Kevin Gary, Chair
Srividya Bansal
Sohum Sohoni

ARIZONA STATE UNIVERSITY

August 2016

ABSTRACT

Software engineering education today is a technologically advanced and rapidly evolving discipline. Being a discipline where students not only design but also build new technology, it is important that they receive a hands on learning experience in the form of project based courses. To maximize the learning benefit, students must conduct project-based learning activities in a consistent rhythm, or *cadence*. Project-based courses that are augmented with a system of frequent, formative feedback helps students constantly evaluate their progress and leads them away from a deadline driven approach to learning.

One aspect of this research is focused on evaluating the use of a tool that tracks student activity as a means of providing frequent, formative feedback. This thesis measures the impact of the tool on student compliance to the learning process. A personalized dashboard with quasi real time visual reports and notifications are provided to undergraduate and graduate software engineering students. The impact of these visual reports on compliance is measured using the log traces of dashboard activity and a survey instrument given multiple times during the course.

A second aspect of this research is the application of learning analytics to understand patterns of student compliance. This research employs unsupervised machine learning algorithms to identify unique patterns of student behavior observed in the context of a project-based course. Analyzing and labeling these unique patterns of behavior can help instructors understand typical student characteristics. Further, understanding these behavioral patterns can assist an instructor in making timely, targeted interventions. In this research, datasets comprising of student's daily activity and graded scores from an under graduate software engineering course is utilized for the purpose of identifying unique patterns of student behavior.

ACKNOWLEDGMENTS

I would like to specially acknowledge Dr. Kevin A. Gary for all the support and motivation that you have given me over the last two years. I sincerely admit that I wouldn't have been able to accomplish as much as I did if you weren't constantly pushing me to do better. I am absolutely grateful for having received an opportunity to work with you! Thank you!

I would also like to thank my committee members, Dr. Srividya Bansal and Dr. Sohum Sohoni for the invaluable inputs that I have received during this period. Dr. Sohoni, from the first time I mentioned my work, you have been onboard, reviewing my research and helping me whenever I needed. Thank you for this! Dr. Bansal, I understand that my choice of a defense date was not ideal considering you had other plans during the time. I am sorry and I really appreciate you taking the time to be there and for being so accommodating.

A few other names that certainly need to be mentioned here; Shweta Tripathy, it takes a huge deal of effort to put up with me especially in these last few weeks of my thesis work. For always being there, supporting and encouraging me despite the distance and time difference- Thank you! Rajanna (Nikhil Aravind), for helping me out initially during my research work, covering for me especially when I most needed it, thanks a lot! And my family off course, needless to say I couldn't have done any of this without them. Thank you all for your continuous support and love.

Finally, I would like to acknowledge State Farm for the grant that funded my research work over the last year and my participation in the FiE conference in October 2015. Thank you!

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

Software engineering education today is a technologically advanced and a rapidly evolving discipline. Being a discipline where students not only design but also build new technology, it is important that they develop a hands on or applied learning experience via a project centric course. Research has shown that such a hands on approach to learning produces more engaged learners compared to traditional, lecture based courses (Sproull, 2005). Project-based learning in this context refers to a pedagogical tool where students working in small teams may leverage an industry-practiced software process methodology to define, design, construct, and validate a quality software product. In a project based environment, students learn both technical competencies in the face of a complex scalable problem, but also contextual knowledge of how process mechanisms help manage that complexity.

In any learning environment, it is not only sufficient that a student learns new concepts but also that they learn it the "right way". It is important to augment a project centric course with a system of formative feedback (Crisp, 2007) so students can evaluate their learning consistently. This however, is in stark contrast to the inherent nature of student behavior. Students tend to be deadline driven with a majority of their work being completed only towards the project due date. This approach has an adverse impact as students that defer work to the last minute usually shortchange the concepts they are to discover and apply in favor of just "getting it done". Frequent, formative feedback (Trotter, 2006) has been shown to have positive impact on a student's learning behavior and can be used to provide regular "course corrections" to steer students away from this deadline driven model. A project centric course that incorporates frequent,

formative feedback creates a system of student learning, practice and continuous evaluation which makes for a well-rounded learning experience.

Project-based courses are typically designed in a predictive, forward engineered manner with a predefined process model where in the steps of the process are elaborated and arranged using tools such as task activity networks. Students in the process execute the scheduled tasks, and the output is obtained when the tasks are completed within the specified constraints. In contrast, agile learning methods (Schwaber & Beedle, 2001) utilize empirical process control theory which suggests that complex processes are not amenable to a predefined set of scheduled steps, as the environment is too fluid. Instead, the process should be driven by a feedback loop where small adjustments are constantly made based on continuous feedback. Popular agile software engineering process models are based on this principle of empirical process control. Agile is best characterized as a mindset, focused on learning and discovery. In contrast to traditional engineering process models, Agile methods embrace change by assuming that outcomes are not completely known in advance and these outcomes must be discovered through a learning process that iteratively refines the team's understanding through concepts of transparency, visibility, continuous feedback, and adjustments using empirical process control.

Irrespective of the nature (traditional or agile) of the project based course, it is imperative that the learning model is supported by a system of frequent, formative feedback ensuring that students constantly evaluate their learning process, making corrective changes and are continually aligned with the learning process. This can be achieved using continuous assessment. Continuous assessment in the field of educational research is defined as the practice of performing frequent assessment in the context of a course in order to ensure that student learning is continually aligned with

the learning expectations (Ghiatău, 2011). Recent studies (Lopez, 2007) have indicated to the positive impact of continuous assessment on improving the student learning experience. The goal of continuous assessment is to ensure that students are provided with frequent, formative feedback and are motivated to develop a rhythm of consistent working. Performing continuous assessment using a dashboard of visual reports holds tremendous promise as it provides a convenient and straight forward medium for students to self-assess and also contrast their progress against that of their peers. An objective of this research is to evaluate the use of visual reports as a means for providing continuous assessment (frequent, formative feedback) for students in a project-based course and to analyze the impact of these visual reports on student's compliance to the learning protocol (learning expectations). Compliance in this context is related to the clinical concept of "protocol compliance", where adherence to a protocol is essential to achieving the protocol's benefits. We employ quasi real time visual reports and notifications of student's activity on online tools via a personalized dashboard. This dashboard is provided to students in three different software engineering courses and the impact of these visual reports on student compliance is measured using surveys and logs of dashboard activity.

Analyzing the data gathered to power these quasi real-time visual reports can be beneficial as it provides insights about the daily activity patterns of student's contribution on these online tools. Learning analytics is an emerging field that is devoted to utilizing data science in the area of educational research. Applying machine learning algorithms on this dataset of daily activity can help cluster unique patterns of student behavior. As classroom sizes grow, it becomes increasingly difficult for instructors to keep track of individual student behavior in class. Providing instructors with these unique patterns of student behavior obtained by using learning analytics can help them

tailor the course to better suit the changing needs of students. Further, clustering various patterns of student behavior has the added benefit of helping instructors understand which cluster of students are in need of greater attention. This helps them make targeted interventions and the ability to perform well-timed intervention ties back to concept of continuous assessment where students receive timely feedback and are encouraged to develop a rhythm of consistent working and refrain from being deadline driven (Figure 1). This research is devoted to using machine learning algorithms on student activity datasets to identify unique patterns of student behavior in a project-based environment. We evaluate two different formats of datasets (daily activity data and student graded scores data) and identify unique patterns of behavior displayed by students in the context of the course. These patterns are labelled based on the observed behavior and a unique profile is generated for students in each of these clusters.

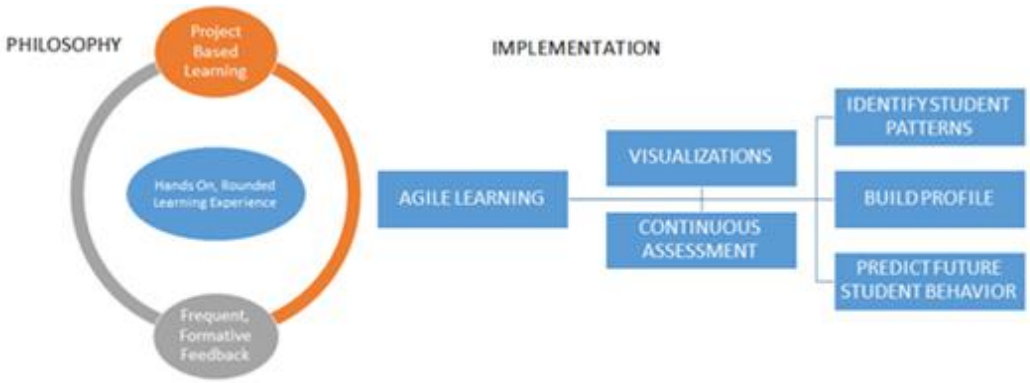

Figure 1: Visual Summary of this Work

In summary, this inter-disciplinary research work is based on the hypothesis that a project based course supported by a system of frequent, formative feedback, provides students in a software engineering discipline with a well-rounded learning experience. The goal of this research is to explore novel and innovative ways of providing students with frequent, formative feedback so they develop a rhythm of consistent working. We experiment by providing students with a personalized dashboard of visual reports and

notifications as a means of providing continuous assessment. Although promising, we do not find conclusive evidence that points to the impact of these visual reports on student's compliance to the learning protocol. Further, we employ machine learning algorithms on student activity data and successfully identify distinct patterns of student behavior observed in the context of the course. These patterns are labeled and subsequently a profile is created with the intention of helping instructors perform targeted and timely interventions.

The rest of this thesis work is divided as follows. Chapter 2 describes literature review. Chapter 3 sets the context for this research by discussing the need and methods involved in performing continuous assessment using visual reports and learning analytics. Chapter 4 describes the implementation steps for creating the dashboard and to analyze student activity datasets. Chapter 5 details the various experiments conducted using the visual reports and an analysis of the results. Chapter 6 discusses the experiments conducted to identify patterns of student behavior using machine learning algorithms and the subsequent results. Chapter 7 concludes with the lesson learnt and the scope for future work in this area

Chapter 2

LITERATURE REVIEW

This chapter discusses various literature that served as inspiration to the research. Since this thesis is focused on identifying novel ways to provide continuous assessment for students, the first section will discuss the various studies that shed light on the importance of providing timely and quantitative feedback. Subsequently, related work in the field of learning analytics and application of machine learning algorithms is discussed. Finally, various studies that prompted our exploration into the area of learning analytics; supervised learning algorithms applied to student datasets is discussed in order to define the context for this research work.

2.1 FREQUENT, FORMATIVE FEEDBACK AND CONTINUOUS ASSESSMENT

How important is it for students to be provided with feedback on their activity? (Crisp, 2007) used data from a Bachelor of Social Work class called "Introduction to Social Work", where students were required to submit two 1750 word essays during the course of the semester. The aim of the study is to observe if there was a change in score between the first and second essay based on the feedback provided on the first paper. The experiment was conducted on a class of 51 students over the course of the semester. Surprisingly, the author noted that there were no significant changes in the marks awarded to students between essays one and two. While the author attributes the small sample size as a possible reason for the unexpected results, she concludes that a greater focus is needed on the methodology employed to provide feedback and that it is important to provide students with useful feedback and is worth the extra effort involved. (Trotter, 2006) studied the use of tutorial files as a medium for continuous feedback. Tutorial files are files consisting of answers to tutorial and workshop questions (tutorial questions are short numerical or theoretical questions). Students are required

to turn in their tutorial files three times a semester and are graded and provided with feedback. The experiment is conducted on a business taxation module taken by around 45 students in the final year of a three year degree. Students are provided with a questionnaire regarding the effectiveness of tutorial files in improving the student learning environment. The results of the research are highly motivating since a majority of those interviewed admitted to having altered their behavior based on the feedback provided for the first tutorial file. Students who had not performed well were motivated to improve while those who had a good score were encouraged to continue the form. (Aljohani, 2013) stressed on the importance of immediacy in feedback provided for academic performance by combining learning analytics and formative assessment. They proposed the implementation of a system called "Quiz my Class Understanding" (QMCU), built as a cross platform mobile web application. The application is based on a quiz that is conducted post every class session on the content taught in the particular class. The results of the quiz is then immediately uploaded on the QMCU application in the form of a dashboard which also includes charts of how well the student has performed on the quizzes in comparison to the rest of the class. The experiment for this study was conducted in a course with a total of 72 students. The students were then provided with a questionnaire about the effectiveness and utilization of the dashboard. The results were positive. 84% of the students agreed that the dashboard was useful and 80% of the students felt that the ease of access to the dashboard helped them discuss their results with their peers.

All the three papers discussed above are highly insightful to this particular research as they emphasize the importance of feedback; timely feedback and also the utility of a dashboard to help students understand how they are performing in class in comparison to their peers. Although the Trotter study involved only 45 students, the

results were overwhelming since a majority of students admitted to having a positive change when provided with a feedback on their activity. Likewise over 80% of the students surveyed found a dashboard with visualizations of their performance useful. These results were highly motivating for this study as it emphasizes on the utility of a dashboard with real time visualizations and the importance of feedback in impacting positive change in student behavior and also compliance to the course expectations.

2.2 CLUSTERING STUDENT DATA USING MACHINE LEARNING ALGORITHMS

The next two sections describe the utility of unsupervised machine learning algorithms in order to cluster unique patterns of student behavior and analyze how supervised machine learning algorithms can be used in order predict behavior of students in a class. Machine learning algorithms are broadly classified as being either supervised or unsupervised in nature. Supervised learning algorithms refer to those class of problems that involve understanding a dataset to make predictions about the behavior or value of a target variable. Regression (linear and logistic), support vector machines, neural networks are examples of commonly used supervised learning algorithms. Unsupervised machine learning algorithms are those that are used to identify patterns or describe structure to a dataset that is generally unlabeled. Pattern identification and clustering (K-means, hierarchical) are examples of unsupervised machine learning algorithms.

(Oyelade 2010) discussed the application of a clustering technique on the academic result of students for one semester. The authors elaborate on the importance of mining student data for patterns of behavior instead of assigning pre-defined labels. The experiment used the K-means clustering algorithm on a dataset of 79 students. The clustering was conducted with clusters of sizes 3-5 in order to identify those clusters that had the most refined values. The experiment showed that students could be classified as

"Good", "Very Good" or "Fair". While the results aren't overwhelmingly surprising it is highly helpful since it indicates that there are patterns that can be mined from student data. Randomly classifying students as either good or bad would be a very poor reflection of this dataset. (Bindiya, 2011) utilized the K-means clustering and fuzzy c-means algorithms in order to cluster student behavior and identify patterns on the student dataset of 8000 records gathered over a period of five years. The dataset included features such as attendance, internal assessment scores, seminar assessment, class assignment assessment and university marks scored. The results of the experiment throw enormous light on the scope of clustering algorithms in revealing hidden patterns. The study showed that all final university scores were equal to or just below the first semester scores of students. This implies that the first semester scores can be considered as an indicator of what can be expected of a student in further semesters. The study concludes by adding that the results of the clustering experiment helped instructors take remedial actions. Since the first semester scores were found to be a direct reflection of the final university scores, extra effort was made to ensure students performed well during the first semester of the course. (Arora, 2013; Jamesmanoharan, 2013) both discuss a similar study based on the utilization of K-means clustering in order to identify patterns of student behavior in student test scores data. The clustering reveals unique classes of student behavior displayed in tests over the course of a semester.

The above papers shed light on the utility of unsupervised machine learning algorithms in its ability to identify unique patterns of student behavior that may not be necessarily evident. (Bindiya 2011) described how clustering student records reveals the significance of a student's first semester scores on the overall university score. In this research, since student activity data on agile tools is gathered every day, we have access to a large dataset that describes students' contribution through the course of the

semester. If machine learning clustering algorithms are applied to this dataset, it can help reveal unique patterns of student behavior throughout the entire semester. Analysis of these clusters can then aid an instructor in tailoring or modifying the course to better suit the rapidly changing needs of students.

2.3 MACHINE LEARNING ALGORITHMS TO PREDICT FUTURE BEHAVIOR

(Petkovic, 2014) gathered student team activity data and applied machine learning algorithms in order to predict which team had a higher probability of performing poorly so as to provide timely intervention. The experiment is conducted in a software engineering class involving 140 students split into teams of 5-6 members each, all working on the same project. The study only gathered data which was quantitative in nature such as counts of emails sent to team members, number of issues raised, events etc. Activity data is collected from logs of activity and also based on instructor's evaluation of each project. The Random Forest (Wiener, 2002) machine learning algorithm is used in order to classify a student project into predefined labels of A (at or above expectation) and F (needing attention). The results of the random forest algorithm was validates using the precision attribute which measured a value of 90% precision in classifying teams as F (needing attention).

(Cummings 2013) utilized process-level information data of student activity in a traditional classroom setup in order to use machine learning algorithms to address the usefulness of process-level information in machine learning prediction models and also to understand which machine learning algorithm is best fit to deal with this form of data. The study covered multiple machine learning algorithms for both classification (artificial neural networks, support vector machines, logistic regression) and regression tasks (linear regression and artificial neural networks). Regression was performed on the process level data to predict a numeric final grade. Classification is performed on the

same dataset to classify students into groups of A, B and C. The study included data from scores assigned to online quizzes, problem sets, tests and project activity (process level data). For regression, the linear regression model was picked while logistic regression was the choice for classification. The results of the study conclude that process-level information may be very useful to educators to make targeted intervention. It also concludes that simpler machine learning algorithms like linear regression and logistic regression are better suited for this form of data in comparison to other complex techniques.

(Ozaltin, 2015) uses a machine learning algorithm (Bayesian networks) in order to predict if an under graduate student project can be classified as innovative or not. The study collected data from 26 teams working on similar projects to design a biomedical device. Students were grouped into teams of 3-5 and were surveyed twice every week to understand what the student had worked on. The student can pick an answer from one of 89 activities listed. These 89 activities are classified into eight different categories and based on the design activities that a team engages in, and in particular their usage levels, the model allows instructors to predict as well as positively impact the innovativeness of the team's final product. Although the experiment was conducted on a relatively small dataset (26 teams), the results are highly insightful because if a team is doing "poorly" in terms of product innovation the instructor can advise the team on how to redirect its efforts (timely intervention).

(Kim, 2013) utilizes support vector machines to classify if a student is being attentive or not based on the data received from sensors. Although the area of our study is not focused on attentiveness, the ability to predict a student's behavior based on available data is highly significant. (Fang, 2013) gathered scores of student performance on a prerequisite course and applied four different prediction techniques in order to

predict the performance of students on a current course. The study used multiple regression, support vector machines and two flavors of neural networks (multilayer perception and radial basis function). The conclusions, although surprising, stated that the choice of mathematical model had a slight effect on the average prediction accuracy and the percentage of accurate predictions and concluded that support vector machines had the better accuracy of all the other models.

The results and conclusions of the above discussed papers describe the efficacy of using supervised machine learning algorithms in order to make timely predictions and targeted interventions. The data gathered from student activity on agile tools can be used to predict future student behavior. However, unlike the papers above, this research believes that labels for future classification should not be predefined but instead must be derived from an analysis of the dataset available to us. Attempting to classify a student as being “good” or “average” does not paint the “right picture” of the students’ behavior in an agile context. Detailed labels such as “start slow ends well”, “more code, less scrum” would better describe students activity and thus help instructors understand typical student behavior in the context of the course. However, the methodology utilized in the above papers is of great significance in understanding the future scope and possibilities of this research project.

In summary, this chapter reviews literature that signifies the importance of frequent, formative feedback on the student learning experience. Visualizations of student progress and peer comparison have been shown to have greater acceptance. The finding of these literature hold special significance as this research is dedicated to identifying innovative ways of providing students with frequent, formative feedback in a project based course. Visualizations and notifications are provided via a personalized dashboard. Using unsupervised learning algorithms to cluster unique patterns of student

behavior helps instructors understand typical characteristics displayed in class prompting them to modify/ tailor the course to suit these changing needs and also pay close attention to those students who may be in most need of it. This approach is a manifestation of continuous assessment model where students receive timely feedback encouraging them to alter their learning process. Further, if a label can be assigned based on the observed behavior, a profile can be created using which supervised learning algorithms can be utilized to predict what label best describes a student in the future instance of the course. This again allows an instructor to make timely interventions to ensure a better learning experience for the student.

The next chapter sets the context for this research. It outlines the research questions that drive this thesis work and describes a project-based environment that is conducive for performing continuous assessment using visual reports and learning analytics. It highlights the motivation for using visual reports as a means for providing continuous assessment and how student activity data can be clustered to identify unique patterns of student behavior in the context of a course.

Chapter 3

RESEARCH CONTEXT

In the software engineering discipline, project-based courses offer students an opportunity to develop a hands on, applied learning experience. It is important that students in these project based courses are provided with frequent, formative feedback (Crisp, 2007) to ensure that they continually evaluate their learning process and are aligned with the expectations of the course. However, this is in contrast to the inherent nature of students. Students are generally deadline driven with a majority of their work being completed only towards the project due date. This deadline driven behavior can have a negative impact as students who tend to complete their work at the last moment miss out on the valuable learning experience in favor of simply completing the task. One of the biggest challenges for educators is motivating students to forego this deadline driven behavior in exchange for developing a rhythm or cadence of consistent working. Providing frequent, formative feedback (Trotter, 2008) allows students to make regular adjustments in the learning process thereby ensuring that they are continually aligned with the expectations of the course. Continuous assessment in educational research is defined as the process of performing frequent assessment. Continuous assessment in this context can be used as a mechanism for providing students in a project-based course with frequent, formative feedback. The focus of this research then is to identify novel ways of performing continuous assessment for students in a project-based course so they develop a habit of consistent working and not succumb to a deadline driven model.

We explore the use of quasi real time visual reports and notifications as a means of providing continuous assessment. The premise for this method, is that, continuous assessment, where the student always has visibility into her/his activity, will motivate students to adhere to the “consistent work rhythm” expectation, thereby increasing the

student's ability to learn in context. We measure the impact of these visual reports on student's compliance to the expectations of the course (learning protocol). Compliance in this context is related to the clinical concept of protocol compliance, where adherence to a protocol is essential to achieving the protocol's benefits.

In a small classroom setting, an instructor often develops an understanding of the distinct learning behavior displayed by students in the class. This understanding aids an instructor to direct their attention to students who may be in most need of it. However, as classroom sizes grow it becomes increasingly difficult for instructors to identify these distinct patterns of learning behavior. Machine learning algorithms can be used to cluster students based on their learning behavior and provide guidance on strategies for improvement. An instructor's understanding of these behavioral patterns helps them understand typical student characteristics allowing them to adjust their teaching process to better suit the needs of these students. Additionally, understanding clusters of learning patterns in a course helps instructors provide timely interventions for those clusters of students who may be at risk of not complying with the learning protocol. In this research, we explore the use of unsupervised machine learning algorithms to cluster distinct patterns of student behavior in a project based course. Each of these unique patterns are assigned a label based on the behavior observed and subsequently a profile is created of student activity for each of these clusters.

It is important to note that the use of visualizations and machine learning algorithms to provide continuous assessment for students is applicable to any project based course. However, in this research, we discuss the methods of performing continuous assessment in an agile learning context. The rest of this chapter is divided as follows. We first explore the research questions that define the goals and scope of this research. Subsequently, we discuss the agile learning approach to set the context for a

project-based course. We then detail the methods used to perform continuous assessment using visual reports and machine learning algorithms in an agile learning environment. Finally, we talk about the Software Enterprise; an agile learning pedagogy and how it serves as an ideal test bed to experiment with continuous assessment methods deliberated in this research.

3.1 RESEARCH QUESTIONS

This research is focused on identifying various methods of providing continuous assessment in a project based course and the subsequent impact of these methods on student's compliance to the learning protocol.

RQ 1. Does a personalized dashboard with visual reports of learning activity, peer comparison and real time feedback have an impact on adherence/compliance to the learning protocol?

RQ 2. Can historical activity data be used to cluster students based on behavior? Can these clusters be assigned a label to be used in subsequent models to make predictions of future student behavior?

3.2 AGILE LEARNING

Project based courses in the software engineering discipline are typically designed in a forward engineered manner with a sequence of predefined tasks and milestones. Students in this environment execute these predefined tasks and the process is completed when the milestone is achieved within the expected constraints. In contrast, agile methods based on the empirical process control theory (Schwaber & Beedle, 2001), suggests that complex processes are not amenable to a predefined set of scheduled steps, as the environment is too fluid. Instead, the process should be driven by a feedback loop where small adjustments are constantly made based on continuous feedback. Popular agile software engineering process models are based on this principle of empirical

process control. Agile methods are the fastest rising software lifecycle process methods in software engineering today owing to its iterative nature, feedback loop and high tolerance to change (Agile Manifesto, 2001). While engineers adopt agile methods due its iterative nature and feedback loop, educators stress on the importance of providing frequent formative feedback to improve the learning experience. Agile learning refers to a learning model that combines these two features by implementing a learning cycle that is highly iterative and is driven by a loop of continuous feedback. In an agile learning environment, students develop a rhythm or cadence of consistent working where continuous feedback prompts constant modifications or "tweaks" in the learning process leading them away from a deadline driven model. The key feature of agile learning is the focus on building and maintaining a sense of rhythm in the learning process. However, getting students to sustain this rhythm can be an enormous challenge as students generally are deadline driven. It is thus imperative that we explore various modes of providing frequent, formative feedback; continuous assessment.

3.3 CONTINUOUS ASSESSMENT

Continuous assessment as mentioned in chapter 2 is the practice of performing frequent assessments in a course context. This frequent assessment provides a feedback mechanism ensuring students are properly aligned with the learning process. Continuous assessment holds particular promise for project-based learning as it is difficult to provide consistent and timely feedback when employing this pedagogy (e.g. student progress may be obscured in a team setting; students may not make timely progress toward project milestones). In continuous assessment, a continuous stream of learning activity data is collected for the purposes of providing continuous formative feedback and analyzing compliance to a learning protocol. Continuous assessment interpreted from an agile software engineering perspective is closely related to the

practice of continuous integration and testing (Fowler, 2006). This practice is fundamental to agile software engineering in that it supports transparency, visibility, continuous feedback, and adjustments based on the empirical process control. We borrow from the concept of continuous integration and testing (CI & Test) in agile software engineering. CI & Test promotes continual integration and verification of software by performing a software build and regression test for each commit to a software repository, or at least on a nightly basis. This practice avoids the problem of infrequent integration testing and long build processes. CI & Test dashboards (Figure 2) in tools such as TravisCI (travisci.org), Cdash (cdash.org), or Jenkins (jenkins-ci.org) provide visibility of product quality status. Used in conjunction with a scrum board and a mature source code control platform like GitHub (github.com), the project team and all stakeholders have full visibility into the rate of product development, project activity toward requirements, and product quality.

S	W	Name ↓	Last Success	Last Failure	Last Duration
		Project F cubicCoins	20 hr - #106	27 days - #71	9.2 sec
		Project I Entrepreneur Sim	26 days - #30	26 days - #29	10 sec
		Project J Sports	8 hr 7 min - #98	15 days - #81	1 min 7 sec
		Project Team C Gamers	21 hr - #93	17 days - #72	48 sec
		Project Team D Gamers	1 hr 10 min - #68	N/A	1.2 sec
		Project X thefailedandthefurious	5 days 20 hr - #113	20 hr - #123	39 sec

Figure 2: Jenkins Dashboard with Traffic Lights and Weather Metaphors

3.4 CONTINUOUS ASSESSMNENT USING VISUAL REPORTS

A motivation to use visual reports to perform continuous assessment is drawn from CI & Test dashboards (Figure 2) that use traffic light and weather metaphors to provide short-term and long-term feedback mechanism. Visualizations of student

activity provided via a dashboard provides for easy access and encourages students to discuss their progress with their peers (Aljohani, 2013). In this research we use an agile learning course to performing continuous assessment by providing student with visual reports of their activity on online agile tools. Agile learning courses use Scrum (Schwaber & Beedle, 2001) to conduct their projects since it embodies a philosophy of establishing a rhythm to weekly and even daily activities. One of the appealing features of Scrum is not only its emphasis on implementing iterative development using sprints but also the significance it lays on team communication. In Scrum, teams conduct daily, short (five minute long) meetings to discuss progress and hurdles to success. Agile learning courses that utilize Scrum expect students to develop a rhythm of consistent working by making regular updates on their scrumboards (e.g. Scrumwise, Taiga), contribute code on a source code control tool (e.g. GitHub) and conduct frequent stand up meetings. One goal of this research project is to create a dashboard similar to CI & Test platforms (Figure 2) showing the continuous assessment view of a student's activity on these online agile tools.

We create a web-based dashboard (CAssess) that retrieves student contribution data from online agile tools, performs basic statistical analysis and displays visualizations in the form of charts and graphs. The dashboard also provides dynamic notifications or "call to action" based on a student's activity over a time period. The goal of CAssess is to leverage the transparency of the dashboard with the use of notifications to provide timely formative feedback, so students have the ability to immediately correct behaviors. A sample visual report showing student's contribution on the source code control tool (GitHub) is described in Figure 3. To understand whether students are utilizing the tool and are compliant in their learning process, a logging mechanism is implemented in CAssess. This helps us understand how frequently students visit the

dashboard, duration of their visit, tabs visited and notifications viewed. Real time analysis of this log data provides instructors with a temporal view of dashboard utilization by students in class. The aim of performing continuous assessment using visual reports is to ensure that these visual reports motivate students to develop a rhythm of consistent working and adhere to the learning expectations of the class. The impact of these visual reports on student compliance to the learning protocol is analyzed by conducting end of semester surveys and also using the log of dashboard activity. If progress is observed in a student's behavior that is proportional to their visits to the dashboard, then a causality can be established linking the dashboard to the improvement observed in student behavior. The experiments using CAssess and the subsequent results are discussed in chapter 5.

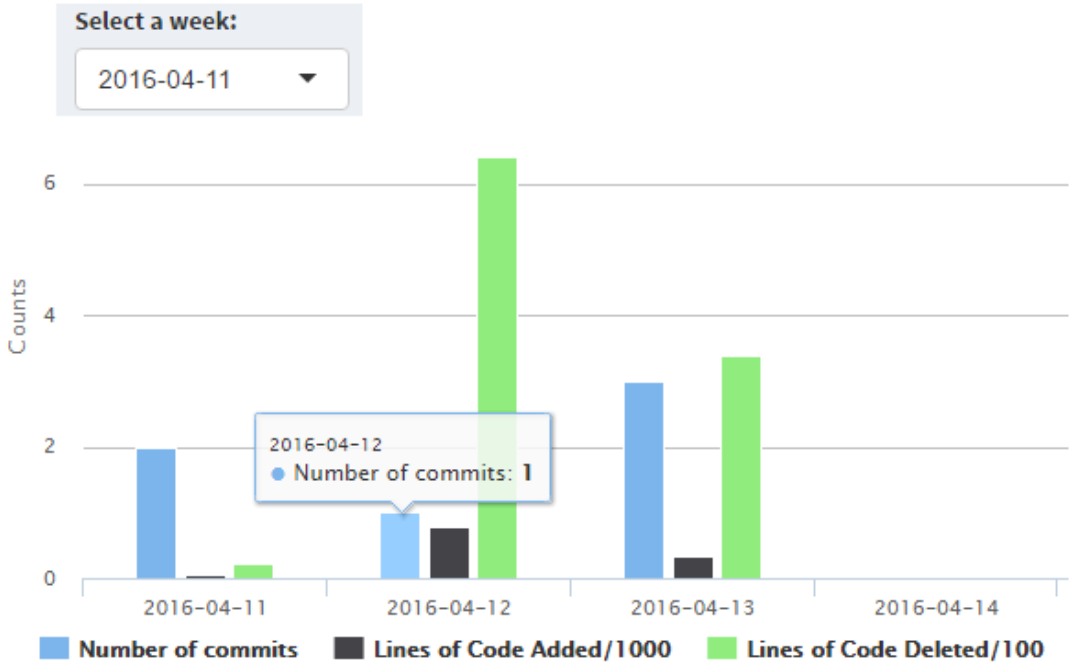

Figure 3: Visual Report Showing Daily Student Activity on Source Code Control Tool (GitHub)

3.5 CONTINUOUS ASSESSMENT USING LEARNING ANALYTICS

Learning analytics is an emerging field that is devoted to collecting student data to analyze behavioral patterns in the context of a course. Learning analytics holds

particular promise in project based courses as identification and analysis of student behavior can help instructors tailor their courses to suit the changing needs and also perform timely interventions to ensure students are “steered in the right direction”. In this research, we integrate principles of software engineering and learning analytics to support continuous assessment in an agile learning environment. Agile methods connect to learning analytics through empirical process control; a data-driven, feedback-oriented approach to process enactment. We utilize machine learning algorithms to cluster distinct patterns of student behavior in the course. Through this research work, we seek to incorporate these patterns of behavior into continuous formative feedback to cause behavior change.

In a project centric course based on agile learning, students use online agile tools (scrumboards, source code control, CI & Test, stand up meetings) to conduct their projects. Gathering data of student activity data on all these tools and subjecting the resultant dataset to unsupervised machine learning algorithms can help understand the nature of student’s contribution. Clustering algorithms such as K-means clustering can be used to group unique patterns of student activity. Once unique clusters are generated, each of these clusters can be assigned a label based on the behavior observed. This method of labelling (“Consistently Compliant”, “Code Centric”) based on the behavior observed in the cluster helps understand student activity patterns better in contrast to creating predefined labels such as “good”, “poor” or “average”. Subsequently, a profile can be created for each of these labels. Creating profiles has dual advantages. First, it serves to explain student activity in each of these clusters. For example, a profile created for the those students who were clustered as being “Code Centric” describes how all students in the particular cluster contribute on each of the agile tools on a daily basis. Second these profiles act as ideal training datasets to apply supervised machine learning

algorithms. Supervised algorithms refer to those class of problems that involve understanding a dataset to make predictions about the behavior or value of a target variable. Supervised learning algorithms applied to this training set can help predict what label/ cluster describes a student in the future instance of the class. This process of predicting future behavior is beneficial as it helps instructors make targeted interventions and ties back to the concept of performing continuous assessment in order to ensure that students develop a rhythm of constant working and not succumb to a deadline driven model.

3.6 THE SOFTWARE ENTERPRISE

This thesis is devoted to exploring novel and innovative ways of performing continuous assessment for students in a project-based environment. To this end we evaluate the utility of a personalized dashboard containing visual reports and notifications of student activity on online agile tools. Further, we use learning analytics on student activity datasets to identify distinct patterns of student behavior with the expectation that early identification of these behaviors can help instructors tailor the course to suit the needs of students and also perform targeted and timely interventions. Although these methods can be applied to any project-based course, in this research we use an agile learning pedagogy, the Software Enterprise (Gary, 2007), as a testbed to experiment with visual reports and learning analytics as a means to perform continuous assessment. The Software Enterprise is based on the Kolb's learning cycle (Kolb, 1998). Kolb's learning cycle describes the significance of a continuous loop of knowledge dissemination, knowledge comprehension and application to enhance the learning experience. A project centric course based on Kolb's learning cycle can help a student learn (knowledge absorption) in a traditional classroom setting (face to face or online), comprehend what's being taught and practice the same within the framework of an

applied project. A visual representation of the Software Enterprise pedagogy based on Kolb's learning cycle is as shown in Figure 4.

The Software Enterprise forms a project spine (Gary, 2013) wherein software engineering concepts are broken into discrete modules and sequenced over the course of a semester to synch up with project activities. In this just-in-time approach, students are exposed to a concept, practice it, apply it on a scalable project, and then evaluate the applied technique. In this research I specifically use two courses both of which conduct their projects using Scrum. Scrum is an excellent process model for the Enterprise as it embodies a philosophy of establishing a rhythm to weekly and even daily activities. Students are assessed after each sprint (4 times per semester) on their process-related activities (updating a scrum board and participating in standup meetings), code-related activity (frequency and significance of commits to the source code repository), integration of modules into the project, and team process activities (story development, sprint reviews and retrospectives). A key principle of the Software Enterprise is the focus on developing a rhythm or cadence of consistent working and students are discouraged from making last minute or even excessive contributions. The Software Enterprise provides a good vehicle for continuous assessment due to its highly modularized curricular organization, emphasis on highly iterative software development and expectation of continuous, consistent work on the project.

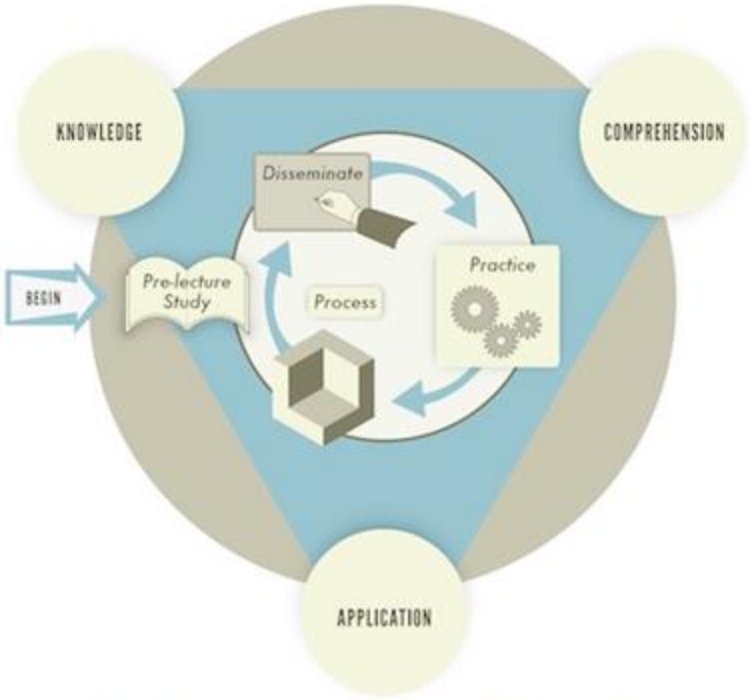

Figure 4: Software Enterprise Based on Kolb's Learning Cycle

In summary, this research explores different means of providing continuous assessment to students in a project-based course so as to encourage them to develop a habit of consistent working and refrain from a deadline driven approach. We utilize visual reports and notifications and measure the impact of these reports on student compliance to the learning protocol. Further, we use machine learning algorithms to identify distinct patterns of student behavior in a project-based course in order to assist an instructor in tailoring the course to suit the needs of the students and also perform timely and targeted interventions. We use the Software Enterprise pedagogy as a medium to conduct our experiments and validate the results.

In the next chapter, we explore the steps involved in implementing a dashboard that provides continuous assessment to students in a project-based course using visual reports. The dashboard with notifications are provided to students in three different software engineering courses and their activity on the dashboard is monitored. Further,

we gather a dataset comprising of grades assigned to students across all sprints of a project-based course to discover distinct patterns of student behavior. Finally, a dataset comprising of a student's daily contribution on online agile tools is subjected to an unsupervised machine learning algorithm to identify similar patterns of behavior.

CHAPTER 4

Continuous Assessment Platform

This thesis is based on the premise that a project-based course supplemented by a system of frequent, formative feedback provides students with well-rounded, hands-on learning experience. To that end, we first explore the use of a dashboard that provides visual reports and notifications of student activity and measure the impact on protocol compliance using logs of dashboard activity and surveys. Subsequently, we explore the use of learning analytics in helping instructors understand typical student activity characteristics in class so as to provide timely and targeted interventions. This chapter describes the implementation steps involved in designing and building a dashboard with visual reports and notifications. Further, it discusses the implementation of learning analytics on student activity datasets using unsupervised machine learning algorithms to identify unique patterns of student behavior in the context of a course.

Although, continuous assessment can be performed in any project-based course, in this research we use a graduate (SER515) and an under graduate (CST316) course in the Software Enterprise as a testbed to conduct our experiments using visual reports and learning analytics. Students in these courses use Scrum in order to conduct their projects as it embodies a philosophy of establishing a rhythm to weekly and even daily activities. In a typical software enterprise course, students are given the directive to work on their projects for a fixed number of hours per week, and save 1 hour out of class for preparation activities. Students are asked not to go beyond these specified hours even if their project is falling behind, as the intent is to have the students rely on a team process, not heroic efforts, to complete the sprint goal. Students are assessed after each sprint based on their process-related activities, code-related activity, integration of modules into the project, and team process activities. Individual process activities include

updating a scrum board and participating in standup meetings (at least 3 times per week). Code activities relate to individual velocity and are measured by frequency and significance of commits to the source code repository. Module integration is determined by a student journaling evidence and presenting it as part of a sprint individual reflection. Team process activities are based on adherence to Scrum principles such as user story development, estimation and tracking on burndown charts, and sprint reviews and retrospectives. A student's final grade is the aggregation of each of the sprint grades plus a percentage reserved for the end-of-semester product deliverables. Due to its highly modularized curricular organization, emphasis on highly iterative software development and expectation of continuous, consistent work on the project, the Software Enterprise provides a good vehicle for performing continuous assessment.

4.1 VISUAL REPORTS

The dashboard of visual reports codenamed "CAssess" (short for Continuous Assessment) was employed in three Software Enterprise based courses: two junior level Software Construction course (CST 316 during Spring 2015 and Spring 2016 which had two sections; online and face-to-face) and a graduate level course (SER515 during Fall 2015).

Students in the Spring 2015 batch of CST 316 were provided with the first version of CAssess (CAssess v1) in the final sprint of the course. CAssess v1 was a web-based platform that supports the integration of two tools: Scrumwise (scrumboard- www.scrumwise.com/) and GitHub (source code control tool- www.github.com). The platform is created using JAVA, JavaScript and extracts student activity data using open APIs exposed by these two tools. The API response is parsed by an appropriate library, transformed and stored in an array list. An authenticated request by the user prompts CAssess to retrieve student data in the form of an array list which is then visualized using

the C3JS JavaScript library. Data extracted from Scrumwise includes team statistics such as number of tasks owned by a team in the current sprint, the corresponding status and the remaining hours for completion of these tasks. Similarly, the data retrieved from GitHub includes individual team member statistics such as number of code commits and corresponding lines of code added and deleted over the period of a week. The visual reports are simple charts and graphs that provide students with a relatable and effective medium to analyze their progress on the project. Based on the results of a survey conducted at the end of the course, enhancements to CAssess v1 were envisioned that improved personalization (Chapter 5) and handling of real time data: CAssess v2.

CAssess v2 was provided to 90 graduate students in the SER 515 course during the Fall 2015 semester. CAssess v2 was employed as a personalized dashboard that supported a scrumboard (Taiga- https://taiga.io/) and a source code control tool (GitHub). The dashboard was created using the R programming language (www.r-project.org), which allows for efficient data collection, analysis and visualization. Utilizing the httr package in R, CAssess makes a GET call to the REST APIs exposed by each of these online tools (Taiga, Git Hub) on a daily basis. The data retrieved for each student is transformed by converting it into a data frame and stored in a local database. The data retrieved by CAssess v2 includes number of tasks owned by a student corresponding status and remaining time for completion. Likewise, it gathers statistics such as number of code commits, corresponding lines of code and deleted on a daily basis. At the end of every week, CAssess v2, analyzes a student's contribution for the week against the rubrics defined by the instructor. For example, students in the graduate-level project course are told that they must perform 3 significant code commits per week, and work on at least 2 tasks on the scrum board. Based on these rubrics defined and data collected by CAssess v2, it computes a score for each student and

generates a notification or a call to action based on their activity over the week. Every authenticated student request on CAssess v2 provides access to a dashboard containing visual reports using charts, graphs and a notification feature. Although CAssess v2 provided personalization and quasi real time visual reports. However, initializing student information on the dashboard and provisioning student accounts on online tools continued to be an onerous task, especially with increasing classroom sizes. This difficulty was resolved through the introduction of the third version of CAssess.

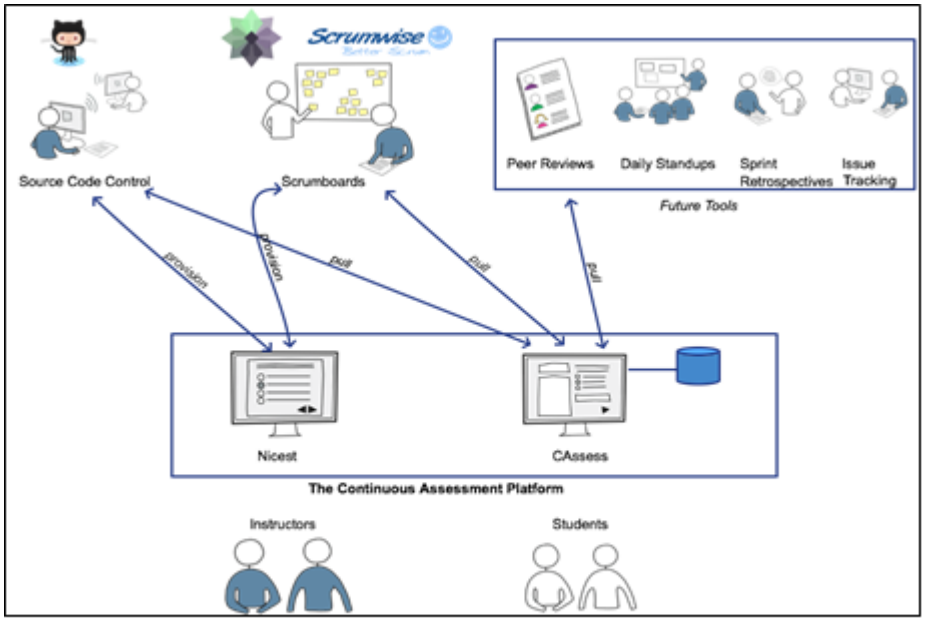

Figure 10: Architecture of the Continuous Assessment Platform (CAP)

The Continuous Assessment Platform (CAP) is designed to support adaptive interfaces to any tool of a given category. The vision for CAP is to extend the capability of dynamically integrating tools for generating learning activity streams via the "TinCan API" (also known as the Experience API or xAPI, https://tincanapi.com). The objective of the CAP is to gauge project activity and provide real time, formative feedback to enhance learning experience and ensure compliance in project-based courses. The current version of the CAP supports the integration of 3 tools: GitHub for source code management, and Scrumwise, Taiga for scrum boards. The current platform is a web-

based integration platform (Figure 10) composed of two subsystems, each with its own user-facing components. The primary component, CAssess v2 (here on referred to as CAssess) provides features for 1) integrating data streams from open tool APIs, 2) performing basic statistical analysis, and 3) displaying visualizations and notifications to students and instructors. The second and supporting system is called Nicest (Nicely Integrating Complex Education Software Together), and has primarily responsibilities for user and team management, and for provisioning the various tools being integrated into CAssess, again via open APIs.

Nicest (Murphy, 2016) is a tool integration platform for educators. Nicest (Figure 5) allows for provisioning of tools using “recipes”. Recipes are lightweight workflows modeled as an ordered list of steps (or tasks) that produce a type of project setup. For example, the “Code Project” recipe allows for configuration of GitHub (source code control), Taiga (scrum board), and CAssess for individual or team projects. Nicest divides recipes into extensible plugins that can be added, removed, or modified without affecting the core functionality or functioning of other plugins. However, plugins can optionally depend on another plugin to provide functionality, allowing plugins to serve as reusable sources of functionality for recipes to operate. Additionally, Nicest is designed to be able to pass project configuration information to CAssess so that dashboard knows what tools to pull from for student activity data. Passing details on tools that have been setup allows external systems to directly access those tools, in turn allowing instructors to provide qualitative analysis of student engagement.

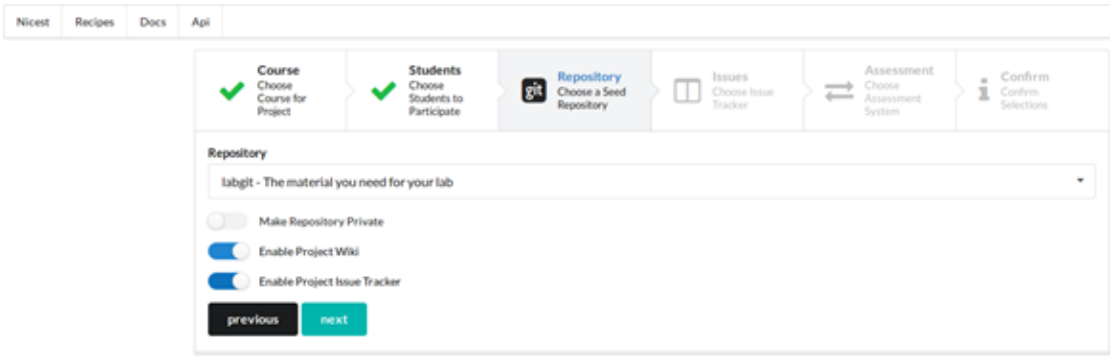

Figure 5: Sample Nicest Recipe Enactment

CAssess receives student and course information from Nicest, retrieves data from agile tools, analyzes student performance against instructor-defined rubrics, and provides visualizations and notifications via a personalized dashboard. The objective of CAssess is to leverage the transparency of the dashboard with the use of notifications to provide timely formative feedback, so students have the ability to immediately correct behaviors. The dashboard in its current version includes visualizations of daily activity on Taiga and GitHub, a peer comparison radial chart and a notification widget with directed feedback. Once Nicest (Chapter 5) provisions a course and the associated students into the online tools, it notifies CAssess by pushing this information to a local database. This prompts CAssess to generate a membership table linking a student with their accounts on these particular tools. Using the details of these accounts, CAssess makes GET calls to the API exposed by these tools, transforms the retrieved data and stores it into the same database. This data is then utilized by CAssess to power the various visual reports on the dashboard. To understand whether the students are utilizing the tool and are compliant in their learning process, a logging mechanism is implemented in CAssess. This helps understand how frequently students visit the dashboard, duration of their visit, tabs visited and notifications viewed.

4.2 LEARNING ANALYTICS USING MACHINE LEARNING ALGORITHMS

A significant amount of research is being conducted today in the field of learning analytics that is focused on collecting student data and performing continuous analysis to improve the learning experience. As discussed in chapter 2, (Petkovic, 2014) and (Ozatlin, 2015) describe how machine learning algorithms can be used to predict the outcome of student projects as being successful or innovative. However, classifying students into predefined labels such as “successful”, “not successful”, ”good”, “bad” does not provide insights about a student behavior during the period of the course. One aspect of this research work is focused on using machine learning algorithms to identify unique patterns of student behavior in class such that suitable labels can be assigned in order to help instructors understand typical student behavior in class. Understanding these unique patterns can help instructors make interventions for those clusters of students who may be in most need of it. In this section we explore the implementation of two unsupervised machine learning algorithms (K-means clustering, Hierarchical clustering) that can be utilized to identify patterns in student activity datasets.

K-means (Hartigan, 1979) clustering is a commonly used yet simplistic algorithm that efficiently groups data into a predefined number of clusters. The algorithm begins by creating K centroids where K is the number of clusters specified by the user. Every new instance of the dataset is then compared to these K centroids and assigned to the one that is closest to it. The distance between a new observation and the centroids is calculated using the Euclidean distance between the two. On addition of a new observation to an existing cluster, the centroid is recalculated and if needed the observations are redistributed to other clusters based on their proximities to the newly created centroid. The in-cluster variance is calculated by computing the mean of those observations which produce the least within-cluster sum of squares. The other

observations are then shuffled across the remaining clusters. This process repeats until no observations are found to be switching between clusters. A pseudocode for the K-means clustering is described below:

Repeat until no observations switch clusters

1. *Initialize K observations as cluster centroids*
2. *Assign new observation to a cluster based on the shortest Euclidean distance to its centroid*
3. *Recalculate centroid value based on mean of within-cluster sum of squares, move other observations to the next cluster.*

One of the characteristics of the K-means clustering algorithm is that the user needs to have prior knowledge of the number of clusters required. This can often pose a difficulty while using datasets such as the ones in our research. Fortunately, there are several methods available that provides an approximation of the number of clusters in a given dataset. In the elbow method (Hothorn, 2014), we plot a chart of within-cluster sum of squares against a sequential number of clusters and look for an "elbow" in the chart giving us an approximation of the number of clusters to use for the dataset. Figure 6 describes the within-cluster sum of squares plot for a student activity dataset. In the plot there is a slight elbow at cluster 5, so we experiment with values of k between 4 and 6 until a stable pattern is observed in each cluster.

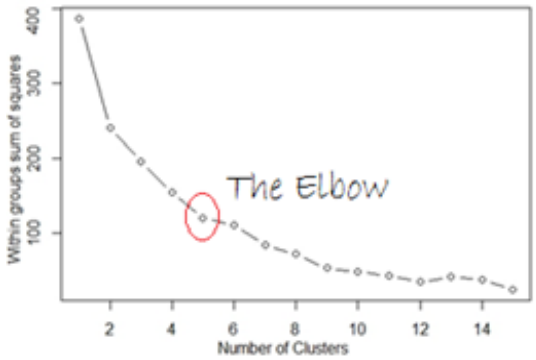

Figure 6: The Elbow Method

In order to compare the results of the clusters generated by the K-means clustering algorithm, we explore another unsupervised learning algorithm: Hierarchical clustering. The hierarchical clustering technique begins by creating an individual cluster (singletons) for every element of the input dataset. This is followed by the agglomeration phase wherein all the clusters are merged together until only a single cluster exists. The input to the agglomeration phase is a distance matrix and in the smallest two clusters are combined into one. The next step is to identify a suitable criterion that can combine the other singleton clusters. There are several methods that can be used for this process. One commonly used technique is the Ward's method of minimum variance (Ward, 1963). This method combines only those clusters where there is a minimum increase to the overall within-cluster variance. The variance is simply a measure of the weighted square distance between two cluster centers. Once all the clusters have been successfully merged, a tree like structure called a dendogram is generated that details the entire merging process (Figure 7).

The clusters generated by hierarchical clustering and K-means clustering algorithms were compared and it was observed that K-means clustering generated better, finer clusters in comparison to the hierarchical clustering method and thus,

subsequent experiments in this thesis were conducted using the K-means clustering algorithm.

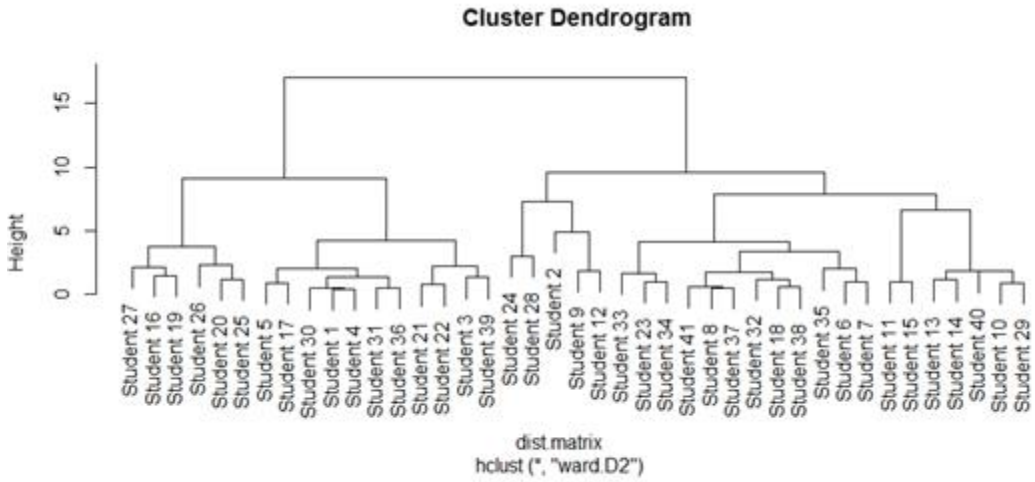

Figure 7: The Hierarchical Clustering Dendogram

In conclusion, this chapter describes the details of implementation for a dashboard (CAP) that is designed to integrate tools of any given category and provide visual reports and notifications to students in order to help them develop a rhythm of consistent working. Further, this chapter also describes the working of unsupervised machine learning algorithms that is used to cluster student activity datasets in order to obtain distinct patterns of student behavior. We explore two unsupervised learning algorithms and shortlist one of them (K-means clustering) in order to conduct further experiments. The next two chapters describe the experiments, results and interpretation of visual reports and learning analytics.

CHAPTER 5

VISUAL REPORTS

One purpose of this thesis (as described in RQ1) is to evaluate the use of visual reports as a means of providing continuous assessment for students in a project-based course and to measure the impact of these visual reports on student's compliance to the learning protocol. Students are provided with quasi real-time visual reports and notifications via a personalized dashboard and the impact of these reports on student's compliance is measured through surveys and logs of dashboard activity. These visual reports (CAssess v1, v2, v3) were provided to students in three different courses (both graduate and undergraduate). This chapter analyzes the experiments that were conducted using these visual reports, the results of the experiments and the limitations observed with scope for future improvement.

5.1 CAssess v1

5.1.1 Experimental Context

CAssess v1 was provided to 42 students in the junior level Software Construction course (CST 316) during the Fall 2015 semester. CAssess v1 was deployed in Sprint 4 (the last full project sprint) of the course and it included visual reports represented using charts and graphs. Figure 8 describes all the visual reports provided to students as part of CAssess v1. Figure 8a shows team process progress. A data point on this chart represents the number of tasks owned by a team for a particular day. It measures, as a stacked area chart, the number of To-Do, In Progress, To Test, and Done tasks per day on the team's scrumboard. This particular chart shows good team progress over the sprint, as the tasks are getting moved across the states of the scrum board at a fairly consistent rate. Figure 8b is a view of an individual's code activity. The grouped bar charts show frequency and significance of commits normalized to a common visual scale

with a double y-axis chart. This particular student shows decent behavior, though in several cases there a number of insignificant commits, as shown by the blue bar being higher than the orange (commits outweighing code addition, meaning several trivial commits performed). The most important chart however, is Figure 8c, as it shows a view of the four components of the sprint grade, laid out on a radial chart. Further, the individual student's component scores (shown in the small dark area) are overlaid with the team's average component scores, and then the component scores of the entire class. The student can view immediately, her/his performance with respect to the rest of the class. These charts are worthwhile in that they show a rate of progress (Figure 8a, 8b) and instant comparison (Figure 8c) to cohorts within the class. They are easily interpreted by the student, and are rendered in one place. A survey was conducted at the end of the course to understand the impact of CAssess v1 on improving the student learning behavior. 22 students responded to the survey. The students were asked to rate how helpful they found each of the charts and if the stacked area chart of activity on the scrum board allowed students to understand team progress.

5.1.2 Results and Interpretation

Of the 22 students who responded to the survey, 15 students indicated that the stacked area chart (Figure 8a) helped them understand team progress, while 1 said it did not help and the other 6 remained neutral. For the individual chart of code activity (Figure 8b), 10 students indicated it helped, while 8 said it did not help and 4 were neutral. Finally, for the peer comparison chart (Figure 8c), 7 students said it helped, 5 said it did not, and surprisingly 10 were neutral. While the results overall were positive, particularly for Figure 8a, it was disappointing to note that the radial chart (Figure 8c) did not generate stronger opinions among the students. Off course this was an ad-hoc survey and the visual reports were only provided to students in sprint 4. We did not have

the time to determine if availability of this data on a daily basis during the entire semester would incur a change in results.

One of the limitations of CAssess v1 is that it lacked real time data. The stacked bar chart of student contribution on the source code control tool described student contribution on a weekly basis and not daily while the radial charts were powered with data from the sprint grading sheet (Figure 14) which is generated only at the end of each sprint. Further, the stacked area chart of activity on the scrumboard detailed a team's progress and did not highlight an individual's contribution. These factors meant that CAssess v1 lacked real time data and personalization.

In conclusion, while CAssess v1 provided visual reports that were easy to interpret and relatable, it lacked in personalization and real time data. These two limitations of CAssess v1 were addressed by CAssess v2 which created a personalized dashboard with individual progress charts and notifications.

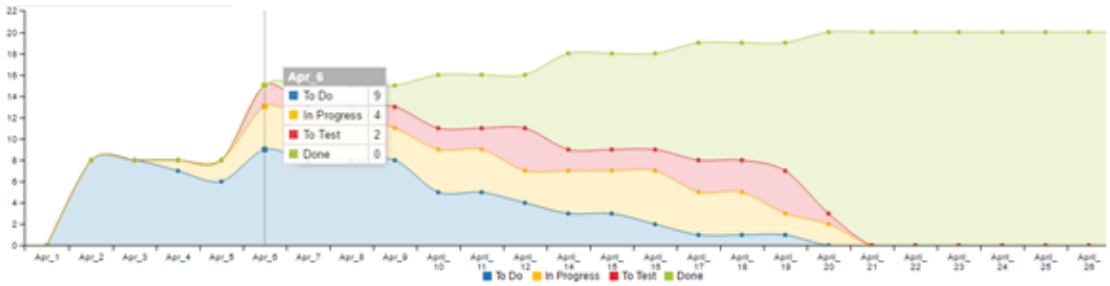

Figure 8a: CAssess v1- Stacked Area Chart- Scrum Board Team Progress

Figure 8b: CAssess v1- Stacked Bar Chart- Individual Code Activity

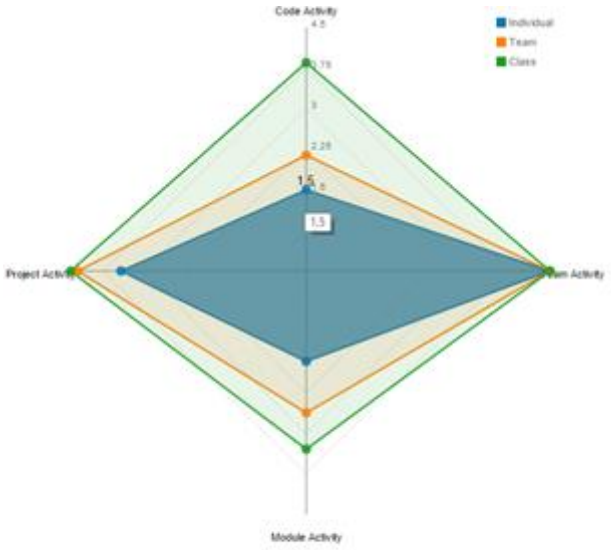

Figure 8c: CAssess v1- Radial Chart- Peer Comparison

5.2 CAssess v2

5.2.1 Experimental Context

CAssess v2 was provided to 90 students in a graduate Software Enterprise course during the Fall 2015 semester. Unlike v1, CAssess v2 was provided during the second sprint of the course so students could utilize the dashboard for a major period of the course. CAssess v2 addressed the limitation observed in v1 via a personalized dashboard equipped with quasi real time visual reports and notifications. The dashboard contained individual tabs that provided visual reports of a student's daily contribution on their scrum board and source code control tool. The dashboard also generates a notification or "call to action" every week based on a student's contribution against the learning expectations defined in class. Based on their contribution during a week, CAssess also generates a score for each tool and powers the radial chart which contrasts a student's weekly activity against that of their peers. Figure 9 describes the visual reports as part of the CAssess v2 dashboard. The notification feature (Figure 9a) in CAssess v2 is an attempt to personalize the dashboard and provide students with frequent formative

feedback. Figure 9b describes a stacked column chart on the dashboard that indicates the number of tasks owned by a student on a daily basis. This particular chart shows consistent activity on the scrum board as the student updates his/ her task status frequently. Figure 9c describes a stacked column chart detailing a student's contribution on the source code control tool daily. A radar chart (Figure 9a) is updated every week with axes for frequency and impact (normalized to a common scale). The radar chart allows the student to compare her/his contributions to other populations, which by default are the student's team and the entire class. As observed in Figure 9a, the personalized notifications and charts of individual contribution supported by quasi real time visual reports allows CAssess v2 overcome the drawbacks in CAssess v1.

Figure 9a: CAssess v2- Radial Chart- Peer Comparison

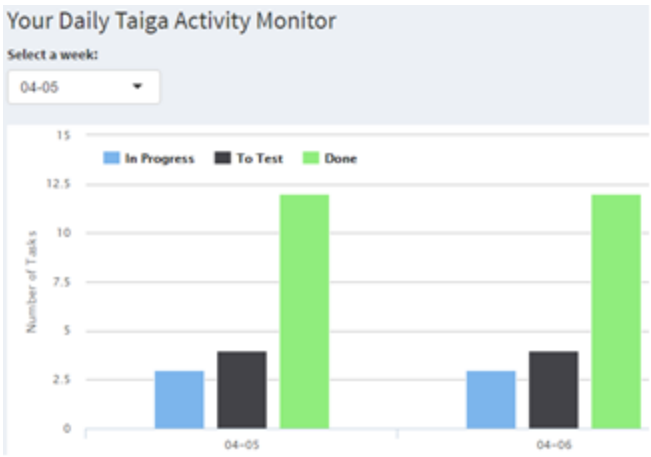

Figure 9b: CAssess v2- Stacked Bar Chart- Scrum Board Activity

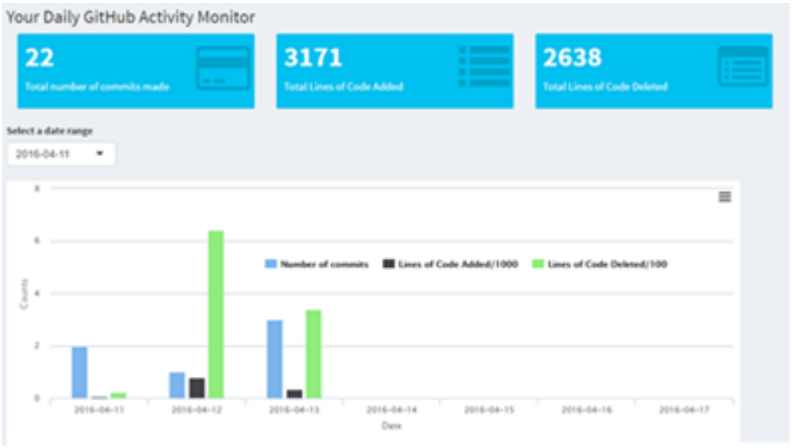

Figure 9c: CAssess v2- Stacked Bar Chart- Source Code Tool Activity

5.2.2 Results and Interpretation

An end of class survey was conducted to understand the impact of the visual reports on improving student behavior. The aim of the survey was to understand how frequently students visited the dashboard, if they liked the concept of a dashboard and the effect of notifications on improving student learning pattern. Students were also asked if the dashboard accurately reflected their individual contribution and if they wanted more tools such as stand up meetings logs, CI & Test tools integrated into the dashboard. 56 out of 90 students responded to the survey that included a multi-part 5-response Likert scale (“Strongly Disagree”, to “Strongly Agree”). The results of the survey are tabulated as shown in Table 1.

Question	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Total Response	Mean
I frequently checked the tool to verify my activity	1	6	13	27	9	56	3.66
I thought the tool gave accurate feedback on my project activity	7	19	13	14	2	55	2.73
I liked the tool	3	9	21	19	4	56	3.21
I wish more tools, like burndowns, CI &Test, and Google site activity were in the tool	1	5	17	22	11	56	3.66
The notifications feature help me know what I needed to do on the project (answer only if you had Notifications)	1	8	16	11	0	36	3.03

Table 1: Student survey response to CAssess Version 2

A majority of the students admit to having visited the dashboard frequently and also indicate that they would like to see more tools integrated into the dashboard. However, it is interesting to note that a significant amount of students specify that the dashboard did not accurately reflect their contribution on the online tools. This can be attributed in part to the limitation of CAssess v2 to only consider student code activity on the master branch of their source code control tool (GitHub). As discussed in chapter 4, an alternative option of using an online GitHub statistical tool, GItInspector was also explored, however GitInspector too provides statistics of student contribution only in the recently checked out branch and not the activity across all the branches.

The process of initializing student information on CAssess v2 and provisioning of student accounts on online tools (Taiga, GitHub) is an onerous task especially if the size of the class increases. CAssess v2 does not have a feature to provision student accounts and basic student information needs to entered manually into the system so CAssess v2 can retrieve student activity data via open APIs. This is another limitation of CAssess v2.

In conclusion, while CAssess v2 improved personalization through notifications, individual progress charts, quasi real time visual reports and faster loading times, it did have drawbacks in accurate data reporting (GitHub) and the process of student information initialization. These drawbacks of CAssess v2 was addressed through the introduction of the Continuous Assessment Platform (CAP)

5.3 CAssess v3

5.3.1 Experimental Context

While CAssess v2 addressed the limitations of v1, it had several drawbacks of its own. Primarily in the area of accurate data reporting and student information initialization. CAssess v3 addresses the issue of data accuracy observed in v2 via an R script that gathers statistics of code activity across all branches in the source code control tool (GitHub). Further, it integrates with another open source system, Nicest (Chapter 4) in order to remedy the issue of student account provisioning and student information initialization in CAssess. The instructor provisions student accounts on Nicest and this information is provided to CAssess v3 via a shared database. CAssess v3 utilizes this information to retrieve student activity data via open APIs and powers the visual reports on the dashboard. In addition a logging feature was implemented in CAssess v3 to track student activity on the dashboard. This feature helps keep a track of how long students visit the dashboard, tabs visited and notifications viewed. The visual reports provided in CAssess v3 are similar to the reports in CAssess v2 with the exception of a spline chart

that contrasts an individual's progress against the learning protocol. The additional charts observed in CAssess v3 in comparison to v2 are as seen in Figure 11a and Figure 11b. Figure 11a describes a student's code contributions displaying inconsistent activity patterns with unusual spikes observed during certain weeks of the sprint. Figure 11b shows the scrum board contribution of another student who is consistently compliant with the expectations of the course. The dashboard was provided to 62 students in the face-to-face class of the CST 316 batch of 2016 and 53 students in the online version of the same class. Students in the online class were provided access to the dashboard in the second and third sprints (out of three total sprints). However, these students were not trained on using the dashboard and only 16 out of 53 students visited the dashboard on their own accord. However, in the face-to-face class, the dashboard was introduced in the second sprint (out of four sprints) and students were provided with classroom training on how to utilize the dashboard.

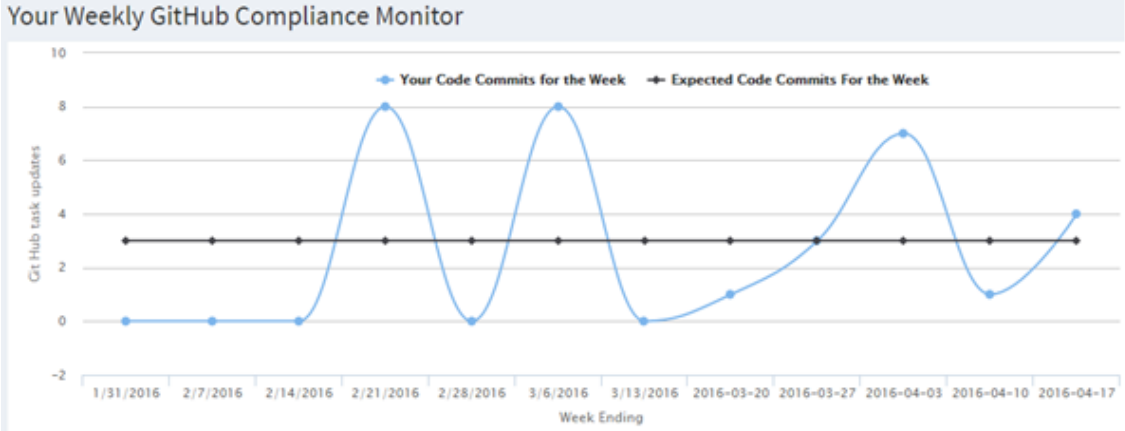

Figure 11a: Spline Chart Showing Student Progress on Scrum Board

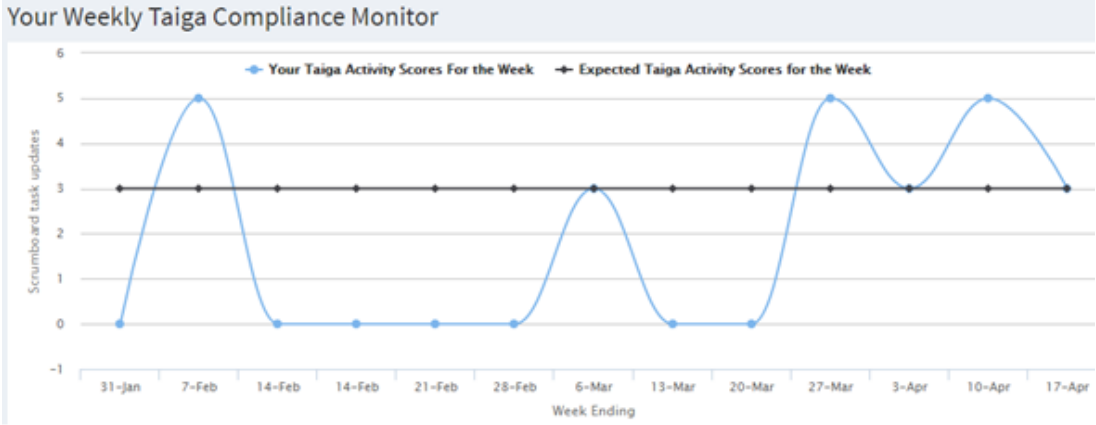

Figure 11b: Spline Chart Showing Student Progress on Source Code Control Tool (GitHub)

5.3.2 Results and Interpretations

An end of semester survey was conducted in order to understand the impact of CAssess v3 in improving student behavior in the project-based course. Separate surveys were provided to the online and face-to-face classes including how often students visited the dashboard, the accuracy, the need for more tools, impact of notifications and finally if the students liked the concept of a personalized dashboard. A detailed analysis of student response on the survey to each question (in the online and face-to-face class), coupled with results of the logging feature of CAssess v3 is described below.

A visual representation of the survey results from the online class is as observed in Figure 12. Although 52 students responded to the survey, we only consider and analyze the responses of those 16 students who visited the dashboard on their own accord without any training.

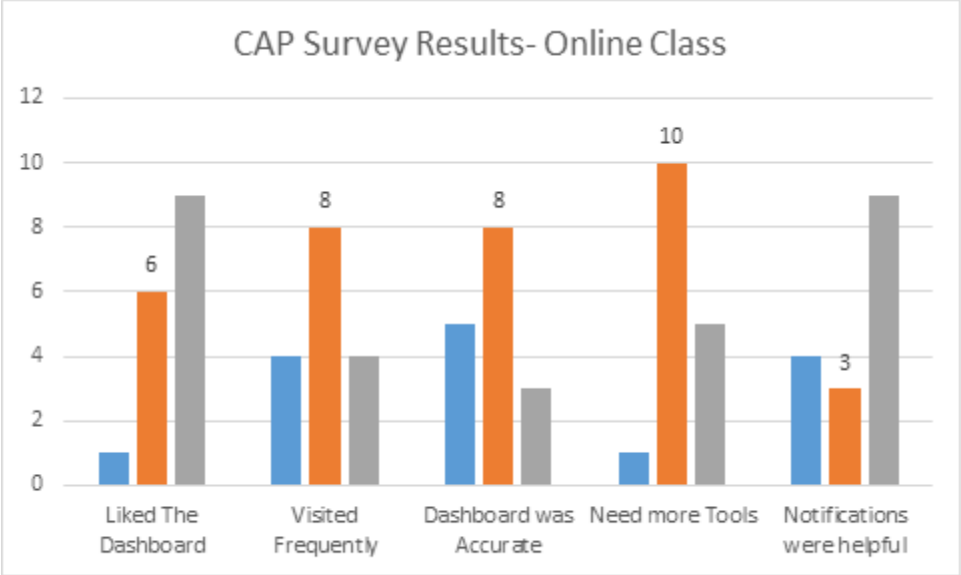

Figure 12: Student CAP Survey Response- Online Class

"Frequency": 4/ 16 (25%) students said that they did not check the dashboard frequently. The log of their dashboard activity concurs with their response. 8/ 16 (50%) students said they checked the dashboard frequently of which 1 student has only visited the dashboard once. The rest have visited the dashboard more than 5 times post the introduction of the dashboard. The remaining students were neutral. "Accuracy": 5 out of 16 (31%) said that the dashboard did not accurately reflect their contribution on Taiga or GitHub while 8 out of 16 (50%) agreed/ strongly agreed that the dashboard accurately reflected their contribution. 4 of these students visited the dashboard less than 5 times. "Like the tool": only 1 of 16 (6%) students said that they did not like the tool while 9/ 16 (56%) said that agree/ strongly agree that they liked the tool. "Need more tools in the dashboard": Only 1 of 16 (6%) said that they do not agree with more tools being added to the dashboard while 10/16 (62%) said that they agree/ strongly agree with the addition of more tools in the dashboard. "Notifications were helpful": 4 out of 16 (25%) said that the notification did not help them in understanding what they needed to do in the project while only 3 (19%) said that they agree/ completely agree that the notifications helped them understand what they needed to do in the project. The rest were neutral.

Almost half of the online class students responded positively regarding the accuracy, frequency of dashboard visit, needing more tools and liking the dashboard. Surprisingly, only 3 of the 16 students admitted to finding the notification feature useful. While overall the results of the survey of the online class shows an improvement from the results observed in the SER 515 class of 2015, we need to assess why the notification feature does not seem to have the impact that was expected in this research.

Students in the face-to-face class of CST 316 (Spring 2016) were provided with training on using the dashboard and all 62 students in class responded to the end of semester survey. A visual representation of the responses is described in Figure 13.

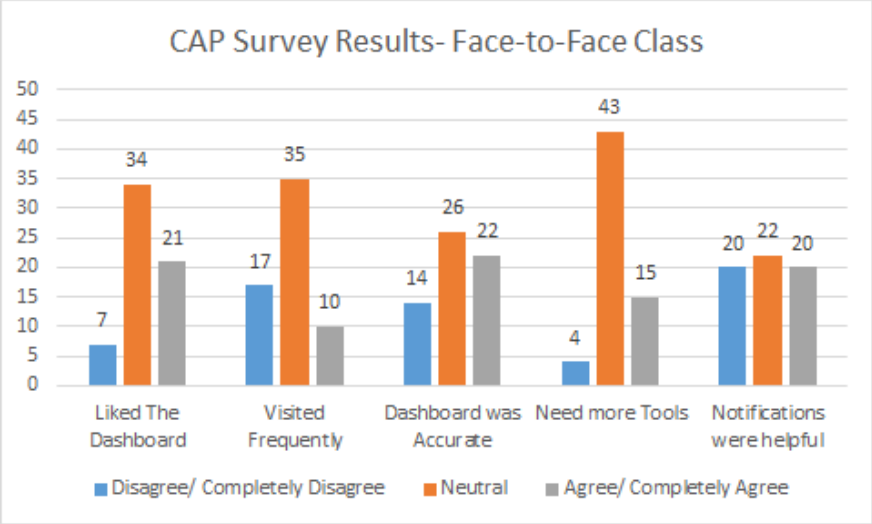

Figure 13: Student CAP Survey Response- Face-to-Face Class

"Frequency": 17 out of 62 (27%) said that they did not check the tool frequently. The dashboard activity log concurs with this observation. However there are 3 students in this group who have visited the dashboard more than 15 times in the semester. 35/62 (56%) said that they visited the dashboard frequently. However 6 out of these students have only visited dashboard less than 10 times in the entire semester. The remaining 18 students said they neither agree nor disagree. It is interesting to note that of the 18, 6 students have visited the tool more than 10 times in the semester. "Accuracy": 22 out of

62 (35%) said that the dashboard did not accurately reflect their contribution on Taiga or GitHub. Of the 17, 9 students have visited the dashboard less than 10 times. Only 26 out of 62 (42%) agree/ strongly agree that the dashboard accurately reflected their contribution on both the tools- out of which 18 students have visited the dashboard more than 15 times in the entire semester. “Like the tool”: Only 7 of the 62 (11%) students said that they disliked the tool while 34/ 62 (55%) agreed/ strongly agreed that they liked the tool. The remaining students remained neutral. “Need more tools in the dashboard: Only 4 students out of 62 (6%) said that they wouldn’t want more tools to be incorporated into the dashboard. While 43 /62 (69%) said that they agree/ strongly agree that they would like it if more tools were incorporated. The remaining students were neutral. Notifications were helpful: 20 /62 (32%) said that the notification feature did not help them know that they needed to do in the project while 22/ 62 (35%) agreed/ strongly agreed that the tool helped them to know what they needed to do in the project. The remaining students remained neutral.

A distinctly similar pattern is observed in the response to the survey questions between the face-to-face and online classes. A vast majority of students responded positively to the questions regarding frequency of dashboard visit, need for more tools and liking the dashboard. However, when it comes to accuracy, although a large number of students agree that the dashboard accurately reflects their contribution, surprisingly, a significant number of students remained neutral. With regard to the effect of notifications on improving student learning behavior, the responses were almost equally split. This can be attributed to several factors. It has been observed that students tend to misinterpret the visual reports; e.g. students are expected to develop a rhythm of consistent working and to that end it is expected that they make frequent updates to the status of their tasks on the scrum board based on regular code commits. However,

students who do not follow the concept of consistent working and complete tasks in bursts expect positive notifications since they "manage to get it all done". Another factor that affects the accuracy of the reports is that students tend to not follow guidelines and commit code on their repository under different user ids. This impacts the data collected by CAssess and subsequently the visual reports displayed on the dashboard.

Overall, the Continuous Assessment Platform addresses the drawbacks observed in CAssess v2 using Nicest and an R script that fetches statistics from every branch in the source code control repository. However, CAP in its current form has certain limitations as well. The vision for the CAP is to support adaptive interfaces to any tool of a given category. In the current version, CAP only supports the integration of 2 types of tools (scrum board and source code control) while students in the project are expected to utilize more tools such as Slack (stand up meetings), Jenkins (CI & Test) and Qualtrics (peer review). Providing visual reports of activity on only two tools does not entirely reflect a student's contribution on their project and is currently the biggest limitation of CAssess. This limitation provides scope for further improvement of the dashboard and this is discussed in chapter 7.

In conclusion, while the results of these three experiments are promising, it does not provide sufficient evidence to conclude that these visual reports have a significant impact in improving student compliance to the learning protocol. It is an onerous task to get students to believe the data captured by CAP and is even more challenging to get students to use another website to view their progress in addition to the online tools they already utilize. This research work takes on the challenge of getting students to use a dashboard that prompts students to change their deadline driven behavior in exchange for a habit of consistent working. Although we try, we do not receive the expected

response to the dashboard and thus more challenging questions need to be tackled in order to make the dashboard of visual reports more effective.

The next chapter discusses an alternative form of performing continuous assessment; using learning analytics. Unsupervised machine learning algorithms are used to identify distinct patterns of student behavior so instructors can understand typical characteristics displayed by students in the context of the course and make targeted and timely interventions.

CHAPTER 6

LEARNING ANALYTICS USING MACHINE LEARNING ALGORITHMS

This research is devoted to identifying novel and innovative means of providing continuous assessment to students in a project-based course. To this end, we explore the use of learning analytics and machine learning algorithms to identify distinct patterns of student behavior to help instructors perform targeted and timely interventions. As discussed in chapter 4, we use the K-means clustering algorithm to identify patterns of student activity in two different datasets; dataset of sprint grades, dataset of daily activity on online tools.

6.1 CLUSTERING GRADED SCORES DATA

6.1.1 Experimental Context

In this research, we use the Software Enterprise as a test bed to conduct experiments using learning analytics and machine learning algorithms. We hypothesize that an unsupervised machine learning algorithm (K-means clustering) can be used to identify unique patterns of student behavior in a project-based course. The first set of experiments to test this hypothesis was conducted on a simple dataset obtained from the sprint grading sheets for the CST 316 course of 2014 and 2015. Students in the Software Construction course use Scrum to conduct their projects. Students are graded after every sprint based on their process activity (scrum board updates, stand-up meeting participation), code activity (frequency and significance of commits), module activity (integrating software engineering modules into the project) and delivery of working software. A sample sprint grading sheet assigning scores for Process Activity (PA), code activity (CA) and module activity (M), agile project management (APM) and delivery of working software (DWS) is shown in Figure 14. Students are graded on a scale of 5 based

on contribution and consistency of their contribution during the sprint and the instructor or the grader assigns these grades.

The Code Activity, Process Activity and Module Activity scores (see Chapter 4) are extracted separately and subjected to the K-means clustering algorithm. We extracted each activity separately because clustering on the overall scores may not reveal smaller facets of student behavior. For example, we observe students who write excellent code but do not partake equally well in the process activity. The elbow method discussed in chapter 4 is utilized to evaluate the estimated value of K and the experiment is repeated several times using multiple values of K. This experiment is conducted separately on the sprint grades dataset for the 2014 and 2015 classes.

		CA(5)	PA(5)	M(5)	Total	Comments
Student 1	Sprint 1	5	5	4	23	Good activity on GitHub and Taiga. (
	Sprint 2					
	Sprint 3					
Student 2	Sprint 1	3.5	5	4	21.5	Need more code activity on GitHub
	Sprint 2					
	Sprint 3					
Student 3	Sprint 1	3.5	4	0	16.5	Very few tasks owned in Sprint 1
	Sprint 2					
	Sprint 3					
Student 4	Sprint 1	5	5	5	24	Good work!
	Sprint 2					
	Sprint 3					
Team Rub	DWS(5)	APM(5)	Comments			
Sprint1	5	4	The team still has user stories from Sprint 1 that are in progress and			

Figure 14: Sample Sprint Grading Sheet

6.1.2 Results and Interpretations

This section describes the results of applying K-means clustering algorithm to the dataset of sprint grades assigned to students in the 2014 and 2015 batch of the Software Construction course. The result of clustering each dimension of activity (code, process and module) can be summarized as observed in Table 2.

Code	**2014 (43)**	**2015 (41)**	***Process***	**2014**	**2015**	**Module**	**2014**	**2015**
Excellent	19	17	*Excellent*	8	13	*Excellent*	20	9
Good	6	4	*Good*	16	12	*Good*	7	N/A
Incline	3	4	*Incline*	N/A	6	*Incline*	N/A	5
Decline	8	6	*Decline*	9	5	*Decline*	5	9
Poor	7	4	*Inconsistent*	8	5	*Inconsistent*	N/A	14
Exception	N/A	6	*Poor*	2	N/A	*Exception*	11	3

Table 2: Cluster Information for the 2014 and 2015 graded dataset

Table 2 shows the number of data points per each cluster per each dimension. Cluster labels correspond to the instructor's assessment of individual student activity in each dimension over the course of all sprints in that semester. *Excellent* means the student demonstrated consistently high frequency and significance of the activity over all sprints. *Good* means the students demonstrated consistently above average evaluation but not high enough to be in the Excellent cluster. *Incline* means the student scored poorly in the initial sprint but then demonstrated increased scores thereafter; whereas *Decline* suggests the opposite, scoring well in the first sprint but then performing poorly by the end of the project. Inconsistent means that the student demonstrated variations between high and low scores in consecutive sprints. *Poor* is for students that demonstrated consistently low scores. Exception implies a form of outlier, usually a single sprint that shows a dramatically different behavior than the others. We did not discern any causality to the outliers, specifically looking for when in the semester the outliers occur.

The number of unique clusters in each dimension varied between 5 and 6 depending on year, though differences exist in the set of clusters. The size of each cluster is reasonably consistent across dimensions and years when reading them in rank order (top to bottom). While we have not accounted for bias in the preliminary analysis, we note the same instructor was responsible for assigning scores in both classes, and the course and project requirements stayed essentially the same, as did the class size.

While the differences between years in each dimension are interesting (particularly for process and module), we find it more interesting that there is a similarity between the identified clusters that relates to a pattern of behavior. This is especially clear in the code dimension. If the set clusters remain stable over time, it represents an expected set of student behaviors for which interventions may be designed. The most exciting result however, is to consider the relationship between dimensions. Intuitively one might expect that the rows in Table 2 align; that is, a student with Excellent code activity also has Excellent process and module activity. This is often but not always the case. For example, closer examination of the 2015 dataset shows that students in Decline in the code dimension are counter-intuitively usually in the Excellent or Incline clusters of process activity though they are in Decline or Inconsistent for modules.

The distinct clusters created and labeled above indicate to the success of the generic version of the K-means clustering algorithm in identifying distinct patterns of behavior in the sprint grades dataset. However, this dataset comprises of an indirect measure (a score on a scale of 5) assigned to a student by the instructor. Since the aim of the research is to encourage students to develop a rhythm of consistent working, analyzing sprint grades may not accurately explain student contribution patterns. This is primarily because our research is focused on changing student behavioral patterns and these indirect measures are not entirely reflective of a student's contribution. Further, this research is devoted to providing frequent and timely feedback to motivate a rhythm of constant changes to the learning process. The nature of sprint grades make it infrequent and delayed and there exists a need to explore another level of granularity, a direct measure of student contribution; the daily activity dataset.

In conclusion, applying K-means clustering on the dataset of student sprint grade scores has been insightful in establishing the ability of the algorithm to identify distinct patterns of student activity. However, for the purpose of this research, sprint grades are an indirect measure and do not entirely reflect a student's contribution on the project. This prompts the need for apply K-means clustering algorithm on a dataset that describes the daily contribution of students on online agile tools.

6.2 CLUSTERING ON DAILY ACTIVITY DATASET

6.2.1 Experimental Context

The K-means clustering algorithm applied to the sprint grades dataset reveals distinct patterns of student behavior observed in the context of the course. However, these results are based on a dataset containing scores that were assigned by a grader or the instructor. These indirect measure of evaluation do not necessarily reflect the extent of a student's contribution and we wish to drive the feedback period to zero. Thus the second phase of our experiments based on the student's activity on online agile tools (scrum board and source code control) on a daily basis. These set of experiments are conducted on a dataset obtained from student activity in the online (53 students) and face-to-face version (62 students) of CST 316 course of 2016. Data is collected daily based on the number of tasks owned and its corresponding status on the scrumboard (Taiga), and the number of significant code commits made to the source code control tool (GitHub). Data collection is performed through CAssess by making GET calls to the open APIs exposed by each of these tools. The retrieved data is transformed and stored in a local database. An extract of the daily activity dataset is as shown in Figure 15.

in_progress	to_test	done	date	name
0	0	0	25-Jan	Student1
0	0	0	26-Jan	Student1
0	0	0	27-Jan	Student1
1	2	1	28-Jan	Student1
1	0	3	29-Jan	Student1
1	0	3	30-Jan	Student1
0	1	5	31-Jan	Student1
0	1	5	1-Feb	Student1
0	1	5	2-Feb	Student1
1	1	5	3-Feb	Student1
1	1	5	4-Feb	Student1
2	1	5	5-Feb	Student1
1	5	5	6-Feb	Student1
0	4	9	7-Feb	Student1
0	4	9	8-Feb	Student1
2	4	9	9-Feb	Student1
1	5	9	10-Feb	Student1
1	6	9	11-Feb	Student1
1	7	9	12-Feb	Student1
1	7	9	13-Feb	Student1

name	date	commits	loca	locd
Student1	2/7/2016	0	0	0
Student1	2/8/2016	0	0	0
Student1	2/9/2016	0	0	0
Student1	2/10/2016	0	0	0
Student1	2/11/2016	0	0	0
Student1	2/12/2016	0	0	0
Student1	2/13/2016	9	818	1365
Student1	2/14/2016	3	3	60
Student1	2/15/2016	3	1020	319
Student1	2/16/2016	0	0	0
Student1	2/17/2016	0	0	0
Student1	2/18/2016	0	0	0
Student1	2/19/2016	0	0	0
Student1	2/20/2016	0	0	0
Student1	2/21/2016	1	185	15
Student1	2/22/2016	0	0	0
Student1	2/23/2016	0	0	0
Student1	2/24/2016	0	0	0
Student1	2/25/2016	0	0	0
Student1	2/26/2016	0	0	0
Student1	2/27/2016	1	392	46
Student1	2/28/2016	2	609	245

Figure 15: Sample Daily Activity Dataset. Taiga (L), GitHub (R)

The base dataset used for the experiment is the database of daily student activity on Taiga and GitHub. The base dataset is then converted to a binary format where the ones represent student activity for a particular day on a particular tool and the zeroes represent no activity for the day. Since the purpose of the experiment is to analyze compliance patterns among students and enforce the habit of consistent working, the binary representation of the data would be beneficial in understanding how frequently a student contributed on the tool. Summation of all the ones in every row (for each week) indicates the number of days in the week that the student was active on the said tool. The learning protocol defined for the course expects students to contribute significant code on their source code control tool (GitHub), at least twice every week and a corresponding number of task updates is expected on their scrumboard. Using the weekly summation dataset of each sprint for GitHub and Taiga as source, the K-means clustering algorithm is applied to identify unique patterns of student activity. Each cluster is then assigned a label based on the distinctive behavior observed in the cluster. The experiment is repeated for all the sprints to identify progressive behavior of each student for the entirety of the course. Further, the experiment is repeated on the combined dataset of

GitHub and Taiga activity to identify patterns of behavior in each sprint across the two tools. On completion, we analyze the statistics obtained from this experiment including the number of students in each cluster for each of the sprints, the migratory patterns and finally discuss if each student can be assigned a distinct label based on their compliance behavior on both the tools during the entire semester.

6.2.2 Results and Interpretations

The K-means clustering algorithm is applied on the daily student activity datasets of the online and face-to-face class of CST 316 course during Spring 2016. The same number of distinct clusters has been observed in both versions of the class. Each of these clusters are labeled based on the behavior observed in each cluster. The labels assigned to each cluster and their description is as describe below.

i. *Poor*: Students in this cluster display very low signs of compliance by getting on each tool not more than 2-3 times during the course of the entire sprint.

ii. *FinalWeek*: Students in these cluster displayed activity on both the tools- GitHub and Taiga only during the final week of the sprint.

iii. *ScrumCentric*: Students in this cluster displayed significantly more contribution on Taiga in comparison to their activity on GitHub.

iv. *CodeCentric*: Students in this cluster displayed more activity on GitHub compared to their activity on Taiga.

v. *Inconsistent*: Students in this cluster displayed wavering activity on both the tools during the course of the sprint. These are students who contribute on both the tools more than a student who is classified as poor does but less than the class expectation.

vi. *Compliant*: Students in this cluster displayed consistent activity over the entire period of the sprint period on both the tools.

The numeric distribution of students in each cluster for the face-to-face and online class is described in Figure 16 and Figure 17.

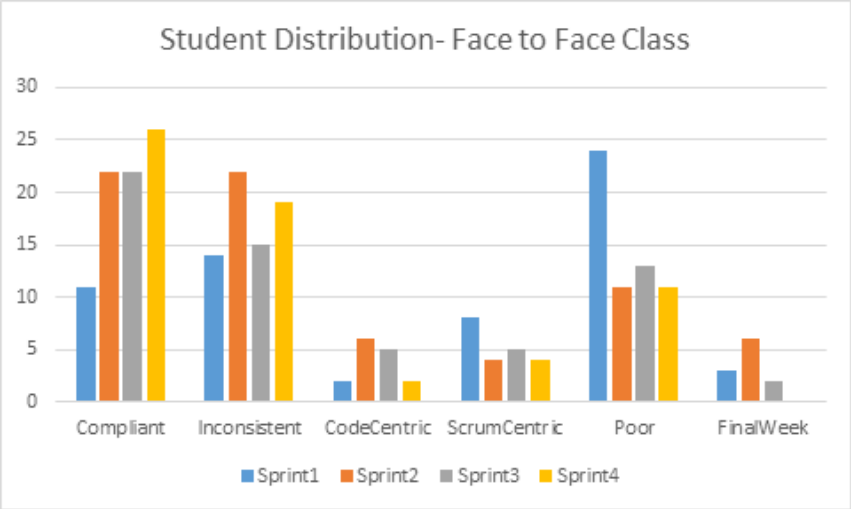

Figure 16: Student Distribution- F2F Class

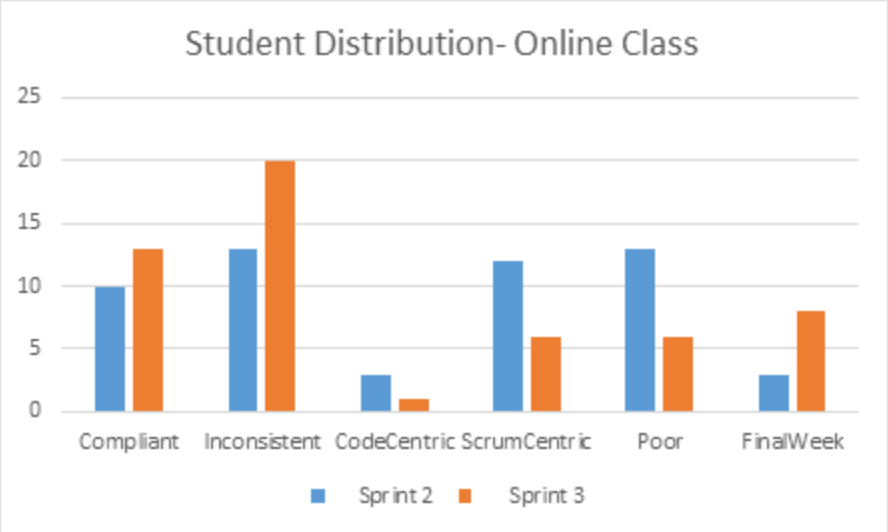

Figure 17: Student Distribution- Online Class

Comparing the student distribution of students in each cluster for both versions of the course offers interesting insights. As observed, there exists similar pattern of transition in the last two sprints for the Compliant, Inconsistent and CodeCentric clusters. In the face-to-face version of the course, there is marginal difference in the number of students in the Poor and ScrumCentric clusters. This however is not the case

in the case with the online class, which shows a significant drop in the numbers between the last two sprints. The only significant difference in student transition patterns are observed only in the FinalWeek cluster which shows an increase in number in the online version while the number of student in the cluster drops significantly in the face-to-face version of the course.

Analysis of the results of the clustering algorithm reveals a stable number (6) of clusters in both versions of the class (online and face-to-face). While analyzing the number of students in each cluster for both versions of the class provides insights into how the population of clusters vary between the sprints, the focus however must be on individual student patterns. In this research we are focused on identifying those group of students who display a change in behavior during the period of the course. A change in cluster indicates a change in behavior and it is important to understand the factors that impacted this change. These factors can include grades, class break, personal or team issues. Analyzing these factors that prompt a change in student behavior for both the online and face-to-face medium is crucial in providing personalized, formative feedback. This process of analyzing change in behavior by a student who migrates between clusters is referred to as "migration analysis" and is visually represented as shown below.

We first discuss the migration analysis in the face-to-face class that describes how students in each cluster migrate over the course of the semester. Table 3 provides a tabular version of students in each cluster.

Cluster Name	Sprint1	Sprint2	Sprint3	Sprint4
Compliant	11	22	22	26
Inconsistent	14	22	15	19
CodeCentric	2	6	5	2
ScrumCentric	8	4	5	4
Poor	24	11	13	11
FinalWeek	3	6	2	0

Table 3: Student Distribution in each cluster (face-to-face)

Poor: Of the 24 students who were classified as being poor in compliance for sprint 1, only 2 of them continued the same behavior in sprint 4. 10 students were classified as being compliant in sprint 4. However only 4 of those students showed a steady improvement in compliance. The remaining 6 displayed traits of consistency only in the final sprint. Figure 18 describes the migration patterns of all students who were classified as being poor in sprint 1. The orange arrows indicate those students who display a significant change in their migration patterns. (e.g. students who move all from the poor cluster all the way to compliant post the first sprint)In this case students highlighted in orange display traits of progress by migrating from the poor cluster in sprint 1 to the compliant and inconsistent clusters in subsequent sprints and continue that pattern of compliance.

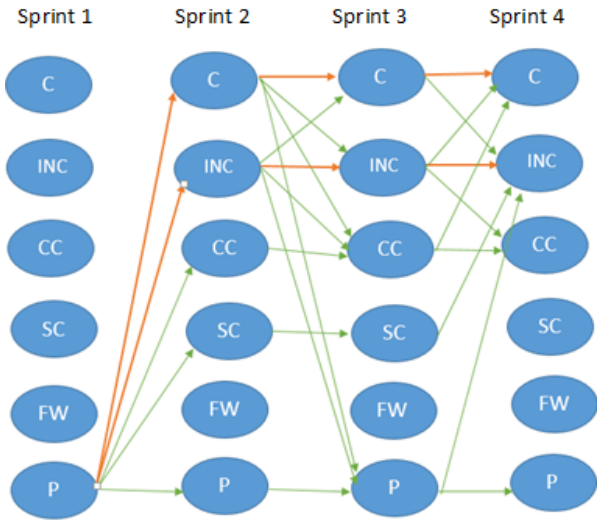

Figure 18: Migration Analysis: Poor Cluster- F2F Class

FinalWeek: All the 3 students who were classified into this cluster for contributing on the scrum board and source code control tool only in the last week of the sprint eventually wound up in the poor cluster. Figure 19 displays the migratory patterns of all students in this cluster.

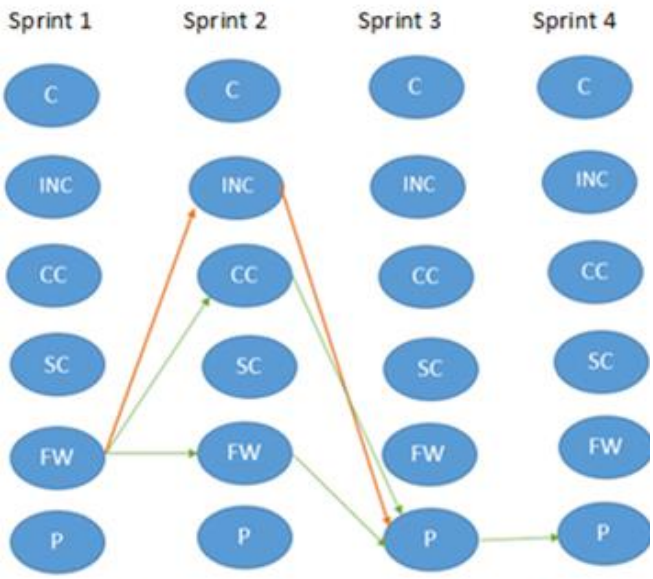

Figure 19: Migration Analysis: FinalWeek Cluster- F2F Class

ScrumCentric: Of the 8 students in sprint 1 who contributed more on their scrum board than on their source code control tool, only 2 continued that behavior in sprint 4. However, students who start their sprint in this cluster do not indicate signs of consistency across the sprints. Most students who were classified into the compliant cluster at some stage of the course, could not manage to continue that behavior in subsequent sprints. Figure 20 describes the transition pattern of students who make up for low code contribution by making excessive contribution on their scrum boards. The arrows marked in orange highlight those students who make drastic changes in their compliance patterns during the course of the semester. In this particular case students who start out as being classified as ScrumCentric progress to the Compliant cluster. It is important to observe that students in this cluster do not display consistent progress through sprints.

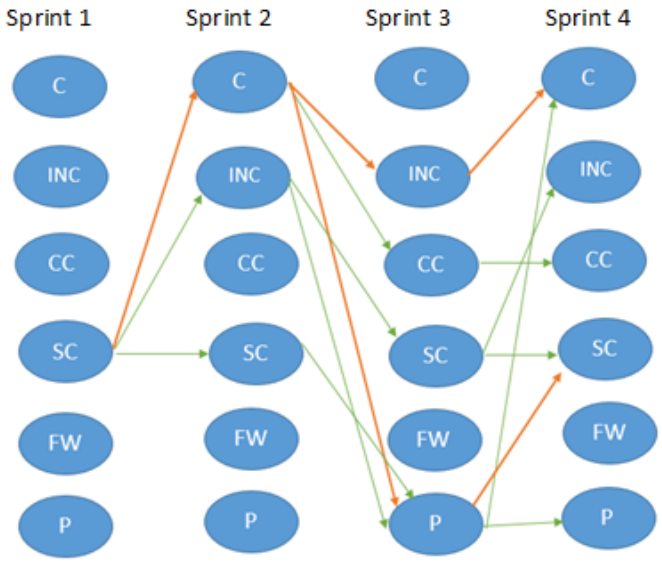

Figure 20: Migration Analysis: ScrumCentric Cluster- F2F Class

CodeCentric: An assumption of students in this cluster was that students who contribute more code in comparison to their scrum board activity, generally tend to continue that behavior across sprints. However, in this case all students who displayed behavior of contributing more code than their scrum board activity ended up in the poor cluster at the end of the sprint (Figure 21).

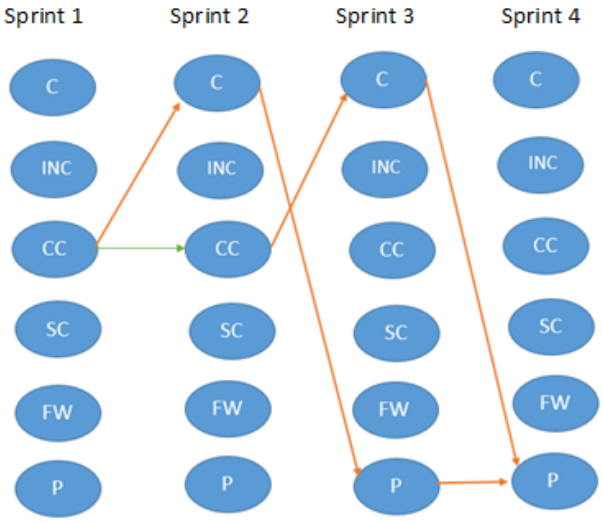

Figure 21: Migration Analysis: CodeCentric Cluster- F2F Class

Inconsistent: Of the 14 students who were classified as being inconsistent with their compliance to the learning protocol, a majority of the students continued that behavior or displayed signs of decline in compliance. However, of all the students who improve their performance and are grouped as being compliant, most of them have continued the behavior of compliance into subsequent sprints except for one student. (Figure 22). The orange arrows indicate those students who display significantly varying patterns of behavior between sprints. In this particular case a student who continues in the Inconsistent cluster till sprint 3 drops to the Poor cluster in sprint 4. Likewise the black arrows indicate those group of students who migrate from the Inconsistent cluster to the Poor cluster and back to Inconsistent.

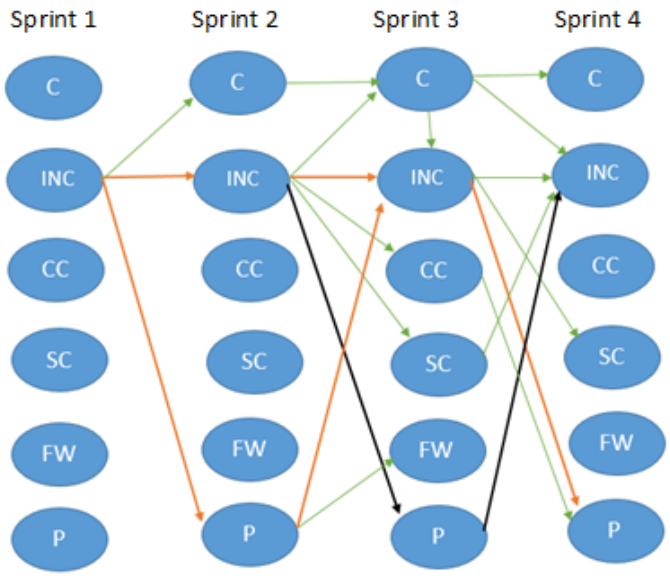

Figure 22: Migration Analysis: Inconsistent Cluster- F2F Class

Compliant: A vast majority of the students who start out as being compliant, continue that behavior across the subsequent sprints. It is interesting to note that students who show signs of decline in behavior, eventually end up being compliant in the final sprint except for one student. (Figure 23)

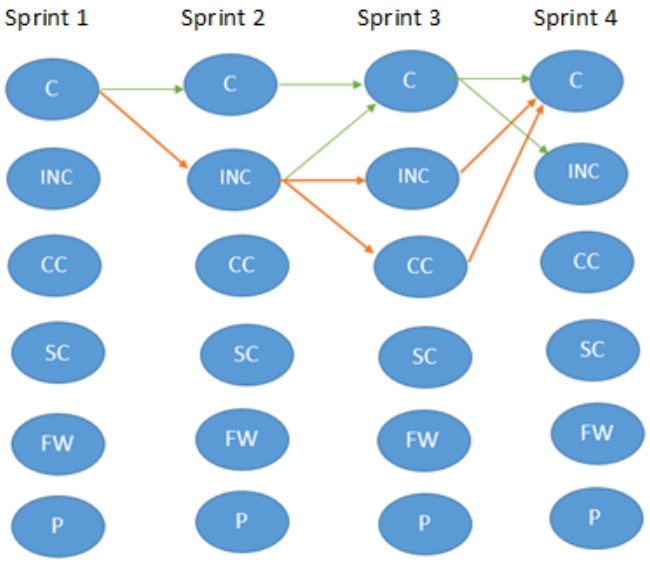

Figure 23: Migration Analysis: Compliant Cluster- F2F Class

The section below describes the migration analysis of students in each cluster of the online class. The tabular representation of students in each cluster is described in Table 4.

Cluster Name	Sprint 2	Sprint 3
Compliant	10	13
Inconsistent	13	20
CodeCentric	3	1
ScrumCentric	12	6
Poor	13	6
FinalWeek	3	8

Table 4: Student Distribution in each cluster (Online)

Poor: Of the 13 students who were classified into the poor cluster in sprint 2, only 4 of these students continued displaying poor behavior in sprint 3. The others however displayed only marginal progress- 3 students moved into the Inconsistent and FinalWeek clusters and 1 each to the remaining 3 clusters. Figure 24 shows a pie chart describing the transition of student population in the poor cluster from sprint 2 to sprint3.

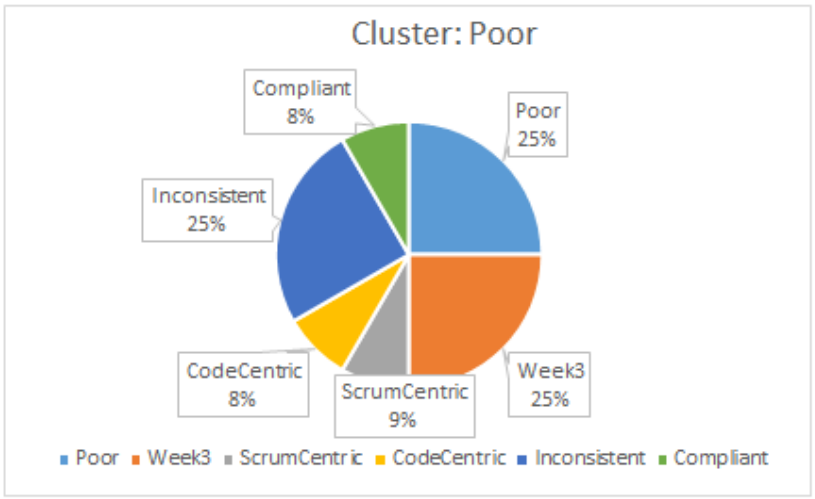

Figure 24: Migration Analysis: Poor Cluster: Online Class

FinalWeek: Students in this cluster displayed any activity on GitHub and Taiga only in the last 4 days of the sprint. Of the 3 students who were classified into this cluster, one continued this behavior into sprint 3, while the other 2 were classified into the Poor and Inconsistent clusters.

Inconsistent: 13 of the 54 students in the online class were classified into this cluster in sprint 2. Of which, a majority of them, 6 continued similar behavior in sprint 3 as well. 4 students in this cluster displayed an improvement and moved into the consistent cluster, while the remaining 3 displayed behavior of last minute activity by being classified into the FinalWeek cluster (Figure 25).

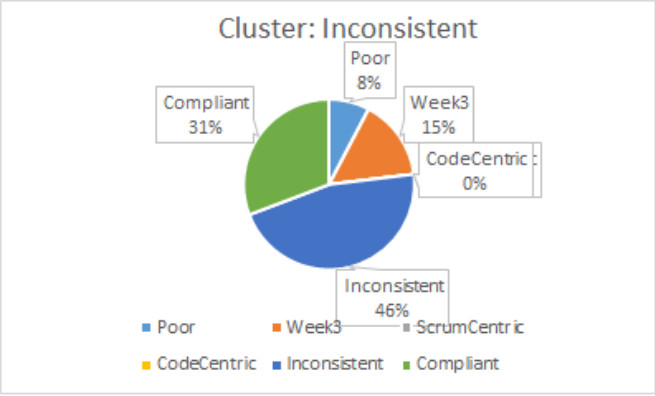

Figure 25: Migration Analysis: Inconsistent Cluster: Online Class

ScrumCentric: Out of the 12 students who were classified as displaying more Taiga activity in sprint 2, 5 students continued the same in sprint 3. 2 students from this cluster moved into the FinalWeek cluster and 3 students were classified into the Inconsistent cluster. It is encouraging to note that 2 students from this cluster eventually moved into the compliant cluster in sprint 3.

CodeCentric: Similar to the online class, only a few students were observed to be displaying significant GitHub activity in comparison to their Taiga activity. All the 3 students from this cluster wound up into the Inconsistent cluster in sprint 3.

Compliant: Of the 10 students who displayed consistent activity on both the tools in sprint 2, a majority, 6 of them continued the same into sprint 3. However all the remaining 4 students were classified as being inconsistent in Week2.5 (Figure 26)

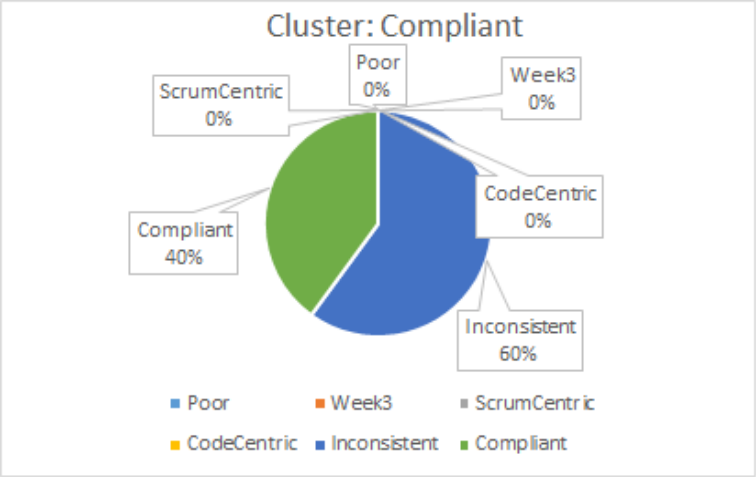

Figure 26: Migration Analysis: Compliant Cluster: Online Class

An interesting pattern observed in the face-to-face class is the factor of consistency. A sizeable number of students who start out as being classified compliant continue that behavior into subsequent sprints while students in the other cluster do not show signs of significant improvement with the exception of the “poor” cluster. This can be attributed to the fact that several students admit to not being able to take off as expected in sprint 1 and end up being classified as being poor with regard to compliance.

The number of students in each cluster remains fairly constant post the second sprint. This however is in contrast to the behavior observed in the online class. Part of the reason can be attributed to the online class having shorter (2 week) sprints. Data of student activity on the online tools was collected starting sprint 2. Based on the results of the migration analysis, we can observe that there are a few patterns that stand out. Particularly, the students whose behavior remains static (poor, consistent or inconsistent throughout all the sprints), students who display an incline (move from the poor/inconsistent clusters to the consistent cluster), decline (students who start out being consistent but eventually display a dip in performance), students whose activity on one tool outweighs their contribution on another (CodeCentric or ScrumCentric through all the sprints). These patterns incidentally are similar to the patterns observed in the analysis of the graded scores dataset from the 2014 and 2015 classes. This finding validates the patterns observed in the initial analysis and lends weight to the presumption that there are distinct, repeatable patterns of student behavior observed in a class over multiple years. Providing instructors with the knowledge of these patterns of student behavior can assist them in tailoring the course to suit the needs of these clusters of students and also paying closer attention to students who may be clustered as being "poor" or "inconsistent" in regard to their compliance to the learning protocol.

The experiment of applying the K-means clustering algorithm on the dataset of daily student activity is overall successful with stable clusters observed in both versions of the course. However, this daily activity dataset comprises only of student contribution data on the scrum board and source code control tool and does not include the student's activity on other tools (scrum meetings, continuous integration etc.). Clustering student activity datasets that includes all their contribution activity can help identify interesting patterns. (e.g. students who only contribute on the "easy tools"- scrum meetings, scrum

board and not on their source code, continuous integration tools). This limitation implies that the experiments conducted in this research do not entirely account for all contributions made by a student in the course. Irrespective of this limitation, this thesis validates the hypothesis that student activity patterns can be observed using unsupervised machine learning algorithms and future instances of the experiment can include student contribution on more tools.

In conclusion, subjecting the dataset of daily student activity on online tools to an unsupervised machine learning algorithm (K-means clustering) generates stable and consistent clusters in two versions (face-to-face, online) of the CST 316 course. "Migration Analysis" of these clusters provides insight into student behavior transition through the course of the semester. Understanding these migration patterns are crucial to identifying factors that influence these migration patterns. Instructors can utilize this information to make targeted and timely interventions for those group of students who may be in most need of it.

The next chapter reviews the problem statement, experiments conducted and details the conclusions of this thesis work. Using these conclusions it also sets context for future scope of this research project.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 CONCLUSIONS

This research is based on the premise that a project-based course supplemented by a system of frequent, formative feedback provides students in the software engineering discipline with a well-rounded learning experience. Continuous assessment is an excellent medium for delivering frequent, formative feedback and in this research we evaluate novel methods of performing continuous assessment in an agile learning, project-based course. An objective of this research then is to provide continuous assessment to students in order to help them develop a rhythm of consistent working in exchange for the inherent, deadline driven approach. To that end, the first part of this research work explores the use of visual reports and notifications as a means of providing continuous assessment and to measure the impact of these visual reports on student's compliance to the learning protocol. The second half of this research focuses on the use of learning analytics and machine learning algorithms to identify patterns of student activity in order to help instructors make targeted and timely interventions. All experiments in this research are conducted in an agile learning, project-based platform: The Software Enterprise.

Students in two undergraduate and one graduate software engineering courses in the Software Enterprise are provided with visual reports and notifications of their project activity via a dashboard. Students in the CST 316 course of 2015 were provided with visual reports in their final sprint while students in the graduate class of 2015 and the junior class of 2016 were provided these reports into their second sprint. Surveys were conducted at the end of each course to understand the impact of these visual reports in helping students comply with the expectations of the class. As detailed in chapter 5, more

than half the students in all the classes surveyed agree that they "liked the dashboard" and would like to see more tools incorporated into the dashboard. However, the goal of this thesis is to understand the impact of these visual reports on student's compliance to the learning protocol (RQ1, Chapter 3). Although the results of this research work are promising, the results obtained from three different experiments do not provide sufficient evidence to conclude that the visual reports have a significant impact on improving compliance to the learning protocol. The reasons for this can be attributed to several factors including students not trusting the data displayed on the dashboard, misinterpretation of the visual reports and inaccuracies observed in visual reports due to incorrect ids used by students while contributing on online tools. An enhanced version of the dashboard that accounts for student's contribution on multiple tools and addresses these issues can provide an holistic view of student's activity. The scope for future work in this research area is detailed in the next section.

In this research, we gather data of student sprint grades from two Software Enterprise courses and subject this dataset to an unsupervised machine learning algorithm (K-means clustering) in order to identify distinct patterns of student activity. The results discussed in chapter 5 describe how unique patterns are observed in student sprint grade scores and how each of these patterns are appropriately labeled based on the observed behavior in each cluster. Following the success of this experiment on student sprint grades, we gather a dataset of student activity on online agile tools daily and subject the same to the K-means clustering algorithm. Yet again, we successfully observe distinct patterns of student activity in both the online and face-to-face versions of the class. Each of these patterns are labeled to help instructors understand typical student characteristics displayed in class. These observations help us successfully conclude that the generic version of the K-means clustering algorithm can be utilized to

identify patterns of student behavior in a project-based course. These findings validate the hypothesis established in research question 2 (RQ2, Chapter 3). The conclusions of this research highlights the significance of learning analytics and machine learning algorithms in improving the student learning experience. This interdisciplinary research work is currently in its infancy and there exists tremendous scope for future work as detailed in the next section.

This chapter describes the conclusions that can be drawn from the results of the experiments conducted in this thesis. Although the findings of this research work are significant, there exists vast scope for future work in the same area. The next chapter explores the various limitations of this research work and describes alternatives for future improvements. It discusses the future scope in the area of continuous assessment using visual reports and learning analytics based on the results observed in this thesis.

7.2 FUTURE WORK

This section explores the scope of future work in the area of using visual reports as a form of providing continuous assessment and learning analytics to analyze and improve student behavior. The continuous assessment platform (CAP) serves as the latest version of the tool that provides visual reports and notifications to students in a project-based course via a dashboard. Currently, CAP supports the integration of only two online tools (scrum board, source code control), Students in a project-based course engage in a wide variety of activities including stand up meetings, continuous integration and so. The vision for the CAP is to support adaptive interfaces to any tool of a given category. The future scope for this research project targets integrations with tools that support Agile “daily standup” meetings, peer reviews, sprint retrospectives, and issue (defect) trackers. These tools can be integrated in future versions of CAssess to provide students an all-round view of their contribution on the project. Adding all these tools as

axes on the radial chart (Figure 6) can help students compare their activity against their peers on all the agile tools used in the project providing for an enhanced feedback mechanism. CAP currently employs a notification feature that is updated every week based on student's contribution against the learning protocol. In its current form, students need to log on the dashboard frequently in order to view these notifications. However we acknowledge that students need to be nudged into visiting the dashboard often and thus propose a system of email /text notifications. Although further research is required to understand effectiveness and impact of these email/ text notifications, this feature allows students to receive quasi real time feedback on their progress with regard to the learning protocol.

Learning analytics is an emerging area in the field of educational research with tremendous scope and potential to help instructors better understand student behavior and improve their learning experience. In this thesis, we use an unsupervised machine learning algorithm to identify patterns of student activity behavior based on their sprint grades and daily contribution on online agile tools. Although the findings of this research work are insightful there exists great room for future work. The logical next step is to use supervised machine learning algorithms that "learn" from these labelled clusters and utilize the same in order to make predictions of students in future instances of the course.

The ability to predict future behavior of students is beneficial to the instructor as this helps them perform continuous assessment by providing timely interventions to steer students in the right direction. The big picture envisioned in the long term plan of this research is to enable instructors to understand the size and nature of distinct clusters of student behavior displayed in any class. This understanding coupled with tailored course work provides students with a hands on, well-rounded, comprehensive

learning experience. Apart from making future predictions, the results of this thesis can also be used to understand patterns of team activity. Analyzing the transitional behavior of all members of a team during the period of the course provides insights into team dynamism. Understanding these dynamics helps instructors with better team formation skills which is a vital aspect of any project-based course. As mentioned earlier, this research project is currently in its early phase with remarkable scope for future work so as to make a significant impact in the field of educational research.

REFERENCES

Crisp, B.R. (2007). “Is it worth the effort? How feedback influences students’ subsequent submission of assessable work”, Assessment & Evaluation in Higher Education vol. 32, no. 5, pp. 571-581.

Trotter, E. (2006) “Student perceptions of continuous summative assessment”, Assessment & Evaluation in Higher Education, vol. 31, no. 5, pp. 505-521.

National Academy of Engineering. Educating the Engineer of 2020: Adapting Engineering Education to the New Century. The National Academies Press, Washington D.C., 2005.

Schwaber, K. & Beedle, M. (2001). Agile Software Development with Scrum, Prentice-Hall 2001.

Fowler, M., & Foemmel, M. (2006). Continuous integration. http://www.martinfowler.com/articles/continuousIntegration.html. Last accessed January 31, 2016.

Gary, K. “The Software Enterprise: Practicing Best Practices in Software Engineering Education”. (2008). The International Journal of Engineering Education Special Issue on Trends in Software Engineering Education, Volume 24, Number 4, pp. 705-716.

Gary, K., Lindquist, T., Bansal, S., and Ghazarian, A. (2013) A Project Spine for Software Engineering Curricular Design, Proceedings of the 26th Conference on Software Engineering Education & Training Co-located with ICSE 2013.

Kolb Experiential Learning: Experience as the Source of Learning and Development, Prentice-Hall, N.J. 1984.

Naif R. A, Davis C. A (2013). Learning Analytics and Formative Assessment to Provide Immediate Detailed Feedback Using a Student Centered Mobile Dashboard. 2013 Seventh International Conference on Next Generation Mobile Apps, Services and Technologies.

Oyelade, O. J, Oladipupo, O. O, Obagbuwa, I. C (2010) Application of k-Means Clustering algorithm for prediction of Students’ Academic Performance. International Journal of Computer Science and Information Security, IJCSIS, Vol. 7, No. 1, pp. 292-295, January 2010, USA

Bindiya M Varghese, Jose Tomy J, Unnikrishnan A and Poulose Jacob K, “Clustering Student Data to Characterize Performance Patterns” International Journal of Advanced Computer Science and Applications(IJACSA), Special Issue on Artificial Intelligence, 2011.

Arora R. K, Badal D. (2013). Evaluating Student’s Performance Using k-Means Clustering. International Journal of Computer Science and Technology. Vol.4, Issue 2, June 2013.

A. J. Stimpson and M. L. Cummings, "Assessing Intervention Timing in Computer-Based Education Using Machine Learning Algorithms," in IEEE Access, vol. 2, no. , pp. 78-87, 2014. doi: 10.1109/ACCESS.2014.2303071.

Ghiatău, R., Diac, G., and Curelaru, V. (2011) “Interaction between summative and formative in higher education assessment: students’ perception”, Social and Behavioral Sciences vol. 11, 2011, pp. 220–224.

J. Jamesmanoharan, S. H. Ganesh, M. L. P. Felciah and A. K. Shafreenbanu, "Discovering Students' Academic Performance Based on GPA Using K-Means Clustering Algorithm," Computing and Communication Technologies (WCCCT), 2014 World Congress on, Trichirappalli, 2014, pp. 200-202. doi: 10.1109/WCCCT.2014.75.

Petkovic D, Pérez M.S, Huang.S, Todtenhoefer. R, Okada. K, Arora. S, Sreenivasen. R, Flores.L, Dubey. S (2014). SETAP: Software Engineering Teamwork Assessment and Prediction Using Machine Learning. 2014 IEEE Frontiers in Education Conference (FIE) Proceedings, Madrid, 2014, pp. 1-8. doi: 10.1109/FIE.2014.7044199.

Ozaltin. N.O, Besterfield. M, Clark. R.M (2015). An Engineering Educator’s Decision Support Tool for Improving Innovation in Student Design Projects. Advances in Engineering Education . Summer2015, Vol. 4 Issue 4, p1-20. 20p.

M. Ross, C. A. Graves, J. W. Campbell and J. H. Kim, "Using Support Vector Machines to Classify Student Attentiveness for the Development of Personalized Learning Systems," Machine Learning and Applications (ICMLA), 2013 12th International Conference on, Miami, FL, 2013, pp. 325-328. doi: 10.1109/ICMLA.2013.66.

Huang. S, Fang. N (2013). Predicting student academic performance in an engineering dynamics course: A comparison of four types of predictive mathematical models. Computers & Education, Volume 61. doi:10.1016/j.compedu.2012.08.015.

A. M. de Morais, J. M. F. R. Araújo and E. B. Costa, "Monitoring student performance using data clustering and predictive modelling," 2014 IEEE Frontiers in Education Conference (FIE) Proceedings, Madrid, 2014, pp. 1-8. doi: 10.1109/FIE.2014.7044401.

L. S. Robles Pedrozo and M. Rodríguez-Artacho, "A cluster-based analisys to diagnose students' learning achievements," Global Engineering Education Conference (EDUCON), 2013 IEEE, Berlin, 2013, pp. 1118-1123. doi: 10.1109/EduCon.2013.6530248.

D. Lopez, J. R. Herrero, A. Pajuelo and A. Duran, "A proposal for continuous assessment at low cost," 2007 37th Annual Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports, Milwaukee, WI, 2007, pp. T1G-1-T1G-6. doi: 10.1109/FIE.2007.4417840.

M. Ross, C. A. Graves, J. W. Campbell and J. H. Kim, "Using Support Vector Machines to Classify Student Attentiveness for the Development of Personalized Learning Systems," Machine Learning and Applications (ICMLA), 2013 12th International Conference on, Miami, FL, 2013, pp. 325-328. doi: 10.1109/ICMLA.2013.66.

A. Bovo, S. Sanchez, O. Héguy and Y. Duthen, "Clustering moodle data as a tool for profiling students," e-Learning and e-Technologies in Education (ICEEE), 2013 Second International Conference on, Lodz, 2013, pp. 121-126. doi: 10.1109/ICeLeTE.2013.6644359.

C. L. Sa, D. H. b. Abang Ibrahim, E. Dahliana Hossain and M. bin Hossin, "Student performance analysis system (SPAS)," Information and Communication Technology for The Muslim World (ICT4M), 2014 The 5th International Conference on, Kuching, 2014, pp. 1-6. doi: 10.1109/ICT4M.2014.7020662.

G. Cheng and J. Chau, "Using an Automatic Approach to Classify Reflective Language Learning Skills of ESL Students," 2015 IEEE 15th International Conference on Advanced Learning Technologies, Hualien, 2015, pp. 375-379. doi: 10.1109/ICALT.2015.82.

Hartigan. J. A., Wong. M. A. (1979). Algorithm AS 136: A K-Means Clustering Algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) Vol. 28, No. 1 (1979), pp. 100-108.

Liaw. A, Wiener M (2002), Classification and Regression by randomForest.

F. Sarker, T. Tiropanis and H. C. Davis, "Linked data, data mining and external open data for better prediction of at-risk students," Control, Decision and Information Technologies (CoDIT), 2014 International Conference on, Metz, 2014, pp. 652-657. doi: 10.1109/CoDIT.2014.6996973.

C. Watson, F. W. B. Li and J. L. Godwin, "Predicting Performance in an Introductory Programming Course by Logging and Analyzing Student Programming Behavior," 2013 IEEE 13th International Conference on Advanced Learning Technologies, Beijing, 2013, pp. 319-323. doi: 10.1109/ICALT.2013.99.

F. S. Mohamad, M. Mumtazimah and S. A. Fadzli, "Integrating an e-learning model using IRT, Felder-Silverman and Neural Network approach," Informatics and Applications (ICIA),2013 Second International Conference on, Lodz, 2013, pp. 207-211. doi: 10.1109/ICoIA.2013.6650257.

H. R. Joseph, "Promoting education: A state of the art machine learning framework for feedback and monitoring E-Learning impact," Global Humanitarian Technology Conference - South Asia Satellite (GHTC-SAS), 2014 IEEE, Trivandrum, 2014, pp. 251-254. doi: 10.1109/GHTC-SAS.2014.6967592.

Y. Ding and M. Tang, "Effective assessment for full-time engineering postgraduate education," Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on, Xi'an, 2011, pp. 223-226. doi: 10.1109/ICCSN.2011.6013814.

Q. Zhou, Y. Zheng and C. Mou, "Predicting students' performance of an offline course from their online behaviors," Digital Information and Communication Technology and its Applications (DICTAP), 2015 Fifth International Conference on, Beirut, 2015, pp. 70-73. doi: 10.1109/DICTAP.2015.711317.

K. A. Gary and S. Xavier, "Agile learning through continuous assessment," Frontiers in Education Conference (FIE), 2015. 32614 2015. IEEE, El Paso, TX, 2015, pp. 1-4. doi: 10.1109/FIE.2015.7344278

Murphy. C (2016). Nicest,Unpublished report.