Work in Progress: A Student Activity Dashboard for Ensuring Project-based Learning Compliance

Suhas Xavier, Arizona State University
Christian Murphy, Arizona State University
Dr. Kevin A Gary, Arizona State University

Dr. Gary is an Associate Professor in the School of Computing, Informatics, and Decision Systems Engineering of the Ira A. Fulton Schools of Engineering at Arizona State University. His interests are broad and deep in all areas of the professorate: research, teaching, and service. His research interests are in software engineering education, web & mobile applications (specifically mHealth), agile methods, and open source. He recently spent a year on sabbatical as the Chief Architect in the Sheik Zayed Institute for Pediatric Surgical Innovation at the Children’s National Medical Center in Washington D.C. He has participated in projects like the open source Image-guided Surgical Toolkit (igstk.org) and microrobots for NOTES (minimally-invasive) surgery. In teaching and scholarship, Dr. Gary has created the Software Enterprise, an experiential learning model which blends multiple methods of delivery into one pedagogical model. The Software Enterprise has received significant support and generated several publications, and forms the core project spine within the BS in Software Engineering at ASU. Dr. Gary is the former Associate Chair for Computing in the Department of Engineering, and remains active curricular design and implementation.
Work-in-Progress: A Student Activity Dashboard for Ensuring Project-based Learning Compliance

Abstract

Project-based learning is an effective pedagogical tool for software engineering education. 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. However, teaching in a project-based environment presents several challenges. The focus of this paper is on the learning process that runs concurrently, and is co-dependent upon, a project’s software development lifecycle process. In this approach, student adherence to a set of learning activities in the conduct of a project is considered from a protocol compliance perspective, borrowing vocabulary from clinical domain. This WIP paper reports on the use of a web-based tool to encourage protocol compliance.

Introduction

Project-based learning is a pedagogical approach focusing on learning-in-context\(^1\). In software engineering education, project-based learning can be employed to great effect provided the focus remains on the application of concepts in the engineering context. However, it is challenging to formatively assess student projects. Formatively, students need work at a consistent rate to effectively utilize concepts. Then instructors then need to provide timely feedback so students understand how to change behaviors and correct the learning process. This research project provides a continuous, formative assessment platform for students working in a project-based context. Project teams utilize Agile methods, specifically Scrum, to conduct projects. Scrum is an excellent process model for this approach as it embodies a philosophy of establishing a rhythm to weekly and even daily activities. Of course a challenge in student teams is that the project is in the context of just one course out of many, and like most courses the project activity is deadline-driven. Yet in the context of a learning process, exclusively performing deadline-driven activities implies the learning process suffers, as compliance drops and contextualized learning benefits are limited. This is one of the prime motivations for a platform that performs continuous assessment, provides timely feedback and identifies compliance issues as soon as possible.

Continuous assessment interpreted from an agile software engineering perspective is closely related to the practice of continuous integration and testing\(^2\). This practice is fundamental to agile software engineering in that it supports transparency, visibility, continuous feedback, and adjustments based on empirical process control. This is crucial to agility; instead of a predictive process model that is inflexible to change, empirical control makes small adjustments to keep the process directionally correct\(^3\). Continuous assessment holds particular promise for project-based learning, as it provides consistent and timely feedback on student progress in a team setting.
Continuous assessment is also defined in educational research as the practice of performing frequent assessments in a course context. The practice has become popular in constructivist approaches to (project-based) learning, as frequent assessment provides a feedback mechanism ensuring students are properly aligned with a scaffold learning process. It remains an active area of research, with an ongoing debate surrounding the utility of formative versus summative continuous assessment. Continuous assessment in a university course tends to follow a scaffold learning process, albeit assessing at a much higher frequency than coarse-grained learning processes typical of a traditional, prescriptive university course. We see the agile concept of continuous integration and testing, realized in a software technology platform, as a direct means to perform continuous assessment.

The specific innovation evaluated in this project is the use of visualizations and notifications to ensure student compliance with the learning process. Students can view their progress via online tools on a daily basis and get regular notifications if they fall behind expected progress. Consistent ignorance of these notifications can be brought to the attention of the instructor who can provide the required intervention and support. This paper will report on the construction of a pair of tools, one to easily provision team projects, and the other to create continuous feedback. Preliminary results are based on pilot studies in two project-based classes, one for 41 junior-level software engineering students, and a second for 90 graduate students in a first semester, master’s level software engineering course. The study evaluates the impact of these tools to provide formative feedback on student progress on team projects.

Background

The need for a continuous assessment platform was driven by the need to provide quasi-real time feedback for students in project-based courses. In the Software Enterprise at Arizona State University, a project-based curriculum is offered to undergraduate and graduate software engineering students. In a typical project experience, students are grouped into teams, each working on building a software project by incorporating the principles of Agile. A course project is typically divided into 4-5 sprints spanning 3 weeks each. The requirements for this projected are accumulated into a product backlog created through a planning process. During each sprint, the team identifies a set of user stories from the product backlog and adds them to the current sprint backlog. Teams then identify tasks to do to meet the requirements, and assign each task to a team member. Teams use a number of tools, most importantly a source code control tool (Git/GitHub) and a Scrumboard (Scrumwise or Taiga). Additionally, students are expected to partake in regular standup meetings (3 per week), where the minutes are logged on a Google site.

Students in these courses receive a project grade after every sprint based on the frequency and impact of their process activity (PA), code activity (CA), and application of software engineering concepts or modules (M). PA is measured by scrumboard activity and standup meetings; CA by the amount of impactful code written in contribution to the project; and M by evidence-based reflection. The task of evaluating and providing feedback on student behavior in a sprint is time intensive for the instructor, especially if the size of the class is large. Compounding this is that in such an open-ended paradigm, students require guidance and scaffolding to the project-based learning process. In our earlier work we suggested agile tools can provide the same type of transparency to student projects that they do for industry projects, and this approach might lead
to better formative feedback by itself. Yet we found this approach did not scale; it is prohibitive for an instructor to provide anything more than ad hoc formative feedback at irregular intervals, having detrimental effects on student learning. In the spirit of agility, we would like to drive the formative feedback period to zero, and provide directed behavior modifications required to improve the learning process. We have begun constructing and piloting tools that provide such availability at a potentially scalable level.

The Continuous Assessment Platform

The objective of the Continuous Assessment Platform (CAP) is to gauge project activity and provide real time, formative feedback to enhance learning experience and ensure compliance in project-based courses. The initial version of the CAP platform supports the integration of 3 tools: GitHub for source code management, Scrumwise and Taiga for Scrumboards. These tools were chosen as they provide the most important collaborative functionality for supporting software products and organizing a team’s work. The current platform is a web-based integration platform (Figure 1) composed of two subsystems, each with its own user-facing components. The primary component, CAssess (short for Continuous Assessment), 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. In this section we briefly describe each of these components and how they integrate to provide feedback and ensure learning compliance.

![The Continuous Assessment Platform](image)
Nicest

Nicest is a tool integration platform for educators. Nicest 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 control), Taiga (agile scrumboard), and CAssess for individual or team projects. Recipes are organized as a wizard user experience; in the “Code Project” recipe shown below, the first two steps relate to course and student selection, the following steps each relate to a specific tool to configure, and the final step shows a confirmation of all selections made and allows for project to be created.

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 (described next) so that tool 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.

CAssess

CAssess does the heavy lifting of collecting student activity data from the various provisioned tools that it was told about by Nicest. It 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 Git Hub (Figure 3), a peer comparison radial chart (Figure 4) and a notification widget with directed feedback.
Once Nicest 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 agile tools. CAssess is 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 and Scrumwise) on a daily basis and stores this data into a local database. Data obtained from the Scrumboard (Scrumwise and Taiga) includes the number of tasks assigned to a project member, the corresponding status and the remaining hours for the completion of each task. CAssess uses Git Inspector$^{10}$, an online statistical analysis tool for Git Hub repositories. The statistics returned includes the number of commits made by each author, the lines of code added, stability (a measure in percentage of how much a user’s code survived) and other useful information.
At the end of every week, CAssess analyzes a student’s activity for the time period against rubrics defined by the instructor. For example, students in junior-level project course are told they must perform 3 significant code commits per week, and work on at least 2 tasks on the Scrumboard. Based on these rubrics defined and data collected by CAssess, it computes a score for each student and generates a notification or a call to action based on their activity over the week. A radar chart (Figure 4) 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.

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 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.

Preliminary Results

The CAP was piloted in 2 courses in calendar year 2015. Initial student feedback on the platform-based on informal surveys in both courses was positive. In the junior project-based course in Spring 2015, CAssess was deployed only for the last sprint of the semester. 22 of 41 students responded to an optional online anonymous survey. 15 of the respondents indicated the tool helped them understand team progress on the scrumboard, while 1 indicated it was not helpful (6 neutral responses). 10 respondents indicated the code activity visualizations helped, while 8 said it did not (4 neutral). 7 respondents said the radar chart helped them understand their progress, 5 said it did not, and a surprisingly high number (10) remained neutral. An in-class discussion revealed a lack of familiarity with radar charts as the cause for this high number.

In the Fall semester 2015, CAP was again utilized, this time for a larger and longer pilot course enrolling 90 Master’s students organized into 19 project teams working on a 9-week project. This time the platform was introduced in week 2 of the project and made available for the remainder of the 9-week period. At the conclusion of the course students were given a course survey that included a multi-part 5-response Likert scale (“Strongly Disagree”, to “Strongly Agree”) question on CAssess. 36 of 56 respondents indicated they checked CAssess frequently (7 did not, 13 neutral); 33 respondents indicated they would like to see more tools integrated on the platform (6 did not, 17 neutral). The results of this survey are summarized in Table 1.
Table 1. Graduate course survey utilizing CAAssess

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

The results overall are positive, though these are clearly informal surveys in a course context. Our future evaluation plan considers more rigorous forms of evaluation of the tools for this project. For example, we have not yet completed an analysis of the log information verifying the frequency that students visit the tool as opposed to relying on self-reported survey data.

Again we would like to stress that CAAssess is a formative feedback platform encouraging learning protocol compliance, not an automated grading platform. The nature of project-based learning and the complexity of measuring team interactions makes it difficult to consider what an automated grading platform would look like. Further, in our view, automated systems can be quite rigid, and students quickly infer the scoring rules and attempt to “game the system.” In fact with CAAssess we have had to remind students that it is not a grading platform but there as a tool to see how they are doing at a glance, and attempts to game the platform (repeated commits of redundant code, “fake” activity on scrumboard tasks, etc.) only fool themselves into thinking they are meeting the requirements of the course, and do not positively impact their grades. The grading (summative assessment) approach for the projects in the course is evidence-based as described in the Background section.

Future Work

The vision for the CAP is to support adaptive interfaces to any tool of a given category. For example, two different scrumboard tools were integrated to demonstrate that either tool might be used to measure a software development team’s progress on requirements and development tasks. For the purpose of this research however, it is not important how many tools exist in a given category, but how many different tool categories need to be integrated in order to get a comprehensive picture of individual and team activity on a software development project. To that end, our project plan targets integrations with tools support Agile “daily standup” meetings, peer reviews, sprint retrospectives, and issue (defect) trackers.

For Nicest, there are two long-term goals. The first is to allow Nicest to not only setup projects, but to be able to update projects and manage teams after provisioning. The second is to abstract
the steps in a setup recipe so that Nicest can have a plugin to create recipes, where creation could be as simple as select/drag and drop of the steps that an instructor would like to use.

For CAAssess, work is in progress to incorporate a peer review tool and to incorporate this score as an axis on the radial chart. Further, we acknowledge that students may need to be nudged into visiting the dashboard frequently so we propose email notifications as a reminder to encourage students to use the dashboard more often. Finally, as we continue to accumulate student project activity data, we are beginning to apply data mining and machine learning techniques to the data to understand good and bad patterns of compliance behavior, and see if we can design interventions aimed at early identification and correction of deficient student behaviors.

References