Using a Rubric-based assessment system to improve feedback and student performance in course-management systems

Bob Edmison[1] Stephen H. Edwards[2] Manuel A. Pérez-Quiñones[3]

Abstract – The importance of feedback to students on writing assignments has been well established, but the typical feedback students receive on assignments submitted via a course management system (CMS) is disconnected from the actual content of the student's work. Instructors have developed a variety of methods for providing feedback on electronically submitted assignments, but the bulk of these processes do not provide the feedback within the context of the original assignment. In this paper, we provide a review of the literature about the importance of feedback to the learning process. We review some of the existing tools that have sought to address the issue of contextual feedback. Finally, we describe a new markup system that is a plug-in to the Moodle CMS that uses rubrics as a framework for guiding the feedback process. This system provides a means of adding feedback within the context of an electronic submission, as well as capturing that feedback to be used by the instructor as an overview of the class' performance on the assignment.

Keywords: Computer-aided assessment, rubrics, contextualized feedback

INTRODUCTION

Contextualized feedback is the notion that feedback provided to students is not simply the text that an instructor writes to a student about some aspect of an assignment the student submits. It is also situated within the assignment itself, communicating information about the student's performance based upon its position with the document. In traditional paper-based systems, the "red pen" was used to make notations on student submissions. With the increased use of course management systems such as Moodle [12] Blackboard [8], and Sakai [30], there is now a disconnect between the submission and the feedback. The feedback is now often reported back to the student out of context. This removes some of the effectiveness of the feedback.

This research examines two particular questions:

- What ways exist to provide more effective feedback to students on assignments submitted electronically, and what impact this feedback has on student performance?
- What ways exist to provide feedback to instructors based on the student's performance on written assignments?

These questions are significant in that they will provide understanding about the importance of feedback to students as well as providing insight into how to provide meaningful feedback in situations, such as distance learning, where most interactions between student and instructor are mediated by some form of course managements system.

[1] Virginia Tech Department of Computer Science, Blacksburg, VA, kedmison@cs.vt.edu

[2] Virginia Tech Department of Computer Science, Blacksburg, VA, edwards@cs.vt.edu

[3] Virginia Tech Department of Computer Science, Blacksburg, VA, perez@cs.vt.edu

Additionally, answering these questions will provide understanding about how instructors can modify the class content, as well as the content delivery to provide a more effective learning experience. Before answering these questions, however, we must understand the work that has already been done in this area.

This paper will review the current understanding of the importance of feedback from a pedagogical perspective, situate feedback within a larger context of human-computer interaction (HCI) and educational theory, as well as discuss how feedback is handled in various implementations of web-learning systems. Finally, we will introduce a new system we have developed as an addition to the Moodle CMS to provide rubric-based assessment and feedback.

WHAT IS FEEDBACK?

The importance of providing feedback to students has been well established in educational research since the early 1960s. [See [4],[16],[25]] Feedback is defined generically as any device used to tell a learner whether their response is correct or not [21]. However, as Kulhavy points out, the mechanism for this importance has been open to debate. Many behaviorists argued the position of *feedback-as-reinforcement*. They argued that providing the correct response would reinforce the correct knowledge to the student, thereby improving performance. Secondly, Kulhavy cites a significant body of work that indicates that waiting a day or more before providing the feedback actually *increases* the retention of the provided guidance. So, waiting to tell the student why their answer is or is not correct greatly increases the likelihood that the information is correctly remembered in a later assessment. This is directly counter the typical reinforcement framework, which posits that the impact of the reinforcement decreases as it is removed from the initial event.

While Kulhavy provided a definition for feedback, Hattie and Timperley [17] examined what feedback means in terms of classroom interactions between teacher and student. They argue, "[F]eedback is conceptualized as information provided by an agent (e.g., teacher, peer, book, parent, self, experience) regarding aspects of one's performance or understanding." This new definition indicates feedback should be focused towards a student's performance of a task, or understanding of some concept. Hattie and Timperley go on to say that for feedback to have the most impact, it must occur within in an active learning context. Hattie and Timperley describe a model for effective feedback [17]. Their model identifies three specific questions that must be answered to insure that the feedback is effective, but, they note that the most important aspect of this framework is ensuring that the feedback presented is situated and in context to the learner's current level and in concert with the learner's goals. This is a key point to our argument, which we will revisit later.

Well-crafted feedback situates the learning within the knowledge creation process. The student knows precisely why they missed a question, and that feedback situates the student in the learning context so that they can better retain the new information later. Additionally, as Wang and Wu have shown [33], feedback that provides more elaboration increases the student's motivation, leading to increased student learning effectiveness. Without this situated feedback, the students are less motivated, and the learning less effective. Feedback becomes an activity in an active learning experience. The student receives feedback on an assignment, within the context of the assignment, and makes the connections between the feedback and the concepts. Also, the feedback can serve to provide insight to the instructor, allowing a view into the performance of the students beyond just the single data point encapsulated by the grade value. This is the motivation for one of our goals of making instructional feedback easier to provide for electronic submissions.

For the instructor, feedback is also important. Because learning is a process of knowledge construction, and because active learners are more successful at this than passive ones [17, 21], feedback provides another way for instructors to engage students. Additionally, in our work we are looking at ways of capturing the feedback that instructors provide to the students and using that as feedback that the instructors can use to modify the design of the course as it progresses. In the past, the instructor would have been able to get this type of information by hand-collating the comments he was making. The tool we are designing will capture this information as the comments are being added. Thus, the instructional design of the class can be modified based on data about student performance. We think this is a very powerful tool that speaks to making the online learning tools we have much more situated in the learning environment, for both the student and the instructor.

SITUATING FEEDBACK WITHIN HCI AND INSTRUCTIONAL THEORY

Merriam-Webster's dictionary defines "learning" as "modification of a behavioral tendency by experience (as exposure to conditioning)". An alternate definition is "knowledge or skill acquired by instruction or study". Both of these definitions imply active communication between the instructor and the learner. Current theories of learning make five assumptions, which will be familiar to those who study HCI [15, 22]:

- Learning is a process of knowledge construction, and it requires learners to be active designers rather than passive consumers.
- Learning is situated. People act until a breakdown occurs, at which point a coach can provide feedback to the learner identify the situation that caused the breakdown.
- Learning is knowledge-dependent. A person's existing knowledge is used as the basis for creating new knowledge.
- Learning is influenced by distributed cognition, whereby people learn that the skills and knowledge required to address a problem does not reside in one person, but is spread across may people.
- Learning is affected by motivation as much as by cognition. Learners must understand why it is important for them to learn and contribute.

These five points clearly show distinct parallels between HCI and learning theory. Given the historical roots of HCI in psychology [11], this stands to reason.

Löwgren and Stolterman argued that "designers", whether they be interaction designers or learning systems designers, must identify the "situation" of the design, as it indicates to the designer many of the constraints that will need to be addressed. For them, learning is a complex undertaking that must account for the situations and motivations of the learner and the environment in which the learning is to take place [31]. Our experience bears this out. Learning is influenced both by the internal motivations of the learner as well as the external situation wherein the learning occurs, and there is research to support the position [33]. Feedback is situated within the context of the content to which it references. Without that context, the utility of the feedback is diminished.

As Fischer and Scharff [14] describe, repurposing classroom content for a web-based course often does not result is a successful outcome because the context of the interaction is different. The same can be said of the feedback that is provided to students when course management systems are deployed. Moodle is a CMS that was designed from the outset to support the constructivist model and situated learning [12]. The idea behind Moodle was to build a system that would support the design of learning material in an approach that was pedagogically appropriate for web-based delivery. Content can be created to support knowledge construction by engaging the user to take an active role in the learning process, as well providing and environment that Lave and Wenger might describe as ``tuned to the situation'' [22]. Moodle provides the tools to create situated learning content, but neither Moodle nor any of the other CMS tools that we have surveyed extend the same situational awareness to the assessment and feedback portions of the systems. The result, as we have discussed, is that feedback is often provided out of context to the item to which it applies, such as with student-submitted files. Without this context, the effectiveness of the feedback provided is diminished. Our research is focused on placing feedback back into the situation context where it used to reside before the advent of course management systems. The HCI principles of situated actions and design as a social activity are central to this work.

CURRENT FEEDBACK IMPLEMENTATIONS WITHIN COMPUTER-ASSISTED LEARNING ENVIRONMENTS

Because of the nature of the field, computer science has always been on the forefront of using technology to teach the theory and practice of CS education. Over the years, a variety of systems have been developed that have focused on supporting the education CS students. These systems have taken a variety of forms, but we will focus on how a sampling of these systems have (or have not) dealt with providing feedback to students.

Systems supporting computer science education

Managing the workflow of a CS1 or CS2 class has many different needs [20, 23], including managing file access, accepting student file submissions, and compilation and execution of the student's source code submissions. A

variety of systems have been developed that manage portions of the class workflow process. Some, prior to the wide-spread adoption of the world wide web as an application framework, relied on a series of shell scripts to support compilation and testing [19]. Other systems automated the process of submission, but did not integrate the process of compiling and testing the student's assignments [10]. Luck and Joy developed one of the first systems to incorporate all of these requirements into a single system [24]. Their system was a client-server system that encompassed the requirements outlined above. It provided the grader weighted grading categories, and it provided a means to attach notes to the assignments, but the feedback was provided within the context of the actual source code.

Pardo [26] designed a system that took the ideas developed by Luck and Joy [24]and others, and developed them using an architecture he called a *multi-agent platform*. It addresses the requirements that are identified above, but it still lacks contextual, situated feedback. Assignments are assessed by category, with each category being weighted, but any comments that are generated as part of the assessment are not kept with the assignment. They are emailed to the student as a separate document. This clearly is not an ideal solution for the student. The first author of this paper participated in a course where feedback was provided in this fashion. Most of the comments were very general in nature, possibly due to the difficulty in associating the comments to specific items in the submission. In many instances where specific feedback was provided, it was difficult to match the comment to the location in the original document to which it referred. Both of these issues make it very difficult to establish the context of the feedback.

Blumenstein, et al [9] took a different direction at solving the automated assessment problem. While most systems developed previously were tied to particular language, the GAME system was designed to handle multiple languages. This system allowed students to submit multiple times, and would provide feedback about each submission. However, it did not provide the comments within the context of the original source code, so the student was still left to search out the relevant context.

The demand for students who have an understanding of test-driven development [see [6]] has created a new avenue of exploration for automated assessment. Assessing student projects created using test-driven development in existing automated assessment systems exposed several areas where these systems cannot handle the test-driven development scheme [13]. Two of these areas were the need for clear feedback to students on how they can improved the applications they were developing and the need for the students to see the value of the activities and feedback they were participating in. This addresses Hattie and Timperley's requirements of effective feedback.

Web-CAT [13] addresses these issues. Additionally, it provides two kinds of feedback not available from earlier systems. First, it provides early feedback to students about the performance of not only their application, but also their tests, encouraging the "write a test, write some code" mantra that supporters of test-driven development encourage. Secondly, it provides error messages inline to the student's source code. The result of changing paradigms, as well as supporting test-driven development within the Web-CAT tool was to show a significant increase in student performance, as well as an increase in the quality of the code the students were developing.

Systems supporting using rubrics as grading

Rubrics are another method for assessing assignments. Rubrics have been used in areas outside computer science education for many years. However, it is only recently that they have been applied to assessing programming exercises and other learning activities associated with CS education. Rubrics are useful in two ways [7]. First, effective rubrics require the instructor to enumerate exactly what the qualities and criteria are that the student is expected to achieve in each grading category. This provides students with a form of feedback even before the assignment is attempted. The second way rubrics are helpful is to provide a grading framework that can be applied each grader. This can insure that inter-rater reliability [18] is maintained.

Winters and Payne developed a system called Agar that was designed to provide a level of flexibility not previously seen in automated assessment systems. Winters argued that the "killer app" of automated assessment has not arisen because too often the tools that have been developed have required that the instructor adapt their teaching to the tool, rather than the tool being flexible enough to be modified to handle the different types of a activities that instructors need to offer [35].

The core of the Agar experience is developing a rubric to be used to evaluate the student's submissions. This enumerates the suite of evaluations that will be applied to the submission, as well as handle the states that occur

when a test in the rubric fails. The rubric also serves as a framework for the manual-grading task, where a grader can review the submission and add feedback comments and tie them to the categories identified in the rubric. Additionally, Agar mines the collected data about student data to develop a picture about student performance [34]. This provides quantified feedback to the instructor about overall class performance that heretofore was very difficult to collect. This allows the instructor to adjust the course work and assignments to address the needs of the students *during* the course, rather than waiting to the end for student evaluations.

Powell, et al [28] used rubrics as an assessment framework for a peer-review module for Moodle. In their system, instructors are provided the ability to assign students to provide peer-review to submissions provided by their classmates. The student annotates the submission outside the context of Moodle and then uploads the annotated file back to Moodle, filling in scoring and additional comments after the upload to Moodle. In this system, some feedback is provided in context to the submission, but other feedback, vis-à-vis the rubric, is still provided outside the submission.

While rubrics provide a useful framework for guiding the grading activity, there is also the danger, as Ahoniemi discovered [3], that graders will form ideas about what the grades should be for an assignment before they have completed the evaluation of the submission using the rubric as a benchmark. To mitigate this, Aloha allows the instructor to develop the rubric within the submission system, but the final scores are not assigned until the rubric is filled out for the submission. This prevents the grader from, in effect, prejudging the submission. However, Ahoniemi discovered providing similar feedback to each student continued to be problematic. In a course with several hundred students, it can be very challenging to provide the same feedback to each student and make it meaningful to each. Aloha has been updated in an attempt to address this shortcoming [2], with the results seemimg to indicate that the system has had some success in insuring the objectivity of the graders, though the quality of feedback the system elicits from the graders remains unclear [1].

As this research has shown, the use of rubrics in assessing CS assignments has improved inter-rater consistency, and thus the overall objectivity of the grading activity. However, creating rubrics and managing the comments that are most commonly associated with the rubric entries can be tedious. Auvinen, using Aloha as a base, has developed Rubyric [5], a system to manage to the creation of the rubric as well as a base set of comments. However, Rubyric, like Aloha before it, suffers from the same problem as other automated assessment systems, in that the feedback provided is not delivered within the context of the assignment to which it pertains. So, the graders may generate helpful comments, but the comments themselves are decoupled from the content that caused the grader to create them in the first place. This places an unnecessary barrier between the student and the full impact of the feedback.

Online writing assignment assessment and markup

To this point, most of the computer-assisted assessment systems we have reviewed have been targeted at computer science education. The tools act on programming activities, checking style, syntax, compilation and execution. But several tools and studies have looked at techniques that can be extended to other disciplines. Extended these assessment systems is important because of the importance of writing as a skill across all areas, not just computer science and engineering.

A study by Price in 1997 looked at a CS1 course taught in a distance-learning environment [29]. The course had an established method for submitting and grading paper assignments. The authors developed a system to convert the student submissions into Microsoft Word format, whereby three instructors used macros in the Word files to make corrections and provide feedback to the students. The feedback provided on paper submissions was then compared to that provided on electronic submissions. The study reported that the students and instructors felt that the quality of the feedback on the electronic submissions exceeded that of the paper submissions. Additionally, the time required to complete grading and return assignments was reduced.

However, as before, feedback in the system developed and studied by Price is captured out of context with the subject of the feedback. Popyack [27] argued that using pen-based input, such as that found in the electronic ink feature in the Portable Document Format from Adobe is an alternative that provides a familiar motif for graders while still maintaining the advantages of electronic markup, such as archiving comments, and the ease that larger classes and distance learning environments can be supported. Using digitizing tablets and Perl scripts to convert student submissions to PDF, graders add their comments within the PDF file, which is then stored back to their

course management system. The PDF specification provides a method to lock the file to prevent editing once the comments are added.

This system gets back to the "red pen"-style of markup that was common before electronic assignment submissions, providing feedback within the context of the assignment. However, it lacks three important features. First, it does not provide for commonly used comments to be recycled. Secondly, there is no way to automatically collect the comments into a single database and associate them with the original submission. Finally, because it is unclear from their report, it is difficult to determine if inter-rater reliability is increased, or at least, maintained by this design.

Turner, et al [32] observed that students conducting a peer-review session using a pen-and-paper method generated significantly more comments than when using a pen-based system on a tablet-PC or a point-and-click system on a desktop PC. Also, he noted that pen-and-paper reviews resulted in very detailed comments, such as crossed out letters within words. He reported that these types of edits appeared very seldom in the electronic review options, where the students seemed to focus more on the higher-level concepts because of the constraints of the review system. Contextually, the desktop review system used language to provide the reader with context of the comment, rather than "gestural" context such as an arrow or box indicting the item to which the comment corresponds. Thus, providing context in a desktop system would seem to rely on the proximity of the comment to the content.

A NEW RUBRIC-BASED ASSESSMENT SYSTEM

Moodle provides a robust plug-in API framework that allows third-party modules to extend the major functionality of the system. We have designed an extension module for the Moodle course management system to enhance student assessment. Our tool has two major components: a tool to create the rubrics that can be applied to an assignment, and a tool that actually applies the rubric the submissions uploaded by a student.

The rubric creation tool allows the creation of rubrics that are composed of a hierarchy of components. Categories are the major delimiters of the rubrics, and describe broad areas being assessed, such as "Grammar/Style", "Conceptual Clarity", etc. Within each category, the rubric can have one or more question assigned. A question takes the form of what a grader would ask himself while assessing the submission to determine the score the student gets for that portion of the assignment. An example of a "Grammar/Style" category question might be "Did the student proof read the submission for spelling and grammatically errors?"

Each question can be one of two types, a ratings question and a checklist question. Ratings questions allow the instructor to assign several ratings labels (i.e. "Excellent", "Good", "Fair", "Poor") to a question that can be assigned to the student's performance. Additionally, each rating level can have language assigned to describe what level of performance needs to be attained in order to achieve that level of assessment. Checklist questions have a list of criteria that the instructor has identified as being required for satisfying the requirements of the question. Each question has a total point value assigned to it. For ratings questions, each rating has a point value that is awarded when the rating is selected. Point values are not assigned to checklist items for checklist questions. Scoring for checklist questions is handled manually.

Creating the rubric is only part of the solution. The instructor also has to be able to apply the rubric to the student's submission. We have integrated the assessment mechanism into the grading subsystem of Moodle. Currently, rubric assessments can be applied to single-file assignment types. The system is able to accept submissions in several file formats, including plain text, HTML, Adobe PDF, and Microsoft Word DOC. The newer Microsoft DOCX format is partially supported. Currently, inline images and tables are not supported. The system converts the student submission into a format that allows the document to be marked up with grader-provided feedback comments. This is accomplished in the feedback view.

The submission view has a tabbed interface that provides three views of the student's document. First, it shows the submission with all of the comments that have been entered. The comments are shown in context to the original document. To enter a new comment, the grader simply clicks the line of text where the comment should apply. A box appears (see Figure 1) that allows the user to select the rubric category the comment applies too, as well as a comment type. The comment types are "Error", "Warning", and "Comment". These allow the grader to provide additionally context to the comments. Comments are color-coded based on their type, providing the feedback

additional context. Once the grader is finished entering the feedback, the comment is listed on the left side of the screen, and the line of the original document is highlighted to indicate that this line has feedback.

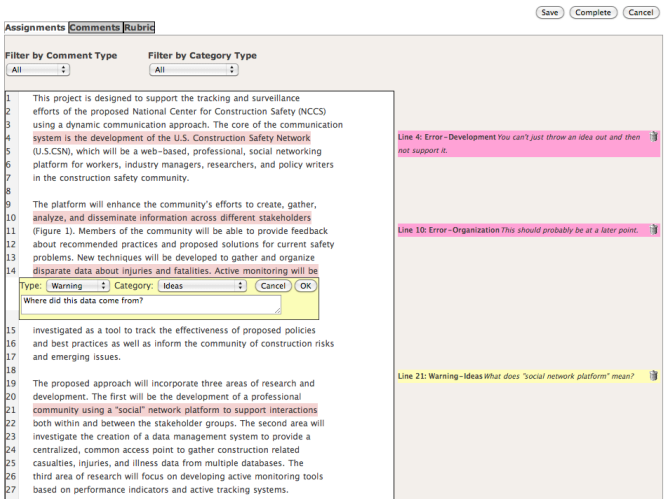

Figure 1 The Assignment view allows the grader to read the submission and add comments contextually to the body of the document.

The second tab, shown in Figure 2, summarizes the comments by the type of comment. Each comment is listed, along with a line number that is a hyperlink, which will take the user back to that line in the document. The third tab shows the rubric itself, and summarizes those comments that are associated with each rubric category and question (Figure 3). Also in the third tab, the grader can select the ratings for ratings type questions, as well as enter a final score for checklist-type questions. When the assessment is complete, the grade is recorded in the Moodle gradebook.

When displaying feedback to the student, Moodle displays the comments that the grader has entered along with the grade for the assignment. Typically, when the assignment is not assessed with a rubric, this is the contents of a text field that the grader fills out when entering the grade for the assignment. When using a rubric for assessment, we replace this comment field information with a summary about the comments that the grader has entered (Figure 4), along with the student's scoring broken out by rubric section. Additionally, we provide the student a link to the assessment view that is read-only that shows all of the comments in the context of the original submission.

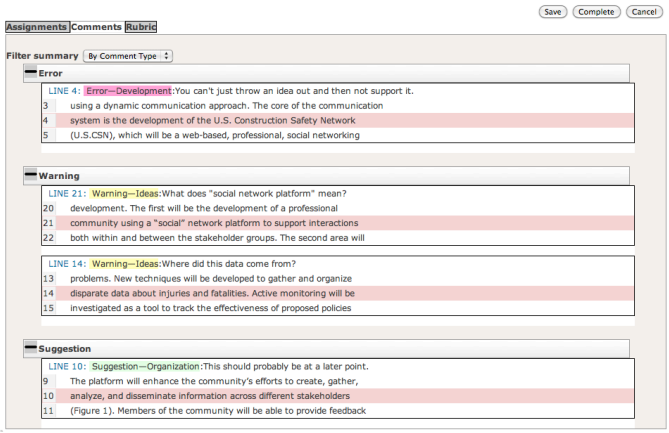

Figure 2 The comments tab summarized the grader feedback by type and also shows the comments within the context of the document.

When displaying feedback to the student, Moodle displays the comments that the grader has entered along with the grade for the assignment. Typically, when the assignment is not assessed with a rubric, these comments are the contents of a text field that the grader fills out when entering the grade for the assignment. When using a rubric for assessment, we replace this comment field information with a summary about the comments that the grader has entered (Figure 4), along with the student's scoring broken out by rubric section. Additionally, we provide the student a link to the assessment view that is read-only that shows all of the comments in the context of the original submission.

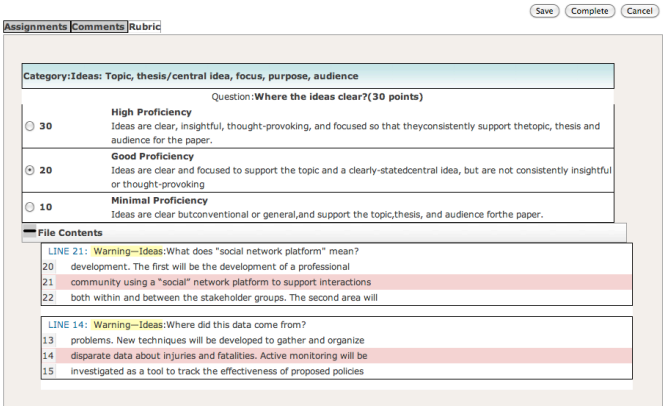

Figure 3 The rubric tab shows the content of the rubric, the points associated with that portion of the rubric, as well as any feedback entered relevant to that section.

For the instructor, additional summary information is provided to give an overview of the class performance as a whole, with regards to the rubric. The aggregate student performance, shown in Figure 5 for each rubric category and question can be summarized and displayed to the instructor. This provides the instructor with an overview of class performance beyond the final number grade, and can provide the instructor with the ability to see where the students need additional instruction, based on their performance.

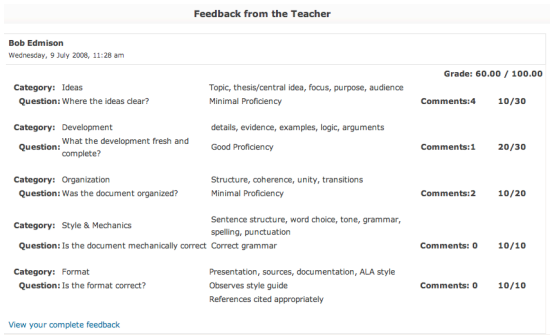

Figure 4 The student feedback section shows the student a summary of their performance relative to the rubric requirements.

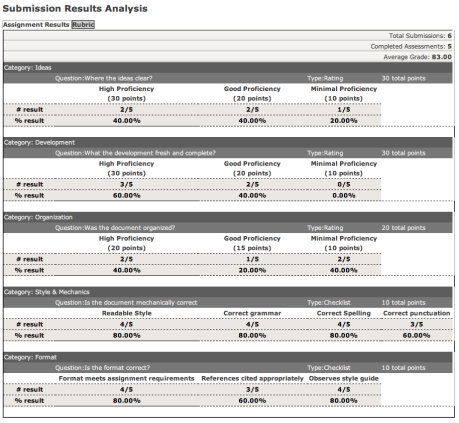

Figure 5 The submissions analysis shows a summary of how the class as a whole performed based on the rubric criteria.

CONCLUSIONS

As we have seen, there are many different approaches to providing computer-assisted assessment. Most efforts have been focused in the computer science discipline, as this is where the researchers active in this field practice. However, that leaves a huge body of the academy unserviced, which might otherwise greatly benefit from these techniques.

Contextualized feedback is critical to learning, and yet very difficult to implement in a automated assessment system. Students benefit from having feedback presented with the original material that caused the feedback. Additionally, keeping the source material and the feedback together provide an opportunity to record what feedback was offered to the student for each assignment. Additionally, we have seen that rubrics offer a powerful tool for framing the assessment of programming and writing assignments, as well as providing a means of directing and categorizing feedback. Coupling feedback and rubrics together can provide a bidirectional feedback path, framing feedback for students about their work and how they should study as well as for instructors, allowing them to quantify exactly how their students are doing on specific concepts based on observed data.

We have developed an extension of the Moodle course management system using rubrics to guide feedback. This tool allows graders to provide this feedback within the context of a student's submission, as well as providing instructors with new insight into student performance beyond the number grade. We believe this will improve the learning experience and outcomes for students, both in traditional settings as well as in distance learning environments.

REFERENCES

[1] Ahoniemi, Tuukka and Ville Karavirta: 'Analyzing the use of a rubric-based grading tool', Jul 2009

[2] Ahoniemi, Tuukka and Essi Lahtinen*, et al.*: 'Improving pedagogical feedback and objective grading'. Proc. SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education, New York, NY, USA2008 pg. 72-76

[3] Ahoniemi, Tuukka and Tommi Reinikainen: 'ALOHA - A Grading Tool for Semi-Automatic Assessment of Mass Programming Courses', Baltic Sea '06: Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006, 2006

[4] Anderson, Richard C. and Raymond W Kulhavy*, et al.*: 'Feedback Procedures in Programmed Instruction', Journal of Educational Psychology, 1971, 62, pg. 148-156

[5] Auvinen, Tapio and Ville Karavirta*, et al.*: 'Rubyric: an online assessment tool for effortless authoring of personalized feedback'. Proc. ITiCSE '09: Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education, New York, NY, USA2009 pg. 377--377

[6] Beck, Kent: 'Test-Driven Development: By Example' (Addison-Wesley, 2003. 2003)

[7] Becker, Katrin: 'Grading programming assignments using rubrics'. Proc. ITiCSE '03: Proceedings of the 8th annual conference on Innovation and technology in computer science education, New York, NY, USA2003 pg. 253-253

[8] http://www.blackboard.com/, accessed 11/04/2009

[9] Blumenstein, M and S Green*, et al.*: 'An Experimental Analysis of GAME: A Generic Automated Marking Environment', May 2004 pg. 67-71

[10] Canup, Mark J. and Russell L. Shackelford: 'Using software to solve problems in large computing courses'. Proc. SIGCSE '98: Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education, New York, NY, USA1998 pg. 135-139

[11] Card, Stuart K. and Thomas P. Moran*, et al.*: 'The psychology of human-computer interaction' (L. Erlbaum Associates, 1983, 0-8985-9243-7 edn. 1983)

[12] Dougiamas, Martin and Peter Taylor: 'Moodle: Using Learning Communities to Create an Open Source Course Management System', Proceedings, Edmedia 2003, 2003, pg. 14

[13] Edwards, Stephen H.: 'Improving Student Performance by Evaluating How Well Students Test Their Own Programs', Journal of Educational Resources in Computing, 2003, 3, (3)

[14] Fischer, Gerhard and Eric Scharff: 'Learning Technologies in Support of Self-Directed Learning', Journal of Interactive Media in Education, 1998, 4, pg. 1-32

[15] Gerhard, Fischer and Scharff Eric: 'Learning Technologies in Support of Self-Directed Learning', Journal of Interactive Media in Education, 1998, 98, pg. 98---94
[16] Gilman, David Alan: 'A Comparison of Several Feedback Methods for Correcting Errors by Computer-Assisted Instruction.'. Proc. In Proceedings of American Psychological Association Meeting, San Francisco, CA, 08 1968 pg. 19
[17] Hattie, John and Helen Timperley: 'The Power of Feedback', Review of Educational Research, 2007, 77, (1), pg. 81--112
[18] Hayes, Andrew F. and Klaus Krippendorff: 'Answering the Call for a Standard Reliability Measure for Coding Data', COMMUNICATION METHODS AND MEASURES, 2007, 1, (1), pg. 77-89
[19] Isaacson, Peter C. and Terry A. Scott: 'Automating the execution of student programs', SIGCSE Bull., 1989, 21, (2), pg. 15--22
[20] Kay, David G. and Terry Scott*, et al.*: 'Automated grading assistance for student programs'. Proc. SIGCSE '94: Proceedings of the twenty-fifth SIGCSE symposium on Computer science education, New York, NY, USA1994 pg. 381--382
[21] Kulhavy, Raymond: 'Feedback in Written Instruction', Review of Educational Research, 1977, 47, (2), pg. 211--232
[22] Lave, Jean and Etienne Wenger: 'Situated learning : legitimate peripheral participation' (Cambridge University Press,, 1991. 1991)
[23] Luck, Michael and Mike Joy: 'Automatic submission in an evolutionary approach to computer science teaching', Computers & Education, 1995, 25, (3), pg. 105 - 111
[24] Luck, Michael and Mike Joy: 'A Secure On-line Submission System'. Proc. Software - Practice and Experience1999 pg. 721-740
[25] Meyers, S. R.: 'Teaching machines and programmed learning', in A.A. Lumsdaine (Ed.) (National Education Association, 1960)
[26] Pardo, Abelardo: 'A Multi-Agent Platform for Automatic Assignment Management', Proceedings of the 7th annual conference on Innovation and technology in computer science education, 2002, pg. 60--64
[27] Popyack, Jeffrey L. and Nira Herrmann*, et al.*: 'Pen-Based Electronic Grading of Online Student Submissions', 2002
[28] Powell, Aaron and Scott Turner*, et al.*: 'An Online Teacher Peer Review System'. Proc. Proceedings of Society for Information Technology and Teacher Education International Conference, Chesapeake, VA2006 pg. 126-133
[29] Price, Blaine and Marian Petre: 'Teaching programming through paperless assignments: an empirical evaluation of instructor feedback', ACM SIGCSE Bulletin, 1997, 29, (3)
[30] http://sakaiproject.org/, accessed 11/04/2009
[31] Suchman, Lucy A.: 'Human-machine reconfigurations: plans and situated actions' (Cambridge University Press, 2007, 2nd Ed. edn. 2007)
[32] Turner, Scott A. and Manuel A. Pérez-Quiñones*, et al.*: 'Effect of interface style in peer review comments for UML designs', J. Comput. Small Coll., 2007, 22, (3), pg. 214--220
[33] Wang, Shu-Ling and Pei-Yi Wu: 'The role of feedback and self-efficacy on web-based learning: The social cognitive perspective', Computers \& Education, 2008, 51, (4), pg. 1589--1598
[34] Winters, Titus and Tom Payne: 'What Do Students Know? An Outcomes-Based Assessment System', Proceedings, ICER '05 Oct 1-2, 2005, Seattle WA, USA, 2005
[35] Winters, Titus and Tom Payne: 'Computer Aided Grading with Agar', Feb 2006

Bob Edmison

Bob Edmison is a Senior Research Associate in the Grado Department of Industrial and Systems Engineering at Virginia Tech, and a PhD student in the Department of Computer Science at Virginia Tech. He received a M.A. Ed and M.IT, both from Virginia Tech. His research interests include the intersection of computing as applied to education, as well as human-computer interaction and software engineering.

Stephen H. Edwards

Stephen H. Edwards received the BS degree in electrical engineering from the California Institute of Technology, and the MS and PhD degrees in computer and information science from the Ohio State University. He is currently an associate professor in the Department of Computer Science at Virginia Tech. His research interests include software engineering, reuse, component-based development, automated testing, formal methods, and programming languages.

Manuel A. Pérez-Quiñones

Manuel A. Pérez-Quiñones is Associate Dean for the Graduate School and Associate Professor in Computer Science at Virginia Tech. He has a D.Sc. from George Washington University. His research is mostly in applied areas of HCI. He is a member of the Coalition to Diversify Computing, ACM, IEEE Computer Society.