International Journal *of* Designs for Learning
2013 | Volume 4, Issue 1 | Pages 39-53

ITERATING OVER A METHOD AND TOOL TO FACILITATE EQUITABLE ASSESSMENT OF GROUP WORK

Micah Gideon Modell, *Indiana University*

As an instructor employing group projects, my students and I have been frustrated by my inability to allocate credit for individuals' contributions to a group's projects. This design case details my efforts to design a method of equitably grading group work and addressing student concerns with respect to distribution of effort and, in tandem, to develop a tool that implements a substantial portion of that method. The method asks students involved in group projects to report the contributions of group members, including themselves, on a weekly basis. The web-based tool reminds students via email to enter numbers or use sliders to represent effort. Reported values are interdependent, meaning a low contribution from one member must be balanced by high contributions from others.

As the sole designer and developer on this project, I found little distinction between design and development activities. The design of the method evolved rapidly as it met with the reality of the tool being developed to support it. While the tool was initially considered for summative assessment purposes, the result focuses on formative assessment capabilities. Conflicting goals resulted in a functional prototype that would serve me for testing acceptance of the method and usefulness of the data, but the tool itself would not evolve further. This prototype uncovered avenues for research and the second iteration begins my exploration of some of these questions while addressing weaknesses. The decisions in the next iteration will focus on implementation of the method and related research resulting in a product that is easier to work with.

Micah Gideon Modell is a Ph.D. candidate in Instructional Systems Technology at Indiana University and an Instructional Designer at Option Six, a Division of GP Strategies. His research interests include performance support tools and meaningful assessment.

INTRODUCTION

Disclaimer

This design case serves as a descriptive report on the inception, evolution, and iteration of the design for a method of self- and peer-assessment and the software that makes implementation of this method realistic. In rendering this written report, I have used my own recollections, version-control records, and student response data to support the reconstruction of the design process.

I will present the path leading to the designed artifact before showing and describing the artifact itself to enable the reader to reach the destination with me. Images and interactive media are interspersed throughout to provide the reader with visuals to accompany the rich text descriptions.

Context

As an Instructional Systems Technology Ph.D. Candidate with teaching experience and a background in computer programming and software architecture, I have sought out opportunities to teach design- and development-related courses. One of these opportunities involved assisting in the delivery of a game design course that split the class into teams that collaborated on a culminating instructional game design project. In the final week of the course, students submitted reflection papers both to help them synthesize their own understanding of the events of the semester and to provide the instructor with insight into the groups' functioning to assist in grading.

When I taught the course the following year, I learned that this approach presented difficulties, including:

- Any discrepancy boiled down to one student's word against another's. There was no clear path to an objective truth I might use in grading.
- If I marked a student down based upon this information, he or she would know there was a problem, but would have no opportunity to fix it. There was no warning and no possibility that students would learn from their mistakes.

- While bold students did speak to me about difficulties in working with group members, this was infrequent and they were reluctant. A student experiencing group conflict did not have an accepted mechanism for communicating such issues.
- In reading the reflection papers, I suspected a recency effect (Furnham, 2010): The papers only covered the final, most stressful and most recent weeks of the semester—a disservice to the student who works hard and contributes much for two thirds of the semester, but encounters problems toward the end—such that only the problems are remembered.

Reflection papers also posed a logistical problem: understanding the content, particularly with the widely variable writing styles and skill levels, can take a great deal of time.

The following semester, I was asked to teach two courses with which I had no previous contact. Both involved delivering highly technical content to non-technical audiences. This required significant instructional design work. I was also overwhelmed by the prospect of having nearly 30 students in one class. My previous maximum was 15, so I sought to reduce an intimidating grading workload. In the face of these challenges, I resolved to incorporate collaborative group projects in my courses for three reasons:

PEER-TO-PEER LEARNING As a Master's student of Instructional Design, Development and Evaluation, I learned much from completing group projects. I always had people I could ask to help me understand things I missed. I also learned through explaining course content to others that I felt I understood.

THE ZONE OF PROXIMAL DEVELOPMENT Vygotsky's (1978) writings on social learning and the Zone of Proximal Development resonated with me. I believed that if each student brought different skillsets to the class, he or she would constantly be working with a "more capable other," and therefore learning.

CORPORATE REALITY My experience in the corporate world indicated that it was likely most of my students would work in groups at some point in their careers—it would be valuable to offer them an opportunity to build and practice those skills.

Inspiration and Precedent

As part of my academic research, I analyzed an article by Tucker and Reynolds (2006) describing the use of peer-assessment in an architecture studio. They employed Internet-based assessments reported on a weekly basis by and about the members of each group. In this study, the instructors had students award their peers a percentage of a team grade. The students rated their peers on a Likert scale evaluation to mitigate the likelihood of peer over-marking—the tendency of students to award higher scores to peers than an instructor might. While few details of the assessment method or the instrument implementing it were included, two aspects of this study caught my attention:

1. Using a weekly survey would offer me a series of snapshots in time and mitigate any end of semester recency effect.
2. Allocating a percentage of a grade focuses raters on the activity and tasks being performed rather than their affinity for the subject of the rating, and implies a cost to peer over-marking, which is known to be a problem (Falchikov, 1986).

I previously worked with Red Hat, Inc. to develop their middleware training materials, and was familiar with the performance-based assessments that back up their respected certifications. Those who hold Red Hat certifications have not simply answered challenging multiple choice questions or provided a detailed description of how they would apply a taught solution in the face of a threat to the systems under their watch. Instead, they have demonstrated their abilities on live systems under time pressure. Their examiners verified that their solutions worked, rather than whether the recommended solution was applied. Therefore, I was acquainted with building elaborate systems to render a meaningful assessment.

In my career as a custom enterprise software developer, I worked with web technologies to build software to power video delivery systems, inventory and sales order management, and market analysis for thousands of users. I knew how these technologies worked and how to design them to be robust and secure. While these skills made this project possible, they would also hinder my progress.

Finally, as a student minoring in Human-Computer Interaction Design (HCI/d) in a program where third wave thinking was emphasized (Cockton, 2008), I was immersed in user-centered design and a concern for non-work life—in this case, my students' spare time. As a software designer, I respect physical constraints and empirically demonstrated principles, but I also recognize that the context of use is important. The convenience of web technology has critical benefits and costs: I could make my system accessible from virtually anywhere, but my system would have steady competition for my students' time and attention, since so many other mobile applications are available.

THE PROBLEM

How can instructors equitably assign grades to individuals engaging in group projects?

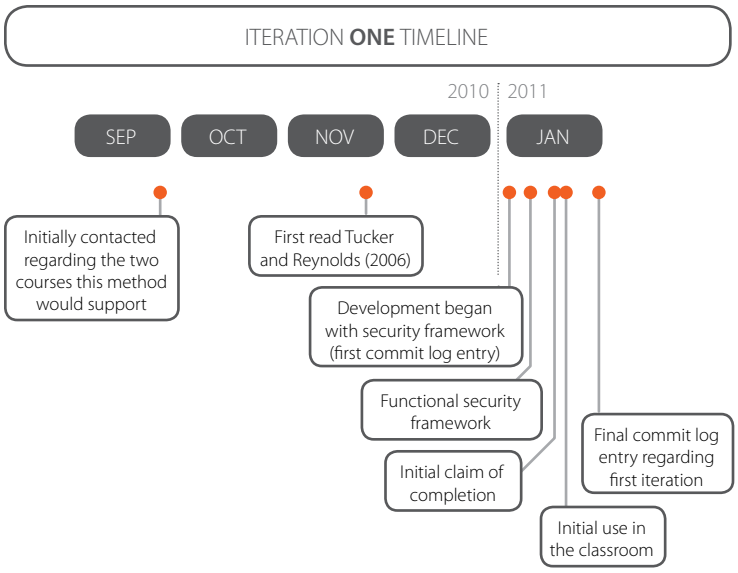

FIGURE 1. Timeline for the design and development of the first iteration.

THE SOLUTION CONCEPT

I ask students involved in group projects to use the tool to report the contributions of their group's members, including themselves, on a weekly basis. I implemented this method in a web-based environment, sending email reminders to students to log in and modify a set of numbers or sliders to represent effort. Reported values are interdependent, as are group members, meaning that lowering the contribution of one member requires an equivalent increase in effort on the part of others. The tool also makes the data available to instructors for both formative and summative assessment purposes.

EMERGENT DESIGN REQUIREMENTS

To support my teaching, my primary requirement was that I needed to develop a method of self- and peer-assessment supported by a tool into which students could enter their data weekly. I would subsequently use this data to calculate a modifier for each student's final grade to reallocate points from those who idled to those who paid for it in extra effort.

I asked students to evaluate the contributions of their teammates with whom they were actively working. However, there were restrictions to this method:

1. This is sensitive information and, to have any possibility of reliability in this reporting, students had to feel that their reports would not be made available to classmates.
2. The process could not constitute a significant imposition upon students' time. If the tool was difficult to use or if I was asking for too much of their time, they simply would not use it. Additionally, if I asked students to complete this task outside of class hours, there was a strong possibility my students would forget or refuse to do it.

My work on both the method and the tool had to be completed on a much tighter timeline than I was used to. The design of the two courses in which I would first use this method began four months prior to their delivery, and I discovered the influential Tucker and Reynolds (2006) article two months prior to delivery. Unfortunately, as I was also designing and developing the two course experiences that this method would support, I was not able to devote effort to this task until three weeks before classes began. By comparison, my past software development projects tended to run three to nine months from start to finish. This was intimidating (see Figure 1 and 7).

In addition to the legitimate demands of the method, I had another desire for the software. One of the two courses it would support was computer programming for educators. I intended to use the software as an authentic demonstration of the skills students were developing. Specifically, I intended to package a portion of the source code and provide it to my students to use to meet one of the objectives of the course—demonstrating the ability to include and use a library of functions from a third party., This also dictated another ill-fated decision—because the course would use the PHP programming language, I locked my own system into using it, too. For further information about the technology tools I used, links are included in the Technology References section following the Bibliographic References.

THE DESIGN PROCESS

To understand the design process, it is important to understand that there were significant delays to devoting effort, as I had to design and develop the two entire course experiences in which this method would be used (see Figure 1 and 7):

FOUR MONTHS PRIOR TO DELIVERY I began designing the two courses in which I would first use this method, along with the supporting tool.

TWO MONTHS PRIOR TO DELIVERY I discovered the influential Tucker and Reynolds (2006) article that would inform the design of the method.

SIX WEEKS PRIOR TO DELIVERY I began development of course materials for the two courses.

THREE WEEKS PRIOR TO DELIVERY I began designing the method and developing the tool for self- and peer-assessment.

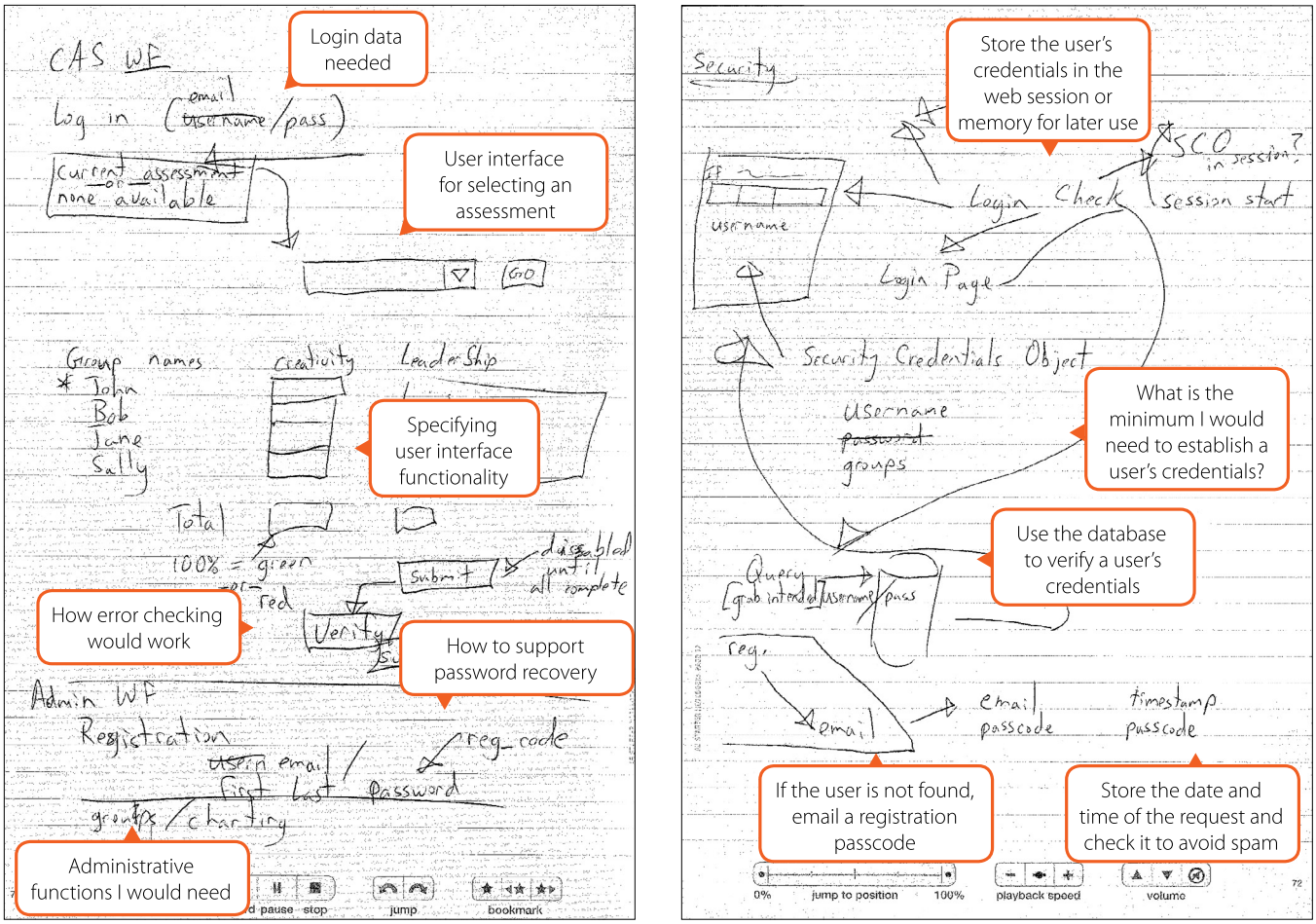

FIGURE 2. Rough initial sketches of database structures, relationships and user interfaces.

Within a few weeks of reading the Tucker and Reynolds (2006) article, I began to envision a web-based system whereby students could log in and allocate effort amongst teammates. Instead of allocating a grade, I wanted to make the link to grading less direct and focus instead on an individual's contribution to the functioning of the team. My initial design efforts consisted of searching for an existing and available implementation of the method I envisioned, or something close enough that I could use. I did not expect to locate such a system, since none had been indicated by the literature—and I was not surprised. I have since identified some existing systems, although none of these appear to be available to me (Fermelis, Tucker, & Palmer, 2008; Willey & Gardner, 2008; Wu, Chanda, & Willison, 2010). However, the searching and discussions with instructional consultants enabled me to sufficiently solidify the concept to describe it in later searches and conversations.

Recognizing that I would have to build the system myself, I had to suspend my efforts and simply research it in my spare time while developing the content and the environment students would experience in my two classes.

When I returned to my efforts, I first created rough sketches (see Figure 2) of database structures, relationships and user interfaces. These sketches were originally captured with audio using a LiveScribe pen, but the original recording has been lost.

I moved into development very quickly and, from then on, there was no distinction between design and development. As I was the sole participant in the project, communications between the designer and the developer were both instantaneous and unambiguous. As the initial design met with reality, I was forced to adapt. A brief sketch occasionally accompanied this, but more often the only permanent records were the check-in messages I recorded in Subversion and, subsequently Git, my source control systems. These show that many of my design decisions could more accurately be described as "what actually worked," as indicated by: "Login finally works! I guess that PHP requires the actual class [a programming construct] rather than simply the interface. Pretty lame, but what can you do?" (1/5/2011). I must also note that this commit message was incorrect as the system capability to accept user registrations was still incomplete and would remain so until 1/9/2011.

Tools

In addition to design sketches drawn with pen and paper, I used digital tools in my design and development process. I used the Adobe Dreamweaver integrated development environment (IDE), as it is a standards-compliant tool used by professionals, and is provided by my institution. I intended to ask my students to use this tool due to its availability and name recognition, and therefore I needed to familiarize myself with its capabilities.

My selection of the open source LAMP (Linux + Apache + MySQL + PHP) stack offered me significant benefits. I could legally download, install, and run the full versions of the server software at no cost to myself. I would also use the https standard for secure web communications, as it would both enhance the security and confidentiality of my students' data, and make the level of security visible—both in the form of "https" and, in many web browsers, other visual indications of security. Additionally, because of the vibrant open source communities that support these tools, there were many resources available via web search when I encountered issues. From a design and development perspective, PHP offered me rapid prototyping and testing capabilities—updating my application's logic was as simple as modifying and saving the file. No compilation or deployment was required.

The final relevant tool was the use of version control. As soon as I configured the basic server software, I installed the Subversion source control system and began checking in my work any time I made and verified any changes. Upon check-in, the source control system looks for and records only what has changed from one version to the next. As a result, every time I broke things in a way that confused me or would take me a long time to repair, I could simply check out a previous version and start again from a known working state. This saved me considerable time.

FLESHING OUT THE DESIGN

Security

Following in the footsteps of Tucker and Reynolds (2006), a web application seemed a natural fit for many reasons. As mentioned earlier, availability on a heavily connected campus would be a big benefit. Existing as a web application also meant I could provide a dynamic and responsive interface that could validate data before allowing it to be submitted. I spent only a small amount of time trying to find usable frameworks for things like security or data access because time was so short, and my past experience with new frameworks indicated that, while adopting one might make some tasks easier, I would likely incur a significant time cost up front as I learned how to use it. In my role as an enterprise software developer, I was once tasked with integrating a new security framework that unexpectedly consumed two months of my time, damaged my own reputation for reliability and cost me a raise. Additionally, the few PHP libraries I had worked with in the past offered only small benefits and, while some of the features I included might indicate otherwise, I did not expect to require sophisticated functionality. Therefore, I quickly decided to begin by building an authentication package to enable logging in and out. This was also driven by the desire to have such a library for my students to integrate as part of the computer programming for educators course.

The authentication system allows students to register without any intervention on the part of the instructor. They simply enter their email address—which served as their username—and then press the "Submit" button to have a password-set link emailed to them. Clicking the link would allow them to set their real name and their new password to complete registration. This elegant solution also served as a secure password reset mechanism if students forgot the password they'd used. Most importantly, I would never have to intervene, and since passwords were encrypted before storage in the database, I could honestly tell my students that no one else would have access to their password. As an added bonus, multiple email addresses could be attached to a single user.

This was where my enterprise software development experience inflicted some damage. I spent nearly a week building and testing this authentication system that I had designed, while I could easily have simplified the system and entered student data myself during the first class. I sacrificed my own development and testing time for other functionality. To be fair, however, I believe the introduction to the system using a clean and professional automated system engendered student confidence in the system and the method.

After packaging up all the security files for distribution, I was ready to move on to the rest of the system.

I was learning through painful experience just how few of the features I had learned to expect from other programming languages were present in PHP, and how many new ones it offered. One such "feature" is PHP's tendency to hide any problems it encounters. Programming logic authored by humans is inevitably flawed in the first version and developers expect to spend time understanding and tracking down the solutions to error messages that pop up. As a web technology, PHP strives to never let an end user see any error; it was designed to conceal any problems that arise and keep running without reporting a problem as well as it can. This means that bugs often do not show up until long after their introduction; this makes their identification very complicated and time-consuming. More information about this issue can be found online by searching for "PHP error reporting problem."

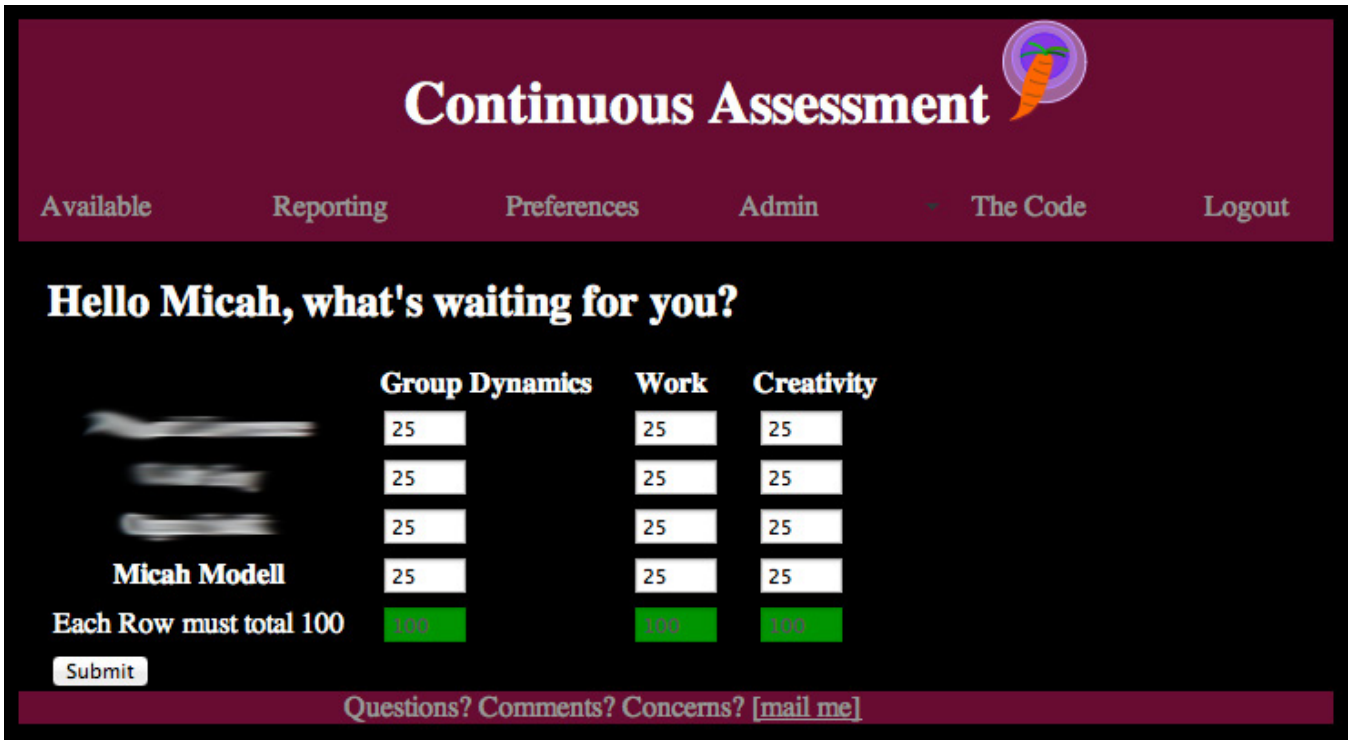

FIGURE 3. Weekly assessment reporting screen.

Having no frameworks to work with meant that I had to handle everything myself. Database access is especially difficult, because the structure of SQL language used for working with databases is very different from PHP. This meant I regularly had to shift my thought process. I tried to build some tools myself, but others have spent years perfecting such tools and I didn't get very far in those few weeks. I did get it all to work, though.

What was the Primary Use Case?

Upon reflection, I realized that I still had to determine exactly what data I should capture. Initially I intended to simply ask for students to report on the effort put in by each member of the group, but some questions arose as the tool began to take shape: Is the type of effort each student puts into group projects the same? How were students to evaluate the efforts of their peers? Wasn't it my duty to guide these evaluations in some way? If students are in the class because they don't yet know the content and need both exposure and practice, could they all be expected to contribute an equal share toward the graded product?

I decided to divide the contributions into three categories: work, creativity, and group dynamics. Each student would allocate contributions amongst their teammates, including themselves, each week (see Figure 3) using the following descriptions:

> **WORK** This refers to the "heavy lifting" or activities that directly result in a graded deliverable. This might include typing up a paper, painting a picture, or performing a dance routine.
>
> **CREATIVITY** This is the generation of ideas that help move the group forward. Perhaps a student can figure out which mathematical formula to use but not the mechanics (the work. Or maybe a student can envision and describe an appropriate graphic design solution but cannot draw a straight line.
>
> **GROUP DYNAMICS** This acknowledges that you can still help your team even if you don't yet understand the content being delivered. This would include making sure to show up on time for team meetings, encouraging teammates and displaying a positive attitude. This might be the person who brings the donuts or coffee to keep everyone going strong.

These categories were derived from my own experiences of group work, not from literature. Originally, I used the term "leadership" instead of "group dynamics," but I feared this might lead to members trying to delegate everything in hopes of earning high scores here. I therefore replaced it with "group dynamics," as I perceived it to cover a broader range of positive behaviors with less potential for misinterpretation. While they might be flawed, potentially unreliable from one rater to the next and they do not satisfy my academic colleagues, I should note that none of my students ever questioned me on their meaning after an initial explanation.

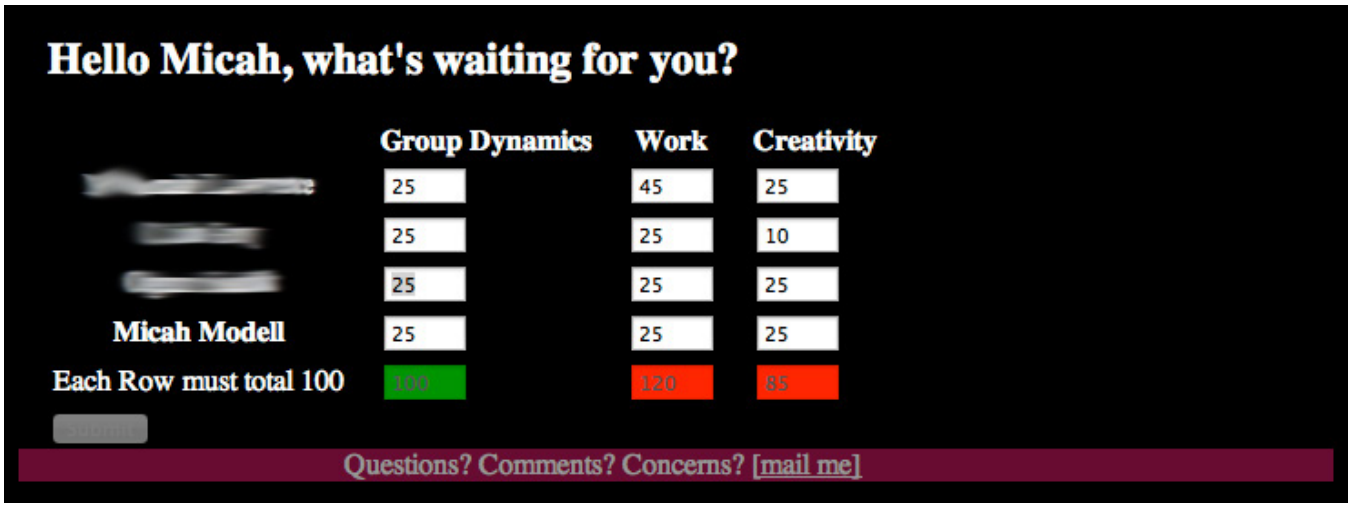

FIGURE 4. Weekly assessment error checking screen.

I decided to present these categories crossed with users in a grid format (see Figure 3) for students to enter their weekly reports. I initialized everyone with a contribution of 25 points in each category, assuming that under ideal conditions, everyone would contribute equally. I also showed the total for each column and indicated that it must equal 100 points to be accepted by the system.

Making it Easy to Use

Ease and perceived speed of use was always on my mind as I built the underlying functionality, but as my confidence increased, I spent more time thinking about the student experience. In line with my HCI/d training, I worked to minimize the method's intrusion on their spare time, setting a goal of requiring no more than five minutes for completion of the reporting task each week.

For example, while the math may not be terribly difficult, I wanted to make this as simple for my students as possible. Therefore, the system would mark any incorrect totals in red and disable the "Submit" button until the numbers were corrected (see Figure 4). This would guarantee that the data I collected would be valid while trying to make it easy for my students to complete the task. A side goal was to make it clear to my students every week that if one person did not do their fair share, others would have to work harder.

The entire system maintains a consistent look and feel with a burgundy-colored header and footer on a black background with the primary text in white. At the time, I felt that a black background might be more relaxing for the eye, as it wasn't as bright as the standard black text on white background. The landing page includes only a brief line of instruction, two text boxes and a submit button. Logging in brings the user to a personalized greeting with a listing of available assessments. Clicking a link in that listing launches the assessment itself (see Figure 5).

Assessments within the system included a start date and an end date (generally the start and end dates of the group project itself), and a start and end day of the week. If an assessment started on Friday and ended on Monday, it would only be accessible during this period each week, which offered some consistency in the temporal framing and also prevented any students from dwelling on this portion of the course. Assessments were only listed on the greeting page if they were actually open at the time and a link only appeared if the logged-in student had not yet completed the assessment. I didn't want students agonizing over their ratings and returning to make modifications.

Additionally, I took cues from Google in maintaining simplicity in the presentation. While I used Cascading Style Sheets (CSS) and JavaScript to render a dynamic and interactive interface, the site is primarily comprised of text. I used only one image throughout the site: a carrot as part of the logo. One of the teams I participated in as an undergraduate used a carrot as a mascot because the alternative spelling, ***caret***,

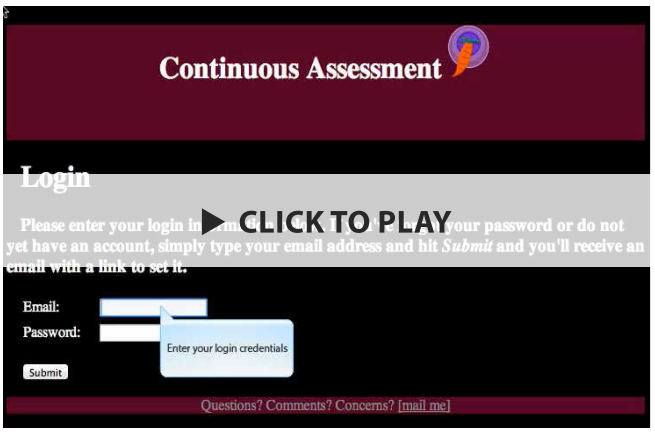

FIGURE 5. Screencast of the first iteration, including the user login, assessment selection, data entry, invalid data, and submission.

is represented by the symbol "^." This symbol represents exponentiation and is read as "to the power of." Ever since, I've used that symbol as a mascot and logo.

How Would I Use the Tool?

The above categories formed the beginning of an informal script I would use in training my students to use this tool; however, as the class approached, I began to recognize a problem. I did not feel that I could stand in front of my students and ask them to take on this additional task every week only for use by the instructor at the end of the semester. If I had this data streaming in throughout the semester and I might be able to detect and help to address obstacles a team might experience, shouldn't I? Furthermore, if I had data early in the semester that could negatively affect a student's grade and I didn't try to help correct it or at least warn the student, was I really doing my job? However, I also could not make the raw data available to individuals because I believed that their perception of confidentiality was crucial to collecting meaningful data.

It was clear to me that I would have to monitor the incoming data and set the expectation with my students that I would monitor it. I added the following content to my informal training script and included appropriate language in my syllabi accordingly:

> *"I will be monitoring these assessments throughout the semester to see how you're progressing. Sometimes you run into a problem that leaves you unable to contribute for a week—life happens. In these cases, you should talk to your team so they can pick up your tasks, and you'll return the favor at another time. I'm looking for trends and not spikes; you can have a bad week, but not a bad semester. If I see a problem, you'll hear from me and we can work together to try to resolve the issue. However, if the issue continues, I reserve the right, as indicated in your syllabus, to reallocate points to those who had to do more than their share of the work."*

The use of this data in a formative fashion implied that I would have to build myself tools to make timely analysis feasible. By specifying that I'd be looking for trends rather than spikes, I would have more time to develop these analysis tools. A few weeks into the semester, I used the RGraph JavaScript charting library to build graphs upon which the collected data was plotted (see Figure 6). RGraph is a highly customizable library that uses pure HTML5 and JavaScript, which meant a lightweight solution (i.e., quick download) that required no additional browser plug-ins or special software.

It was while building this graph that I ran into significant difficulties in working with dates using PHP. Math using dates can be very difficult, especially when the dates are stored as milliseconds after an arbitrary date. Ultimately, I only built

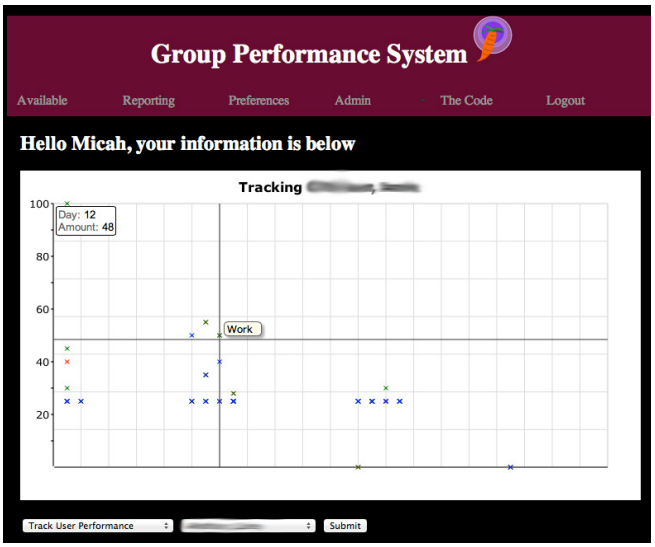

FIGURE 6. Assessment data for an individual user using RGraph.

a single, simple graph showing all the data reported for an individual. This graph was not pretty and the data points seemed to cluster strangely, but it was good enough for me to use.

Concerns After Launch

This system relied upon my students completing their assessments on a weekly basis and, despite these completions being a requirement of the courses, I did not build in any penalties for students failing to submit these reports, nor did I have any simple mechanism for determining those who had not complied. While some sort of punishment was technically feasible, this was not a path I wanted to travel. Instead, I have already described some of my efforts to make it as simple and minimally invasive as possible. However, as the semester continued, I had a distressing feeling that, while students didn't mind completing the assessments, many of them probably forgot about it altogether. I should note that all data indicated that students were performing their duty and I attribute my unease to two thought processes:

1. **PHP LANGUAGE COMPLEXITIES** I did not entirely trust that my PHP algorithms were functioning as I expected.
2. **DISSIPATION OF A NOVELTY EFFECT** I believed that students might be completing their weekly reports because of the newness of the technique and the tool. Therefore, I worried that participation might drop off over time as the novelty wore off (this was seen with the use of clickers in the classroom by Landaverde, 2012).

Enhancement

The final enhancement to my system was the ability to send reminders to students via email so as not to ask them to remember yet another assignment competing for their time. This took the form of a protected page I could access that would show me all the students who had not yet completed their weekly assessments. I could select which students to contact, customize a form email that included a link into the system, and send the email out. Each of these students would then simply have to click the link in the email and a few clicks later, they would be finished for the week.

EXPERIENCES WITH THE FIRST ITERATION

Report From the Field

Having applied this method for three semesters, I have a few observations regarding use. Most important was my belief that the method rendered useful data. I reached out to a number of students over that year and a half via email when the collected data gave me cause for concern. I received a variety of responses:

- Email conversations discussing the situation
- Face-to-face meetings with the students to discuss the situation
- No direct response at all, but instead the data showed that perceptions of their performance changed abruptly in the weeks that followed

However, I never became comfortable making assumptions as to the meaning of the data or what was happening in a particular group.

Students seemed to receive the system well and I never received any negative feedback about it. I feared that the reminder emails might seem intrusive or annoying to students, but no one ever indicated this to be the case. Furthermore, in the last two semesters of its use I obtained consent from 51 students to use their data for research purposes and analysis showed that they completed 89.2% of their weekly reports ,which I view as quite high.

From a support perspective, it was satisfying to know that students did not identify any reproducible flaws. I don't recall ever having been asked for help using the system beyond the initial introduction and training I provided in class, and while a few students indicated having received spurious reminder emails (i.e., after they had already completed their assessments), this error was never reproduced and was not raised more than once. I suspect that user error (on my part or on the part of the students) was probably the cause.

I must also note that I have discussed and presented this method and the accompanying software in a variety of venues and this has resulted in instructors asking if they may use the software in their own teaching. Unfortunately, the first version was not built to support multiple instructors.

Moving Forward

While the software implementation met my needs, I never extended its feature set or modified the method while I taught those courses. This was partly because I found PHP to be difficult to work with and I had little interest in spending more time with it. I found both the method and the tool to be valuable and worthy of further research, but the tool was clearly the work of a single instructor attempting to meet the needs of his own teaching under a compressed timeframe. It was affected by my competing goals (instructor performance support tool vs. packaged source code to serve as content for the class itself), the growing familiarity with the PHP programming language, and prior development experience (prototype for use by 30-40 students at a time vs. an enterprise grade, massively multi-user, high-performance system). I felt the method could benefit from additional features, and further research was required to analyze the meaning of the data collected in implementation. As my time with these classes drew to a close, I began to consider the following areas of exploration:

- What does the data mean with respect to group behaviors? Is it possible to know, from analysis of the data, what was happening in a given group?
- What set of categories/behaviors would provide the most meaningful and reliable insight into the groups' behaviors?
- How could I make this easier to use for students?
- How could I offer support for other instructors (i.e., other than myself)?
- How could I provide greater value for an instructor?

In developing the first question into a draft of my dissertation proposal, I resolved to embark on a second version of the software.

A FULL REWRITE

A more fundamental issue underlay all of these areas of interest: I had to somehow shed my reluctance to engage further with the system as a designer and developer. The chosen language made it difficult to extend or modify the existing system and this was exacerbated by the fact that I never built any automated tests of the functionality. This meant that my confidence was lowered by the fear that any enhancement might introduce a failure into the existing functionality—and I didn't have mechanisms in place to help me find the failure, nor did I have any interest in writing them because I still did not know of any good tools for doing so.

Instead, I began researching alternative platforms. I found that the discussions of the platforms I was familiar with were cast in terms of how they compared to Ruby on Rails.

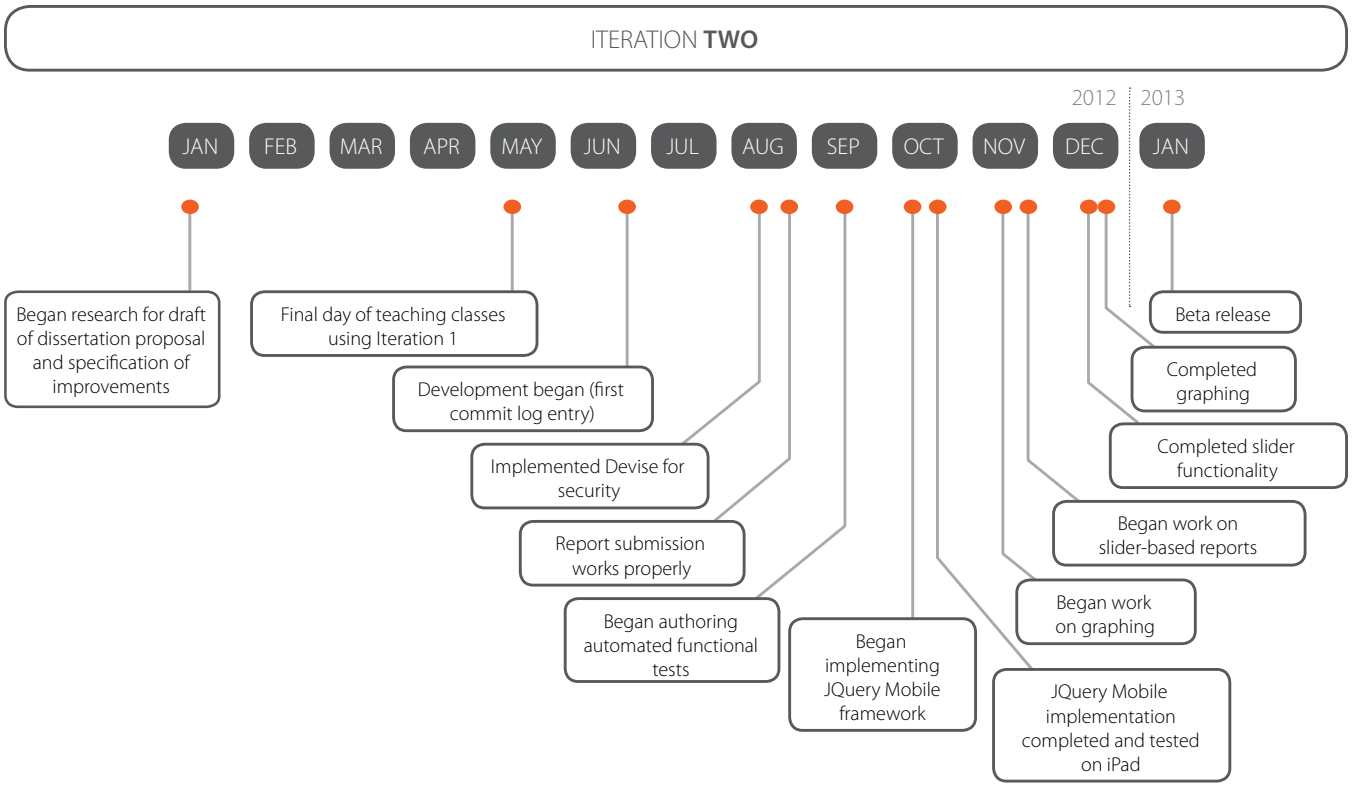

FIGURE 7. Timeline for the design and development of the second iteration.

I had never worked with the Ruby programming language or the relatively new Rails framework, but I'd always heard positive things; some research showed that this platform had cultivated an avid community with numerous tutorials, a thriving community of plugin libraries and active forums offering free support. This would be my opportunity to test it out (see Figure 7).

Moving to a new and unknown (to me) language meant I would have to start from scratch developmentally—this gave me the freedom to reassess all my previous design decisions. I was happy to find that I did not want to remove any of the existing features and that each was worth the time required to rebuild it. I also found that this new platform made it much easier to move forward quickly. While many of the platforms I had worked with in the past were either designed for speed, some form of conceptual purity (e.g., the "object-oriented way") or rapid development, Ruby on Rails' design decisions include "How can we make this intuitive to the developer?" While the date math would still prove confusing, at least the language would no longer get in the way.

This switch to a new and unfamiliar platform came at a significant cost, as I had to learn not only the basic mechanics, but I had to adjust to the Rails mindset. For example, Rails automatically looks for certain types of files in specific locations, which can be very confusing if you are unaware of these assumptions. While the language allows you to do things however you want (e.g., naming conventions, file organization), it makes your life much easier if you do things the "Rails way." Fortunately, the "Rails way" has been developed over time by many experienced software designers and architects, so I could adjust to it both logically and easily. Furthermore, the basic system included many invaluable helper tools and utilities, such as a local server for trial purposes.

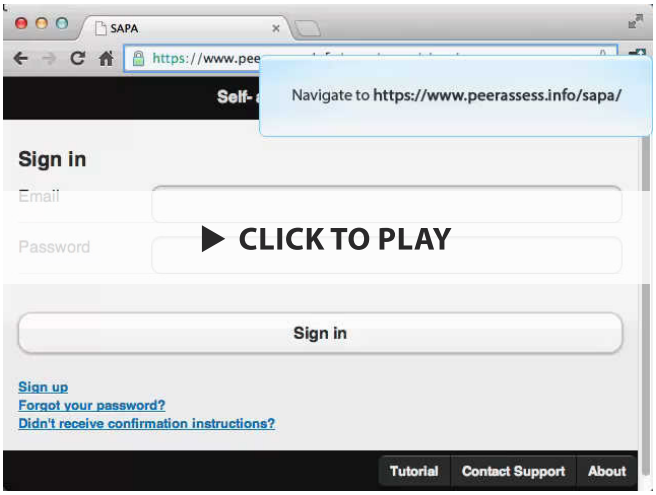

FIGURE 8. Registration screencast, including receipt of the registration email and clicking the activation link.

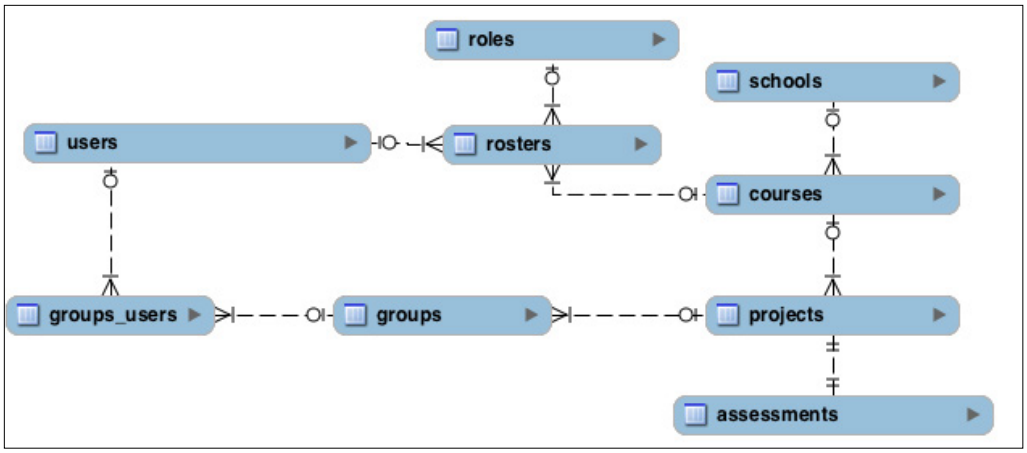

FIGURE 9. Partial model of the revised data structure.

Going beyond the platform's features, the community has built and packaged numerous extensions (appropriately called "gems") that enabled me to rapidly add common functionality. While authoring the security framework myself in PHP took roughly a week, with Ruby on Rails, I was able to use a popular gem called Devise to set up email registration, login, and more within an hour or two (see Figure 8).

With this iteration, I was determined to have automated tests in place to protect my users from my tinkering. I chose a framework called Cucumber, which made it intuitive and satisfying to write automated software tests. Cucumber enables you to describe what you expect your software to do, in plain English statements that you bind to testing code. The tests can be run automatically and doing so renders reports that make it clear where problems lie. These tests verified that everything functioned as I expected. More importantly, it gave me the confidence I needed to be creative and daring without fear of back sliding because I could re-run them at any time to re-check the functionality and make sure I didn't break anything.

Support for Instructors

While the original version of the system used a simple organizational structure of individuals in groups attached to assessments, there was no concept of courses, schools or layers of administration. Users were either students or administrators; they could see nothing or everything—no in-between.

The new model (see Figure 9) is more complex, offering the ability to assign multiple assessed projects to a course, each with their own groups. An instructor can run multiple courses or many instructors could take part in a single course. I even have the ability to add new roles like Graduate Assistant or Teaching Assistant with the possibility of custom permissions. Finally, I am able to track the schools to which the courses are attached. The model is the result of many hours spent sketching possible layouts, testing them out in code and then revising them. It is a significant improvement and offers greater flexibility, but I am sure it won't be long before I must modify it to extend its capabilities.

The first iteration was really unfinished. Because I would be the only instructor using the tool, the administrative console was minimal, enabling me to edit groups and assessments graphically, but everything else required me to work directly in the database. Exacerbating the issue was the fact that my design of even these administrative tools was so shoddy that I often thought they were broken. They were not broken, they were just unintuitive. I wanted to ensure instructors could work with and maintain their own courses this time around and this, too, was achieved through a Ruby gem.

The ActiveAdmin gem automatically generates beautiful and useable administrative interfaces for Ruby constructs using three simple lines of code (see Figure 10). Furthermore, these sets of pages are highly customizable. It was so easy that I built a system administrative interface, as well as one that was customized for instructors. The latter was organized differently and limited access to those resources that should be available to the instructor (e.g., their own courses).

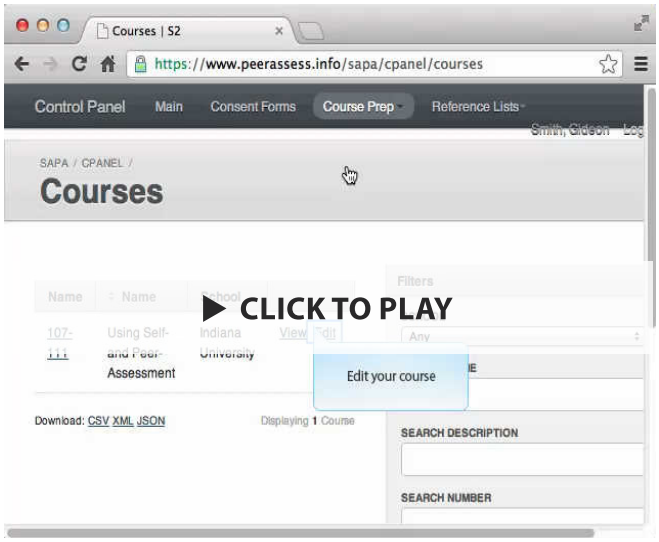

FIGURE 10. Course setup screencast, including a demonstration of how an instructor would edit their course, create group projects, create groups for the projects, and configure assessments for the projects.

In implementing the email reminder functionality, I chose to relieve instructors of the need to manually email students by automatically sending the reminders every evening at midnight. Initially this was achieved using the rufus-scheduler gem. This enabled me to schedule the execution of predefined tasks and it worked beautifully in the development environment. Unfortunately, the emails did not get sent when deployed to the web hosting environment I purchased for this purpose. I spent over a week debugging and searching for the cause of this problem on my own. I tested whether or not the system was capable of sending emails; I tested for flaws in my logic; I tested different time zone configurations. After exhausting each of these avenues, I began to suspect the problem was not in my code at all. Some Internet searching pointed to a known issue with rufus-scheduler in certain web-hosting environments. When executing properly, rufus-scheduler looked like an accidental result of poor programming— something the system was designed to search out and destroy automatically. Once I knew the root cause, it was a simple matter to switch over to the use of an alternative (cron, a standard but external Unix tool) used for scheduled executions.

Although a painful experience, the fact that none of it was caused by mistakes in my logic gave me confidence in the system's configuration and, more importantly, in my own code. It also exposed a deficiency with respect to time zones: my web hosting environment is in a different time zone, so midnight emails are sent at 1 a.m. from the perspective of my local time zone. While this is easy to adjust, it means that the system, as designed, is locked to a single time zone and it is not clear, from a usage perspective, how best to address this. It will have to wait for a future iteration.

Finally, instructors must be able to access their students' data for it to be useful to them, but I ran into difficulties using RGraph to meet my needs this time. RGraph's output was not consistent across platforms; when using the latest version of Google's Chrome browser on Apple's Mac OS X, I saw beautiful graphs with popups explaining each data point on a rollover. When using an older and more common version of Microsoft Internet Explorer on the Microsoft Windows operating system, I was unable to see any graphs at all. After some research, I identified the HighCharts graphing library that I determined to be a suitable alternative, particularly due to its compatibility with the popular JQuery JavaScript framework, which seems to have emerged as the de facto standard in web development. This library provides developers with a single common set of tools for building dynamic and responsive web pages and applications.

HighCharts' flexibility, combined with the ease of working with the data using Ruby, yielded a flexible system in which users can quickly and easily jump back and forth between different views or different users for comparison purposes. After the first chart was built, I could add new charts based

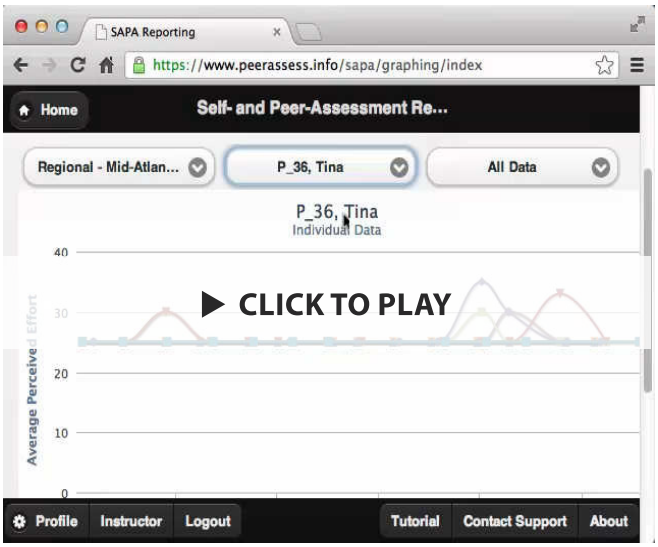

FIGURE 11. Graphing screencast, including individual and group reports and rollover information.

on different data sets within a matter of hours rather than days. As a result, the latest iteration includes multiple individual visualizations and a visualization for group data. The interface is also dynamic and responsive to user selections, loading users and groups after a course has been selected and allowing instructors to zoom in on data that may appear jumbled (see Figure 11).

To test the new graphing functionality, I needed realistic data and, under normal circumstances, I would have to fabricate it. This time, however, I was able to export, transform, and load the data from the first version into the new system. It was at this time that I calculated the response rate (89.2%) using the data from my first version and, with the new system, I was able to build this calculation and make it visible to instructors in less than 30 minutes.

Benefits for the Students

As I finished implementing and testing the core functionality replicated from the first version, I started focusing on ways to make it easy for students to provide the desired information. In addition to automating the reminder emails, I resolved to improve upon the consistent user interface with various help mechanisms, including video tutorials and a link to email support. In addition, having already gone through the process of converting my own personal dossier website to a mobile-friendly format using JQuery Mobile, I decided to do the same with this web application.

The JQuery Mobile framework is built on top of the JQuery JavaScript library (a characteristic shared with HighCharts, which simplified development). This conversion took no more than an hour and enables my students who might be checking their email on tablets or phones to click the link, log in, and complete their assessment with only a minimal

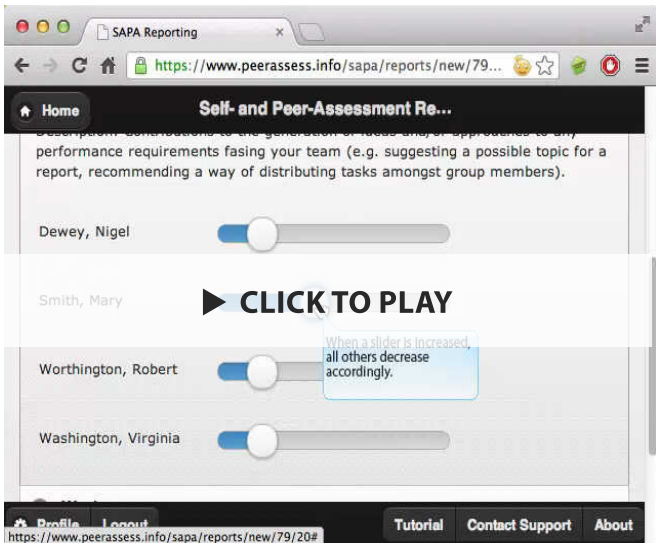

FIGURE 12. Slider-based reporting screencast, including logging in, selecting an assessment and moving the sliders to enter data. The fact the sliders in a behavior group are interdependent is highlighted.

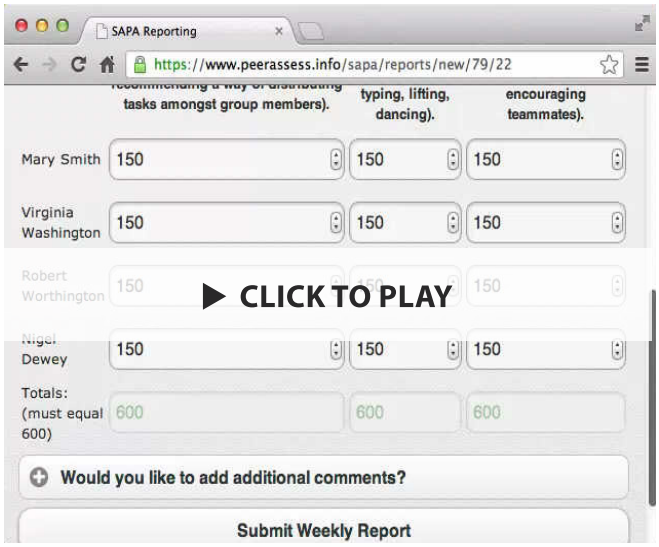

FIGURE 13. Data entry screencast, including user login, assessment selection, and entry of data, highlighting error feedback.

interruption to their day's activities. I also chose to use the JQuery Mobile default color scheme in response to a negative comment regarding the burgundy and black color scheme of the initial system.

I never liked the grid format for students to enter their weekly reports, and a few students voiced supporting statements (though never strenuously). Therefore, I solicited suggestions from students and colleagues studying HCI/d and investigated how they might be achieved. After completing the technically simpler grid-based report (for ease of testing and to make sure I had something which worked), I decided to build and offer a alternative interface option using sets of linked sliders (see Figure 12) to represent users within a behavior grouping. JQuery made it feasible to produce this effect, but after many days' effort, I still had a math error and I began to suspect that it was not anywhere in my code. This drove me to change the way my Ruby code operated.

Some students had requested the ability to provide written comments; this was also added to the new version of the system. However, this new space for optional comments is hidden inside a drawer that opens with a click (see Figure 12 and 13). I did not want students to feel compelled to complete this new item, because it could take significant time and defeat my drive for minimal impact.

According to Wheelan (2009), small groups are recommended to be between three and five members. Group function inversely correlates with group size, with three to four members experiencing the lowest levels of dysfunction and reaching the highest levels of development. Therefore, I set my target at four members and allowed a one-member buffer in either direction for flexibility. Based on this target group size, I determined that the sum of the reported values per category should be 600 (since it divides evenly by three, four, and five and offers room for adjustments). I initially set up the system to return an error if the numbers did not sum to 600. With the math error, I could not guarantee 600, but I could guarantee that students would focus instead on the visual sliders by simply removing all numbers from the display. Therefore, I changed the server logic to proportionately convert the inputs—whatever they were—to add up to 600. This enabled me to preserve the most important part—the relationship between the student contributions represented by the numbers.

Finally, when it was time to record the video tutorials, I used the latest version of Adobe Captivate because the ability to export to HTML5 would mean the tutorials would be available on the mobile platforms I was working hard to support. This effort also meant some reorganization and consolidation of menu items, as I captured the video at a small screen resolution of 640x480 to ensure it would be viewable on mobile platforms.

Research Support

As this system was being built, at least in part, to collect data for my dissertation, I built in some additional functionality to collect demographic data, periodic administration of follow-up questions after weekly reports, and the administration of consent forms. However, I strove to keep this as separate as possible from the core functionality so as not to intrude upon users.

Of these, the consent forms were the least straightforward. When should a student be asked to offer consent? How should the request be presented? Based partly upon recommendations from a more experienced researcher, I initially

built the ability to present actual PDFs with a checkbox to indicate consent every time a student registered with the system. However, I soon realized that some users might wish to use the tool with group projects not a part of research initiatives. Therefore I reworked the consent forms to be presented when a student first tries to access an assessment that requests consent. This is coupled with the ability for students to view and change their individual consent status at any time.

Another lingering issue was the most appropriate categories or behaviors for assessment. My research has not yet turned up a definitive set. As a result, I have built in the ability for instructors to create their own set, though this raises further questions, including: Does the selection of assessed behaviors and/or the sequence of their presentation in the interface affect students' understanding of group process?

SUMMARY AND CONCLUSIONS

This design case details my efforts to equitably grade group work and address student concerns with respect to distribution of effort. As both the sole designer and developer on the project, I found little distinction between design and development activities. In fact, the concept behind the design changed radically when the implementation reached a state where I could begin to envision its use. It became clear not only that the method offered data that could be used formatively in addition to the original summative intent, but also that recognition of these capabilities obligated me to take advantage of them and shift the design accordingly, leaving other portions unfinished.

Under tight time constraints with the first version, I built most of the system from scratch, as I had little time to find existing tools that might ease development. This was compounded by the fact that my enterprise software experiences led me to over-engineer the system, costing me precious development time. The design of the first iteration was further complicated by competing goals for the resulting code: Not only did I want to use the functionality of the system, but I intended to use the development time to gain a deeper understanding of the language I would be using to meet some of the content needs of the course. In the future, I would choose a language I already knew and loved and find an existing library for my students' use.

Through two iterations, I have maintained a consistent core set of functionality and a focus on minimizing an instructor's imposition on students' time. As an instructor and a designer, I kept the students' experiences in mind throughout the process. In some cases, this led to a productive paranoia that drove me to continually improve the method and the tool beyond the initial release. This could easily have resulted in instability and a confusing user experience, but it was tempered by a growing dislike for the PHP language and a desire never to touch it again.

The second iteration applied many of the lessons I learned with the first: I put more thought into the choice of tools and I allocated much more time to design and development. One important result is a codebase that I do not fear but actually enjoy working with and extending. The new system is mobile-accessible and more user-friendly. While the second iteration of the software has not yet been tested with students, my experience with the first system was positive and I have high hopes, even if I am already aware of existing issues (e.g., proper time zone support, assessed behaviors).

The technology decisions were tremendously important. After the first system met my minimum requirements for use, I was reluctant to make any further changes because I perceived it as fragile and I did not have adequate testing measures in place. With the latest iteration, I do not hesitate to jump in and start changing things, because it is both well organized and easy to test for unintended consequences.

Next Steps

The new version of this software will ideally be tested with a small class using formal groups. After making any updates implied by the beta test, this system will be used to support instructors in their teaching and to ask questions about the meaning of the collected data, the impact of the method on students' development of group work skills, and further refinement of the method.

ACKNOWLEDGEMENTS

I would like to thank Laura Chancey for her editorial skill, Shelley Fyman for her comments throughout, the reviewers for their valuable feedback, and my wife and daughter for bearing with me throughout.

REFERENCES

Cockton, G. (2008). Revisiting usability's three key principles. *Proceeding of the twenty-sixth annual CHI conference extended abstracts on Human factors in computing systems* (pp. 2473-2484). New York, NY: ACM Press. http://dx.doi.org/10.1145/1358628.1358704

Falchikov, N. (1986). Product comparisons and process benefits of collaborative peer group and self assessments. *Assessment and Evaluation in Higher Education, 11*(2), 146-166. http://dx.doi.org/10.1080/0260293860110206

Fermelis, J., Tucker, R., & Palmer, S. (2008). Self and peer assessment: Development of an online tool for team assignments in business communication and architecture. *Proceedings of the 2008 Association for Business Communication Annual Convention. Association for Business Communication*. Retrieved from http://businesscommunication.org/wp-content/uploads/2011/04/15ABC2008.pdf

Furnham, A. (2010). The robustness of the recency effect : Studies using legal evidence. *The Journal of General Psychology, 113*(4), 351–357.

Landaverde, L. (2012). Do clickers work? A study of the impact of the use of audience response systems for high school english language learners. (Master's thesis). Retrieved from http://csusm-dspace.calstate.edu/handle/10211.8/232

Tucker, R., & Reynolds, C. (2006). The impact of teaching models, group structures and assessment modes on cooperative learning in the student design studio. *Journal for Education in the Built Environment, 1*(2), 39–56.

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes.* Cambridge, MA: Harvard University Press.

Wheelan, S. (2009). Group size, group development, and group productivity. *Small Group Research, 40*(2), 247–262. http://dx.doi.org/10.1177/1046496408328703

Willey, K., & Gardner, A. (2008). Improvements in the self and peer assessment tool SPARK: Do they improve learning outcomes? *Proceedings of the ATN Assessment Conference.* Retrieved from http://ojs.ml.unisa.edu.au/index.php/atna/article/view/343/258

Wu, C., Chanda, E. K., & Willison, J. (2010). SPARKPLUS for self-and peer assessment on group-based honours' research projects. *Education Research Group of Adelaide (ERGA) Conference 2010: The Changing Face of Education.* Retrieved from http://digital.library.adelaide.edu.au/dspace/handle/2440/61612

TECHNOLOGY TOOLS

Adobe Captivate
http://www.adobe.com/products/captivate.html

ActiveAdmin
http://www.ActiveAdmin.info

Cucumber
https://github.com/cucumber/cucumber/wiki

Devise
https://github.com/plataformatec/devise

Git
http://www.git-scm.com

HighCharts
http://www.HighCharts.com

JQuery
http://www.JQuery.com

JQuery Mobile
http://www.JQueryMobile.com

PHP
http://www.PHP.net

RGraph
http://www.RGraph.net

Ruby on Rails
http://www.RubyOnRails.org

Ruby
http://www.Ruby-lang.org

Rufus-scheduler
https://github.com/jmettraux/rufus-scheduler

Subversion
http://Subversion.tigris.org