A Data Analytics Approach to a Computer Science Senior Capstone Project Management Tool

Dan Tappan
Department of Computer Science, Eastern Washington University, Cheney, WA, USA

Abstract - *Managing students in computer science senior capstone courses is comparable to herding cats. Many students can not or will not perform their tasks, and they go to great lengths to hide this behavior from their teammates and the instructor. This poster introduces a proof-of-concept web-based management tool that helps all stakeholders track activities, report on progress, and identify issues and concerns before they become problems, as well as reflect on the development process from an educational perspective.*

Keywords: capstone, project management, data analytics

1 Introduction

Software engineering has a bad reputation of being far too similar to the age-old joke from the public domain in Figure 1. Despite endless warnings to students in their senior capstone course in computer science that they need to take the development process seriously, the outcome almost inevitably resembles "The Cartoon" to a significant degree. A big part of disciplined development on a reasonably complex real-world project, which is indeed the experience that the students are supposed to be gaining, is to learn how to iteratively make a small plan, execute it, and verify that the actual results reasonably match the expected results. If they do not match, then some remedial action is necessary before continuing. However, this process only works if students have all three of these elements, continuously apply them in a serious, disciplined manner, and honestly assess and report their performance to their teammates, their project sponsors, and the course coordinator. Needless to say, typical team dynamics result in one or more students not carrying their weight and trying to hide this behavior.

This poster showcases a prototype web-based tracking system that helps students account for their activities and those of their teammates in an informative, intuitive way that is not especially onerous. Nobody enjoys generating status reports, but they are a necessary evil in a field where anarchy and disorder are the norm, even among professionals. Students, left to their own devices, generally fare far worse.

This experiment followed eight teams totaling 30 students as each team worked on its own independent project over 23 weeks covering two academic quarters. The development methodology was Agile, which offers considerable freedom, but it also demands a reasonable level of maturity and discipline. The intent was for students to be able to partition their work into bite-sized activities that aligned well with Agile thinking, doing, and verifying. Status reports helped students identity gaps and disconnects in the plans and their execution, as well as monitor the behavior of themselves and their teammates. They also provided a rich opportunity for students to reflect on the process to learn from it, instead of fixating on the product, which is really not the true purpose of the course.

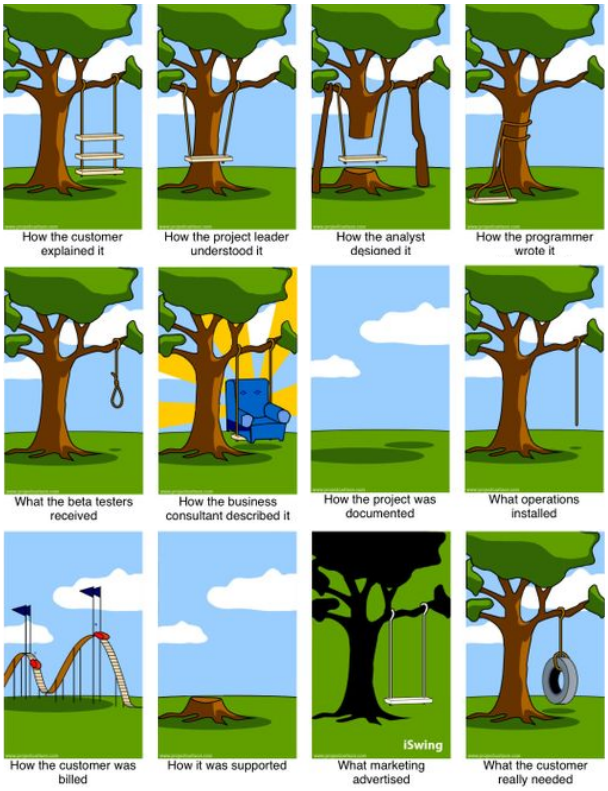

Figure 1: Software engineering reality

2 Report elicitation

The Agile sprint period was one week, Saturday to Friday, with the submission period on Friday. An earlier pilot test in the author's prerequisite software engineering course had successfully used a shorter period, so this approach is very flexible. There are two kinds of reports to submit.

2.1 Individual reports

Each team member completes their individual report on their own. It contains elements that will contribute to the

public and private summaries in the next section. The first section is public and elicits the progress that a student has made. It starts with basic time-keeping of hours of effort per day. Students' grades are not directly tied to these numbers, but because there is an average expectation over the project, students in the past had vastly inflated their numbers. The second section, also public, elicits new activities that were started, open activities that were continued from previous sprints, activities that were abandoned, and those that were transferred to other teammates or shared. Each automatically receives its own unique reference number that is retained throughout the project. A student must briefly describe the activity, including an estimate on completion time for new and open ones.

The third section is private and consists of three subsections that evaluate the claims made by teammates in the last sprint. For each teammate, it summarizes the hours and the four categories of activities and requires the student to indicate whether they concur, do not concur, or are not sure. The latter two require a brief explanation. It also asks whether performance is meeting expectations. Finally, there is a field for general comments.

2.2 Team report

The entire team completes the single team report together. It relates to the general experience of the team as a whole over the last sprint, not to specific activities of individual members. The reflective framework first asks the team to articulate briefly for each of these questions which aspects were both the easiest and hardest:

- *understand*: comprehending what needs to be done;
- *approach*: planning to solve it;
- *solve*: implementing the actual solution;
- *evaluate*: demonstrating that the performance of the solution is consistent with the problem and everything else in the project.

It also requests an estimate on how far along the project is and whether this pace is on target to finish on time. Teams routinely neglect communication with the project sponsors, so the next question asks whether there was any, and if not, when the last contact occurred. The final question addresses whether there are any issues, concerns, or comments not captured elsewhere.

3 Report generation

Reports are HTML-based emails that go to all stakeholders.

3.1 Public reports

The public report goes to each member of each team and the project sponsors. It summarizes the hours in a variety of intuitive statistical ways and depicts trend information over the project to this point. It also organizes the four types of activities from all team members into a readable summary, also with trend information. Finally, it presents the contents of the team report.

3.2 Private reports

The private report goes to the coordinator only. It contains everything in the public report, as well as the private entries. As the coordinator has many teams and students to manage on a weekly basis, the results are presented in such a way that skimming it reveals whether deeper investigation is warranted. A colorful heat map and matrix of green, yellow, and red dots help cross-reference each team member with themselves and each other. There is also an automatic tie-in with GitHub to produce a graphical representation of activity in the code repository by individual and team.

4 Results and discussion

In its first deployment as a proof of concept, this approach has already far and away demonstrated that it expedites the processes of keeping track of teams and individuals. In fact, instructors scheduled to teach this course in the future are envious and want access to this tool. It is difficult to compare this group of students against past ones because each offering of the course involves different projects, but there have been far fewer cases of deceptive behavior, or they have not continued very far before being questioned. The students very quickly got a feel for how to articulate their progress and interpret that of their teammates, which has done wonders for managing expectations. Software development is not a constant, linear activity, and certainly not for students in an academic setting. They all have ups and downs, easy weeks and difficult ones, etc. As long as their teammates are informed of, understand, and accept this performance, then they are functioning effectively as a team. This approach had a notable effect on improved quality in the final products. Anecdotal evidence suggests that it has also improved the process of software development, and especially the behavior of the people involved, who are traditionally the most troublesome factor.

The data elicited throughout the process are both quantitative and qualitative. No amount of manipulation is going to entirely automate the process of making sense of them, but the expectation is that there are likely to be clear patterns related to various aspects of performance. Faculty who have previously taught any course already have a general feeling for when, where, why, and how certain things go right or wrong. This approach does not replace such wisdom, but it appears likely to be helpful in managing the large amount of activities and information from so many students on completely independent projects over a long period. Other analytical measures will likely be added over time to highlight areas of interest and concern, etc. in an intuitive manner that is easier for all stakeholders to identify, monitor, and resolve.