122nd ASEE Annual Conference & Exposition
June 14 - 17, 2015
Seattle, WA
Making Value for Society

Paper ID #13706

Work-in-Progress: Automation in Undergraduate Classes: Using Technology to Improve Grading Efficiency, Reliability, and Transparency in Large Classes

Mr. Lee Kemp Rynearson, Purdue University, West Lafayette

Lee Rynearson is currently pursuing a PhD in the School of Engineering Education at Purdue University. He received a BS and MEng in Mechanical Engineering from the Rochester Institute of Technology and has previous experience as an instructor of engineering at the Kanazawa Institute of Technology, in Kanazawa, Japan. His current research interests focus on learning task design and first-year engineering topics.

Mr. David W Reazin, Purdue University

Dave Reazin is currently a third year student at Purdue University working towards a B.S. in Electrical Engineering with a focus on Automatic Controls and Integrated Software Methods. Scheduled to graduate in 2016, Dave plans to enter industry before returning to school to complete his Masters. Throughout his time at Purdue, Dave has also worked as a Resident Assistant and Staff Resident for University Residences, a Teaching Assistant and Grading Systems Team Lead for the Purdue University First Year Honors Engineering Program, and an Electrical Engineering Intern for United Launch Alliance in Cape Canaveral, FL.

Automation in Undergraduate Classes:
Using Technology to Improve Grading Efficiency, Reliability, and Transparency in Large Classes

Abstract

Large undergraduate classes offer many challenges relating to scale. This paper describes a suite of automated computer tools developed to assist with these challenges, specifically those relating to grading and performance analysis for either individual students or classes as a whole. While the computer tools developed are independent of any Learning Management System (LMS), they could be adapted to operate more closely with an LMS in other academic environments.

The suite of tools in question allow for automated digital rubric generation, collection from students, return to students, and most notably, analysis. Features include the ability to condense several files submitted by one student into a single PDF for review, the ability to execute submitted code in three programming languages (Python 3, MATLAB, and ANSI C) while capturing the output into a PDF, and the ability to track error conditions such as late submission and incorrect file names and automatically assign penalties.

Statistical reports are generated for each assignment automatically, providing a window into students' performance and possible areas of concern. Automated warnings alert the teaching team to potential errors in grading, equity issues (such as one section of the class performing substantially better or worse than another) or opportunities for improvement in the academic process (such as rethinking the pedagogy relating to specific ideas or areas that prove broadly troublesome). These reports streamline instructor workflow and allow for deeper insights into student performance than time would normally allow.

The suite of tools was implemented using Visual Basic for Applications (VBA), Python 3, and MySQL databases. The implementation of these automated tools was inexpensive and provided many benefits to the instructors and graders in terms of convenience, time saved, grader accountability, process reliability, and enabling new diagnostic capabilities. Furthermore, cost savings were realized from reduced grader time and from almost eliminating the use of paper to offset the cost of developing the tools. This paper presents details on the tools developed as a part of this effort, preliminary results of the adoption of the tools in a large first-year class, the potential uses of similar tools in other venues, and avenues for future work and development.

Introduction

The use of computer tools to support education and educational administration and assessment, including grading, continues to rise. Large-enrollment courses are one venue where simply managing grading and assignment return can be difficult or time-consuming, and computer tools such as learning management systems (LMS) can be of benefit on a larger scale in larger scale courses.

The computer tools discussed in this paper were developed to support grading procedures in a large-enrollment first-year undergraduate engineering course at a large Midwestern university.

With several hundred students, dozens of undergraduate grading staff members, and a yearlong course sequence featuring more than 100 graded assignments per student, including more than 30 assignments requiring student-generated code to be evaluated, grading is a significant effort and expense for the course.

The primary driver to develop the computer tools was to speed the grading of assignments with student-submitted programs. While graders could simply 'eyeball' a student's code, there is an increased potential for error compared with reviewing actual program output for various test cases. Running each student's code adds extra steps and time requirements, in addition to the potential for the grader to make a mistake in setting up or running the program. The faculty determined that the capability to automatically run a student's code in the various languages used in the course (Python 3, MATLAB, and ANSI C) and capture its output for multiple test cases (where program input would be unknown to the student) would substantially speed and improve the grading process in the course.

Once that determination was made, opportunities were observed to extend the capabilities of the system to gain other benefits for the course in other areas, including eliminating a variety of grader errors associated with grading on paper rubrics (such as omitted or incorrect student names or incorrect summations of earned points) to increase grading reliability, addressing student reports that different course grading staff members consistently rate students higher or lower than other grading staff members to address an important equity issue, and gaining insight into the performance of students on specific aspects of assignments, overall and between sections of the course, that would otherwise be too time-consuming to produce manually on a regular basis, improving transparency (to the faculty) and allowing for interventions to be implemented when situations of concern arise.

While some features of the overall set of computer tools produced duplicate features that are commonly and commercially available, such as the ability to create digital grading rubrics for each student in the class, others capabilities are less common, especially in formats that are useful for large enrollment engineering classes.

While automated assessment of programming assignments has been pursued since the punch-card era[4] and research continues vigorously[5], their use is not widespread in engineering instruction and it remains common for completely manual inspection of student code to be performed, especially for relatively complex codes that permit design creativity. Our tools retain human review of student code and outputs but allow much greater speed, while supporting several languages used in engineering and feeding into a system for student and course performance analysis.

Grading equity between students has been approached from several directions, including the use of grading rubrics[6] and TA training[3] but classes with many graders may benefit from the use of grading data to examine grader propensity for harsh or lenient grading directly to enable evidence-based faculty intervention in extreme cases. Our course tools support some methods for direct examination of grader performance and suggest avenues for further work in this area.

'Big data' and learning analytics have become ubiquitous terms in higher education in the last few years, and some examples of big data being employed to the benefit of individual courses certainly exist[1, 7] many applications of big data in higher education are occurring at an institutional level. However, the ability to harvest useful data from individual assignment criteria for a wide range of engineering assignments and provide timely information to faculty about performance at such a granular level in an automated fashion is not yet widespread. Large enrollment courses, where it is difficult for faculty to have personal contact with the majority of students and where a quantity of students is present sufficient to enable statistical analysis of performance data may especially benefit from implementing computer tools to collect and analyze course data. The course tools described in this paper illustrate the collection of nearly all course grading data and automated per-assignment results reporting and statistical analysis.

Computer Tools Development & Costs

To achieve the goals laid out in the introduction, seven computer tools were created. Development and deployment occurred on a rolling basis over three semesters. Different versions of the digital rubrics were tested experimentally for student readability and grader grade-entry speed prior to settling on a final design. All hands-on code development was undertaken by undergraduate students employed by the course, overseen by faculty and graduate students. It is estimated that fewer than 200 undergraduate working hours were spent in developing and testing the computer tools, leading to a labor cost of less than $1500.

Developed Computer Tools

The seven existing computer tools are all employed in the grading and analysis of a single assignment. Figure 1 shows the workflow that would be used in the grading and analysis of a single class assignment, illustrating the relationship between the seven tools and their inputs/outputs. Four of these seven tools (1.1, 2.1, 3.1, 4.1) mostly duplicate features that would commonly be available for in-LMS grading workflows (such as automatically generating grading rubrics with each student's name, or returning graded work to students) but were created to support and streamline the process of using the three tools (2.2, 4.3, and 5.1) that provide more interesting features.

The decision to develop standalone computer tools versus attempting some level of integration with the current campus LMS was made based on the judgment that any such integration would involve a great deal of political 'overhead' if it were even permitted, and would also expose the system to changes made to the LMS itself, which would be outside the control of course faculty. Circumstances may differ across time and institutions.

Each computer tool's nature and function will be briefly discussed.

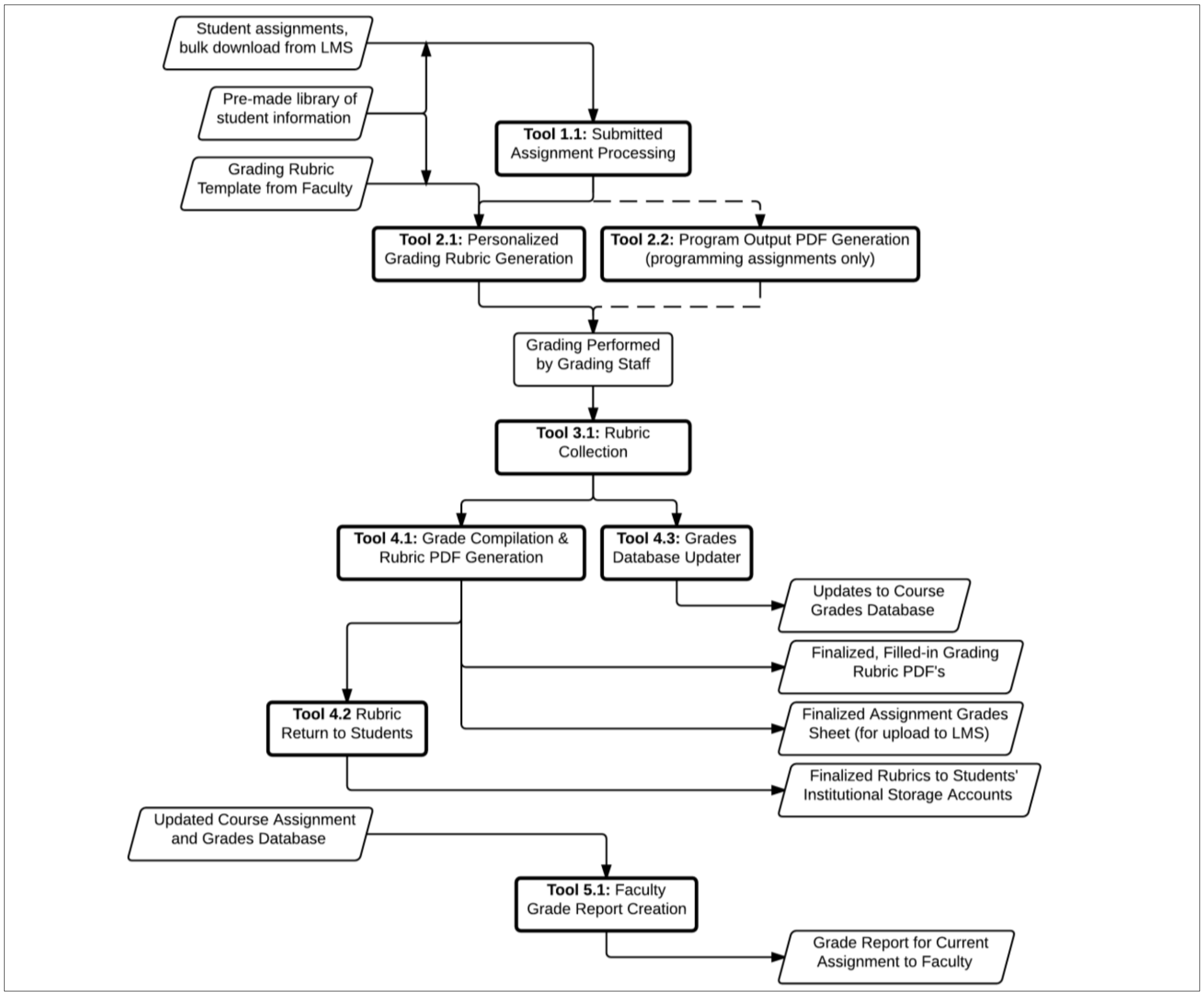

Student assignments, bulk download from LMS
Pre-made library of student information
Grading Rubric Template from Faculty
Tool 1.1: Submitted Assignment Processing
Tool 2.1: Personalized Grading Rubric Generation
Tool 2.2: Program Output PDF Generation (programming assignments only)
Grading Performed by Grading Staff
Tool 3.1: Rubric Collection
Tool 4.1: Grade Compilation & Rubric PDF Generation
Tool 4.3: Grades Database Updater
Updates to Course Grades Database
Finalized, Filled-in Grading Rubric PDF's
Tool 4.2 Rubric Return to Students
Finalized Assignment Grades Sheet (for upload to LMS)
Finalized Rubrics to Students' Institutional Storage Accounts
Updated Course Assignment and Grades Database
Tool 5.1: Faculty Grade Report Creation
Grade Report for Current Assignment to Faculty

Figure 1 - Flowchart of Computer Tool Use, Inputs, and Outputs for a Generic Assignment

Computer Tool	Language	Program Inputs	Program Outputs	Output
1.1: Submitted Assignment Processing	Python	Raw student submissions from LMS	Intelligibly renamed and sorted student submissions, listing of student assignments requiring late or incorrect filename penalty	Varies (.docx, .pptx, .py, .m, .c, PDF, .txt)
2.1: Personalized Grading Rubric Generation	VBA	Master rubric from faculty w' grading criteria	Rubrics personalized for each student w' any late submission or wrong filename penalties applied	Excel (.xlsx)
2.2: Program Output PDF Generation	Python, MATLAB	Student submitted programs (.py, .c, .m)	PDF's combining student-submitted code and code execution output for test cases	PDF
3.1: Rubric Collection	Python	Graded rubrics	Graded rubrics, sorted	Excel (.xlsx)
4.1: Grade Compilation & Rubric PDF Generation	VBA	Graded rubrics, sorted	Finalized assignment grades sheet (for LMS upload)	Excel (.xlsx), PDF
4.2: Rubric Return to Students	Python, LINUX	Graded rubrics, sorted	Rubrics returned to students' institutionally provided computer storage accounts	PDF
4.3: Grades Database Updater	Python	Graded rubrics, sorted	Updated MySQL database storing per-grading-criterion performance	MySQL updates
5.1: Faculty Report Creation	Python	MySQL database	PDF Assignment Reports	PDF

Table 1 - Computer Tool Languages, Inputs, and Outputs

Tool 1.1: Submitted Assignment Processing

After an assignment's submissions are bulk downloaded from the LMS (BlackBoard Learn), this Python tool alters the file names to be shorter and more human-readable, creates a new assignment-specific file structure to contain the student work using a library of student data (name, working team, grader, etc.), and sorts each submitted file into an appropriate folder for later grading. This tool also outputs a text listings of students who submitted late or with incorrect file names, based upon information supplied by the LMS. These outputs are used by Tool 2.1 to assess penalties automatically.

Tool 2.1: Personalized Grading Rubric Generation

Based on a master grading rubric provided by the faculty containing grading criteria, point values, etc., a VBA script is used to automatically generate personalized rubrics in Excel format for each student or team in the class, depending on assignment type. Excel was selected for grade entry as a platform that would be widely available to and understood by our undergraduate grading staff. Rubric personalizations are based on a predefined dictionary of class information and include student name, grader name, and team number. Penalties are automatically assessed for incorrect filenames or late submissions on a student-by-student basis based on the outputs from Tool 2.1 and a message stating the reason for the penalty is applied to the rubric. After the rubrics are generated, a Python3 script sorts the rubrics into the file structure created by Tool 2.1 such that the personalized rubric is placed in the same directory as the renamed student or team submissions.

Tool 2.2: Program Output PDF Generation

To support rapid and consistent grading, a set of scripts were developed to automatically compile (if needed) and run student-submitted programs written in Python 3, MATLAB, or ANSI C. Test cases can be automatically run against all student programs and a single PDF is generated for each student that shows the original student-submitted code, the output of the code against each of the test cases, and administrative information such as the filename, number of lines of code, line numbers, maximum line length, and any error messages that may have been generated during the output PDF creation process, such as the failure of the submitted program to compile. Parallel to this stage, student work is compared via MOSS[2] to assess the likelihood of plagiarism in specific assignments, but that part of the process is not automated at this time. MOSS operations may be automated into this or another tool at a later time.

Tool 3.1: Rubric Collection

After the undergraduate grading staff fills in the grading rubrics, this Python 3 script copies all filled-in rubrics into a single location. The primary reason for collecting all rubrics into a single directory is administrative; it makes certain that all expected rubrics are found and ensures that no rubrics are opened by users when the subsequent script is run.

Tool 4.1: Grade Compilation & Rubric PDF Generation

A VBA script extracts the final grade information from each filled-in student rubric into an Excel spreadsheet. During this process, each rubric is checked for completion, and the operator is notified of potential errors in grading including a line item being left blank or invalid input. This script supports the grading of individual or team assignments in addition to forms of team grading drawing upon the performance of individual students. These include the ability to assign the highest, lowest, or average team grades to all members of the group. These capabilities are employed in the classroom to motivate team interdependence in an active-learning environment. The resulting spreadsheet contains the assignment grades in several different formats for convenient review, upload back into the LMS, and for upload into an LMS-independent overall course grading spreadsheet. While compiling the assignment grades, this VBA script also creates a PDF version of each personalized and filled-in rubric for later return to students.

Tool 4.2: Rubric Return to Students

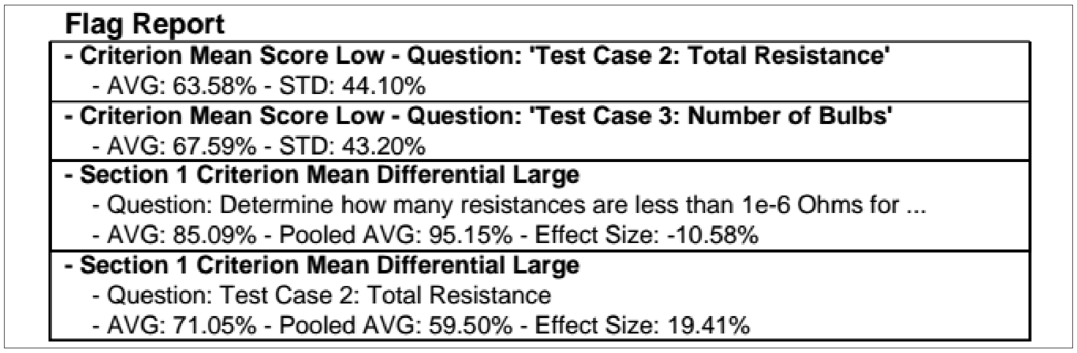
This Python 3 script sorts the finalized rubric PDF's generated by Tool 4.1 into a distribution directory on a LINUX server, where each student has password-protected read-only access to their own rubrics. A companion script is provided to students which installs a scheduled task (cron job) on their account in order to copy their rubrics to their Windows-accessible institutional file storage automatically. This script updates the destination directory name to indicate the date of the last update, notifying students that new or updated rubrics have arrived. The file transfers for each student are scheduled to occur at different times to reduce strain on the hosting servers. A ‘self-destruct’ mechanism is built in to allow course administrators to remotely remove the scheduled tasks from student accounts once students have completed the course.

Tool 4.3: Grades Database Updater

This Python 3 script is run on the graded rubrics to store assignment, rubric, and grade information in a course MySQL database. The script is capable of automatically identifying and accounting for many expected variations in rubric layout while reading all information appropriately. The resulting data is stored in three separate tables, detailing the general assignment information (such as title, total possible points, date completed, and assignment type), question specific information (detailing each grading criterion, including the full text of the criterion, the points possible for full, partial, and failure to complete the criterion, and the line’s relative location on the rubric), and grade information (such as the student’s unique identifier, the grader’s name, the earned grade, and the student’s team, section, and professor). All FERPA-sensitive information is stored in a hashed format, mitigating risks to student confidentiality. The database is organized to be easy to search, query, or analyze as needed. Data sets can be constructed from nearly arbitrary requirements, such that analysis can be done on any set or subset of students, assignments, sections, teams, graders, or other dividing details.

Tool 5.1: Faculty Grading Report Creation

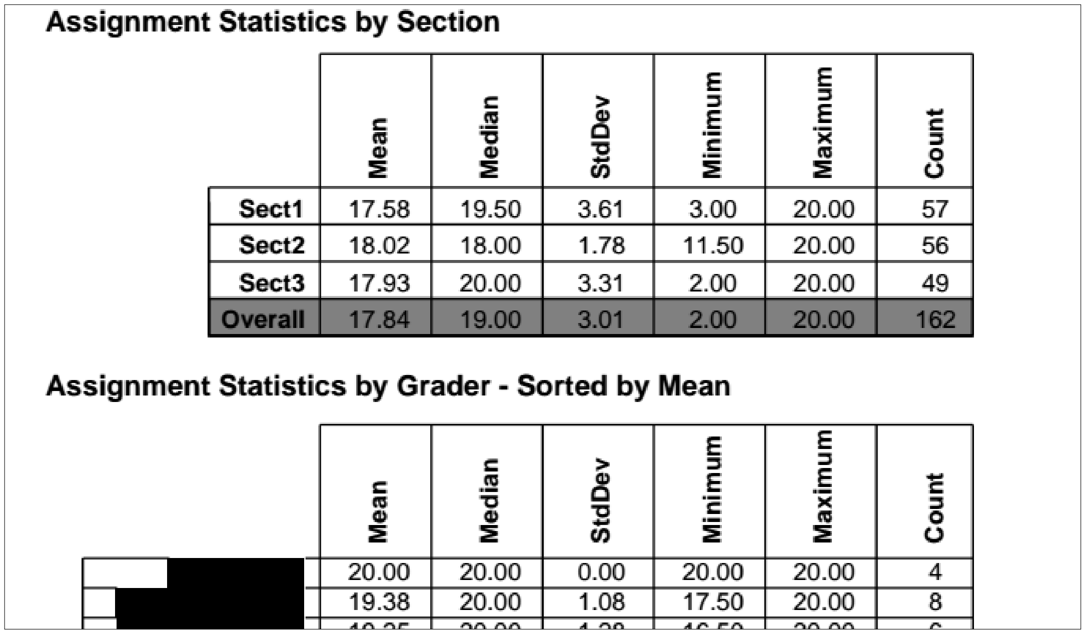
After the course database has been updated with the details from the assignment, this Python 3 script is used to automatically generate a report on student performance on a given assignment. These reports include basic information about the assignment, overall statistics of student performance, statistical breakdown by section, criterion specific analysis, reports on the grades and feedback given by individual course grading staff members, and notifications of any causes for concern with the assignment identified. These notifications include any single grading criterion with a mean classwide score below a predefined threshold, any large differences in performance between sections of the course on the entire assignment or individual criteria, grader performance substantially different than peer performance, and other basic diagnostic tools. While these notifications are known to be potentially confounded by student team effects, grader effects, section effects, instructor effects, etc., they are suitable for directing instructional team attention to matters that may require additional investigation. The generated reports are automatically sent to the appropriate members of the instructional team for their review. Sample report output for notifications is shown in Figure 2. Samples of performance analysis appear in Figure 3 and Figure 4. These images come directly from the automated tool.



Flag Report

- **Criterion Mean Score Low - Question: 'Test Case 2: Total Resistance'**
  - AVG: 63.58% - STD: 44.10%
- **Criterion Mean Score Low - Question: 'Test Case 3: Number of Bulbs'**
  - AVG: 67.59% - STD: 43.20%
- **Section 1 Criterion Mean Differential Large**
  - Question: Determine how many resistances are less than 1e-6 Ohms for ...
  - AVG: 85.09% - Pooled AVG: 95.15% - Effect Size: -10.58%
- **Section 1 Criterion Mean Differential Large**
  - Question: Test Case 2: Total Resistance
  - AVG: 71.05% - Pooled AVG: 59.50% - Effect Size: 19.41%

Figure 2 - Automated Notification Flags for Per-Criteria Performance of Concern



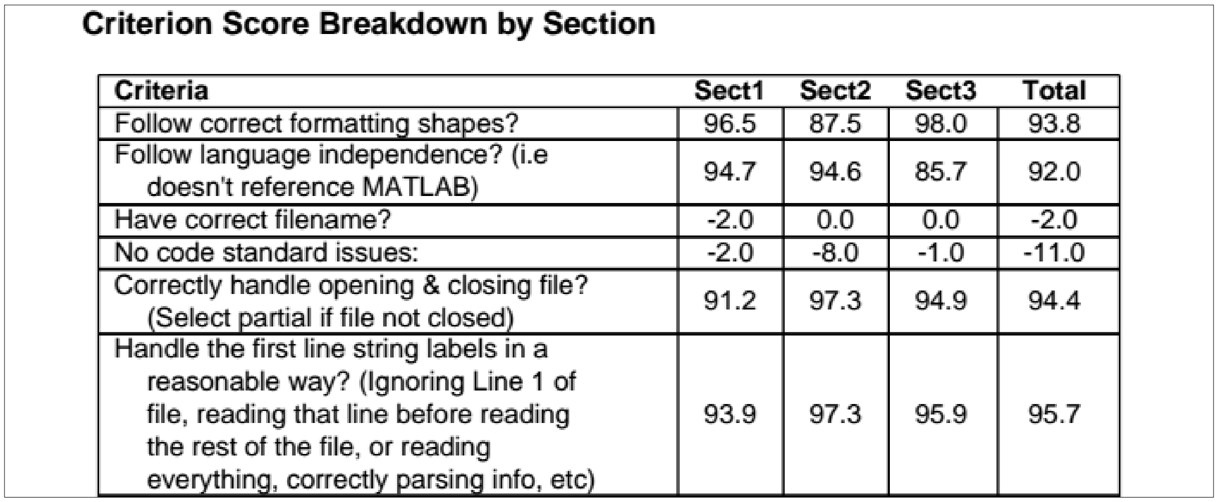
Assignment Statistics by Section

	Mean	Median	StdDev	Minimum	Maximum	Count
Sect1	17.58	19.50	3.61	3.00	20.00	57
Sect2	18.02	18.00	1.78	11.50	20.00	56
Sect3	17.93	20.00	3.31	2.00	20.00	49
Overall	17.84	19.00	3.01	2.00	20.00	162

Assignment Statistics by Grader - Sorted by Mean

	Mean	Median	StdDev	Minimum	Maximum	Count
[redacted]	20.00	20.00	0.00	20.00	20.00	4
[redacted]	19.38	20.00	1.08	17.50	20.00	8

Figure 3 - Automated Descriptive Statistics by Section and Grader (Names Redacted)



Criterion Score Breakdown by Section

Criteria	Sect1	Sect2	Sect3	Total
Follow correct formatting shapes?	96.5	87.5	98.0	93.8
Follow language independence? (i.e doesn't reference MATLAB)	94.7	94.6	85.7	92.0
Have correct filename?	-2.0	0.0	0.0	-2.0
No code standard issues:	-2.0	-8.0	-1.0	-11.0
Correctly handle opening & closing file? (Select partial if file not closed)	91.2	97.3	94.9	94.4
Handle the first line string labels in a reasonable way? (Ignoring Line 1 of file, reading that line before reading the rest of the file, or reading everything, correctly parsing info, etc)	93.9	97.3	95.9	95.7

Figure 4 - Automated Per-section Per-criterion Averages

Undergraduate course staff now regularly operate these tools for course assignments, and have written operations manuals as well as successfully training replacement staff. We anticipate labor costs of system operation to be more than offset by other labor savings.

Data Gathering

It was not administratively desirable to bifurcate course grading such that one cohort of students and graders would employ the previous, paper-based grading system and another the newer all-digital system at the same time. The priority for course faculty during the transition was to ensure the error-free operation of the various tools, and while feedback from students and graders was informally sought at several stages, data was not retained. For instance, several assignments were graded on a timed basis by our graders using both the old and new methods as an early version of the system was being prepared for use, and it was found that grading on the digital rubrics was equivalent in speed or faster for all graders versus paper, but the specific timing data was not retained once the decision to continue with development was made. Therefore, it is difficult to make quantitative statements about the improvements to efficiency and reliability offered by the new computerized course tools. However, as the new systems offer new capabilities and eliminate certain classes of grading error entirely, some effects can be reported on qualitatively. In the cases, the effects and benefits reflect a consensus of the faculty and grading staff actively involved with the use of the computer tools.

Computer Tool Effects and Capabilities

The effects and capabilities brought to the course by the development and use of the seven computer tools are summarized in Table 2. Some of the advantages listed are with respect to grading digitally versus grading on paper, which may be less interesting to readers already employing all-digital grading processes. For example, Tool 4.2 lists "No potential for rubrics to be left in class, taken by another student, lost, etc. as with paper rubrics". However, effects or capabilities listed in bold extend beyond capabilities available through standard LMS deployments. While most items in the table are self-explanatory between the text and the examples shown in the Developed Computer Tools section, items marked with an asterisk in Table 2 receive further explanation subsequent to the table.

Computer Tool	**Program Benefits to Course Administration or Pedagogy**
1.1: Submitted Assignment Processing	-Automated enforcement of late submission penalties -**Automated enforcement of incorrect file name submission penalties**
2.1: Personalized Grading Rubric Generation	-Elimination of rubrics returned without, or with incorrect, student name, grader name, and/or team number information -Elimination of mathematical errors in grading student work, faster grading by automating final grade calculations
2.2: Program Output PDF Generation	-**Reduce grading time by providing automatic compilation and output of student submitted programs in unified, convenient PDF format** -**Eliminate grading errors due to grader use of non-standard software, non-standard versions of software, non-standard compilers, or non-standard compiling flags when assessing student programs**
3.1: Rubric Collection	-Administrative convenience, no additional benefits
4.1: Grade Compilation & Rubric PDF Generation	-Checks for incomplete or invalid grader input, informs user if present -**Assignment grades output automatically provided in various formats required for different administrative needs** -**Automatic inclusion of 'digital stickers' for exceptional work (requested by undergraduate grading staff, appreciated by at least some students)**
4.2: Rubric Return to Students	-Work returned to students quickly, no need to wait for class -Students retain digital copies of their rubrics for later reference -No potential for rubrics to be left in class, taken by another student, lost, etc. as with paper
4.3: Grades Database Updater	-**Automatic long-term archive of class performance information to enable future assessment & research*** -**Enables easy, sorted retrieval of nearly any grading data of interest, down to the level of individual assignment grading criteria**
5.1: Faculty Report Creation	-**Provides automatic, detailed assessment and feedback on student and grader performance for each class assignment*** -**Provides automatic notifications of situations of potential concern in assignment performance to faculty at overall and section-specific levels, allowing faculty investigation and intervention***
Other Effects	-Backup at every stage of the grading process permits detailed investigation and rectification if a grading error is discovered -Reduced printing costs and need for grading labor*

Table 2 - Computer Tool Benefits

Tool 5.1, the Faculty Report Creation tool, has potential benefits and effects that should be mentioned in more detail. By understanding student performance on each assignment on a per-criterion level, faculty have strong evidence to determine when problems exist in student understanding that are severe or recurring, directing faculty intervention (such as additional discussion or examples on the related topic) to areas of greatest need. Unreported problems in assignments themselves have also been diagnosed through the discovery of low student performance. Similarly, large differences in performance between sections of the course (one item that is automatically flagged for review) prompt investigation into what a specific section did or did not do, potentially informing the faculty of specific classroom learning experiences that have proven particularly effective or ineffective. Review and use of assignment performance data is made tremendously more frequent and efficient by automating the reporting of the data; time would not otherwise permit in-depth analysis of student performance on each assignment. With the automated reports, the faculty are able to review and discuss performance on all assignments and any implications for the classroom at regular weekly meetings.

Tool 4.3, the Grades Database Updater, along with the grades database itself, also merits mention in more detail. While other methods than a permanent database could be used to enable the Faculty Report Creation tool (its principal current role) the long-term promise of the database with respect to understanding student performance and success in the class is one of the greatest benefits of the system of tools developed. By storing detailed data on student performance for every criterion of every assignment passing through the system over time, faculty hope to eventually examine aspects of student performance to answer interesting questions about the course overall. For instance, common failure modes on different assignment types could be accurately determined over time and across many assignments or student performance could be related to demographic data including prior preparation. One area of special interest is supporting students with no prior programming experience, such that performance on programming assignments is comparable between students with and without prior programming experience. The grades database and the tool that updates it with data from each assignment therefore offer the potential of long-term course benefits difficult to predict at this time, including the potential to support educational research.

The cost of system development (est. $1500) is expected to be recovered in one academic year or less through reduced costs for printing and grading labor. It is estimated that between 60,000 and 100,000 pages of rubrics and assignment output were either copied or printed per academic year to support class grading prior to the deployment of this system, which is now completely unnecessary. The efficiencies from speeding grader assessment of student work have not been assessed over time (it is expected that regular users of the system will be faster than novices, and novice users tested as equivalent or faster than paper grading as previously mentioned), but informal surveys of grading staff indicate that the new system was seen as very convenient compared to the one it replaced. The largest gains are reported to descend from the pre-prepared student programming output PDF's from Tool 2.2 and from not needing to physically retrieve and transport paper rubrics or student work from place to place.

Discussion

The analysis of grader performance as implemented in the reports from Tool 5.1 is limited in some important respects, by design choice. The faculty were unwilling to require and pay for a subset of assignments be graded two or more times, which eliminates the possibility of implementing a robust system to examine inter-rater reliability. Similarly, the faculty wished each grader to grade the same students across all assignments, as had been traditional, due to the relationships that graders formed with ‘their’ students, which was seen as producing superior interaction with and feedback to students. This precludes accounting for grader effects by analyzing how different graders assessed different students. The remaining tools, examining mean scores and standard deviations on a per-grader basis are confounded by student performance and therefore are less useful in directing faculty attention to unduly harsh or unduly lenient grading. It does provide some evidence to examine when faced with student reports of unduly harsh grading and affords the faculty some feeling for how individual graders are performing. Implementations of similar tools need not face these limitations if course leadership is willing to grade some assignments repeatedly and rotate graders – in these cases it would be much easier to gain insight into relative grader harshness and also to identify assignments with rubrics that have lower grader inter-rater reliability. This could be used to focus faculty or graduate staff attention on rubrics that do not adequately support the assessment process.

The notification flags in the reports provided by Tool 5.1 are also face limitations, but for different reasons. The primary limitation is that there are few hard references available with which to compare the performance of students of the course or individual sections of the course against. Many of the notifications are somewhat arbitrary for this reason. For example, the faculty decided that they would be like to be notified automatically when average performance on a criterion dropped below 75% not because 75% had intrinsic significance, but because that was the level of performance below which they would contemplate intervention to address a problem. There is potential to use the grades database to build up an understanding of expected performance to judge when areas of concern should be automatically flagged, but this remains a future work. Therefore, the notifications currently serve to prompt further discussion or investigation and are not sufficient of themselves to be diagnostic.

Future work to develop additional tools that benefit from student and grader performance data is planned. Currently, the automated analysis and reporting from Tool 5.1 focuses on a single assignment. However, the potential exists for longitudinal assessment across multiple assignments. One example of a future work in this area is a diagnostic tool for predicting student success in the course to direct faculty and staff support to students with low initial performance, similar to Purdue University's Course Signals system[1]. Informed by data from past course cohorts, the potential for strong success prediction based on performance on a few early assignments is seen. The potential for educational research based on student demographic data and longitudinal course performance is also observed.

An additional future work is to begin capturing grader feedback to students, for the purposes of both understanding what feedback is being given and identifying graders who are consistently giving weak feedback to students. Further development of tools 4.3 and 5.1 is ongoing to incorporate features along these lines.

Conclusions

The development and deployment of the computer tools described in this paper has increased the efficiency, reliability, and transparency of the grading process in a large enrollment course. While it is difficult to assess many of the benefits of these tools quantitatively, their administrative advantages are clear, the analytic insight into student and grader performance from the automatically generated reports is significant and helpful to the faculty, and the potential for future development of tools for course administration or research based on the grading information database is compelling. Other large-enrollment courses may find value in creating standalone or LMS-integrated computer tools along similar lines, especially for the tools that support automated capture of student-submitted program output and course-specific learning analytics. It is also noted that the cost of developing these tools was low and is expected to pay for itself in decreased printing and grading labor costs within one academic year after deployment, given the number of students and assignments served.

References

1 Kimberly E Arnold, and Matthew D Pistilli, 'Course Signals at Purdue: Using Learning Analytics to Increase Student Success', in *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge* (ACM, 2012), pp. 267-70.

2 Kevin W Bowyer, and Lawrence O Hall, 'Experience Using" Moss" to Detect Cheating on Programming Assignments', in *Frontiers in Education Conference, 1999. FIE'99. 29th Annual* (IEEE, 1999), pp. 13B3/18-13B3/22 vol. 3.

3 Heidi A Diefes-Dux, 'A Teaching Assistant Training Protocol for Improving Feedback on Open-Ended Engineering Problems in Large Classes', (2013).

4 Christopher Douce, David Livingstone, and James Orwell, 'Automatic Test-Based Assessment of Programming: A Review', *J. Educ. Resour. Comput.,* 5 (2005), 4.

5 Petri Ihantola, Tuukka Ahoniemi, Ville Karavirta, Otto Sepp, #228, and #228, 'Review of Recent Systems for Automatic Assessment of Programming Assignments', in *Proceedings of the 10th Koli Calling International Conference on Computing Education Research* (Koli, Finland: ACM, 2010), pp. 86-93.

6 Anders Jonsson, and Gunilla Svingby, 'The Use of Scoring Rubrics: Reliability, Validity and Educational Consequences', *Educational Research Review,* 2 (2007), 130-44.

7 Marc Parry, 'Big Data on Campus', *The New York Times* 2012.