ASEE'S VIRTUAL CONFERENCE
At Home with Engineering Education JUNE 22 - 26, 2020 #ASEEVC
Paper ID #29451

WIP: Lessons Learned from Applying Standards Based Grading to a Software Verification Course

Dr. Walter W Schilling Jr., Milwaukee School of Engineering

Walter Schilling is a Professor in the Software Engineering program at the Milwaukee School of Engineering in Milwaukee, Wisconsin. He received his B.S.E.E. from Ohio Northern University and M.S. and Ph.D. from the University of Toledo. He worked for Ford Motor Company and Visteon as an Embedded Software Engineer for several years prior to returning for doctoral work. He has spent time at NASA Glenn Research Center in Cleveland, Ohio, and consulted for multiple embedded systems companies in the Midwest. In addition to one U.S. patent, Schilling has numerous publications in refereed international conferences and other journals. He received the Ohio Space Grant Consortium Doctoral Fellowship and has received awards from the IEEE Southeastern Michigan and IEEE Toledo Sections. He is a member of IEEE, IEEE Computer Society and ASEE. At MSOE, he coordinates courses in software verification, real time systems, operating systems, and cybersecurity topics.

WIP: Lessons Learned from Applying Standards Based Grading to a Software Verification Course

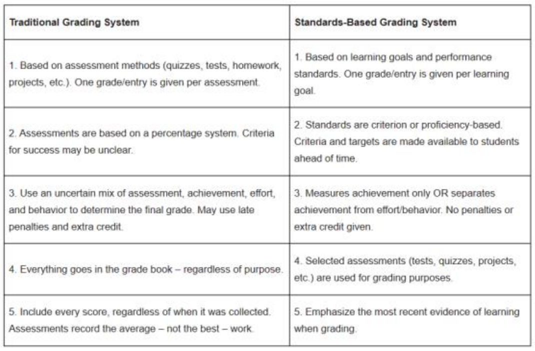
Abstract

One of the newer approaches to grading and assessment is standards-based grading. Standards Based Grading, otherwise known as SBG, directly measures student's proficiency in a course based upon specific course learning outcomes. This approach offers an alternative to traditional, summative based grading systems. Thus far, there have been very few attempts to integrate standards-based grading into computing fields. This work in progress paper will discuss the process of how SBG is being integrated in a sophomore level software verification course. The paper will include a listing of the specific concepts assessed in the course as well as a discussion of the mechanisms used to provide feedback to students on their performance. The paper will also provide a discussion of the challenges of integrating SBG into a college course, and some of the reasons why a complete SBG approach has not yet been undertaken for the course.

Introduction

Assessment of student learning is an important aspect of any software engineering instructor's work. Traditionally, faculty assign grades based on a singular mechanism, a percentage-based score. Assignments are scored on a percentage basis, and those percentages are then weighted to determine a final grade. While this can work well, the process itself may hide problem areas. The grade, while numerically precise, may not necessarily be mapped the learning goals of the class, and the criteria for success may be unclear.

An alternative approach to grading, gaining significant traction in the K-12 system is standards-based grading. With standards-based grading, grading is based upon "measuring students' proficiency on well-defined course objectives." [1] Instead of arbitrary grading scales, students are assessed multiple times regarding their performance on course outcomes. By doing this, there is an increase in student engagement and a more thorough comprehension of course materials. [2] Standards Based grading focuses on the specific, relevant skills a student should learn and helps instructors to assess how well students are learning and tailor their teaching to meet areas of concern. [3] By measuring these goals, students continue to learn. By using rubrics to articulate these goals, students can use this scaffolding to be more effective learners. [4] A comparison of the two systems is shown in Figure 1 and Figure 2.



Traditional Grading System	Standards-Based Grading System
1. Based on assessment methods (quizzes, tests, homework, projects, etc.). One grade/entry is given per assessment.	1. Based on learning goals and performance standards. One grade/entry is given per learning goal.
2. Assessments are based on a percentage system. Criteria for success may be unclear.	2. Standards are criterion or proficiency-based. Criteria and targets are made available to students ahead of time.
3. Use an uncertain mix of assessment, achievement, effort, and behavior to determine the final grade. May use late penalties and extra credit.	3. Measures achievement only OR separates achievement from effort/behavior. No penalties or extra credit given.
4. Everything goes in the grade book – regardless of purpose.	4. Selected assessments (tests, quizzes, projects, etc.) are used for grading purposes.
5. Include every score, regardless of when it was collected. Assessments record the average – not the best – work.	5. Emphasize the most recent evidence of learning when grading.

Figure 1: A comparison of Standards Based Grading and Traditional Grading [5]

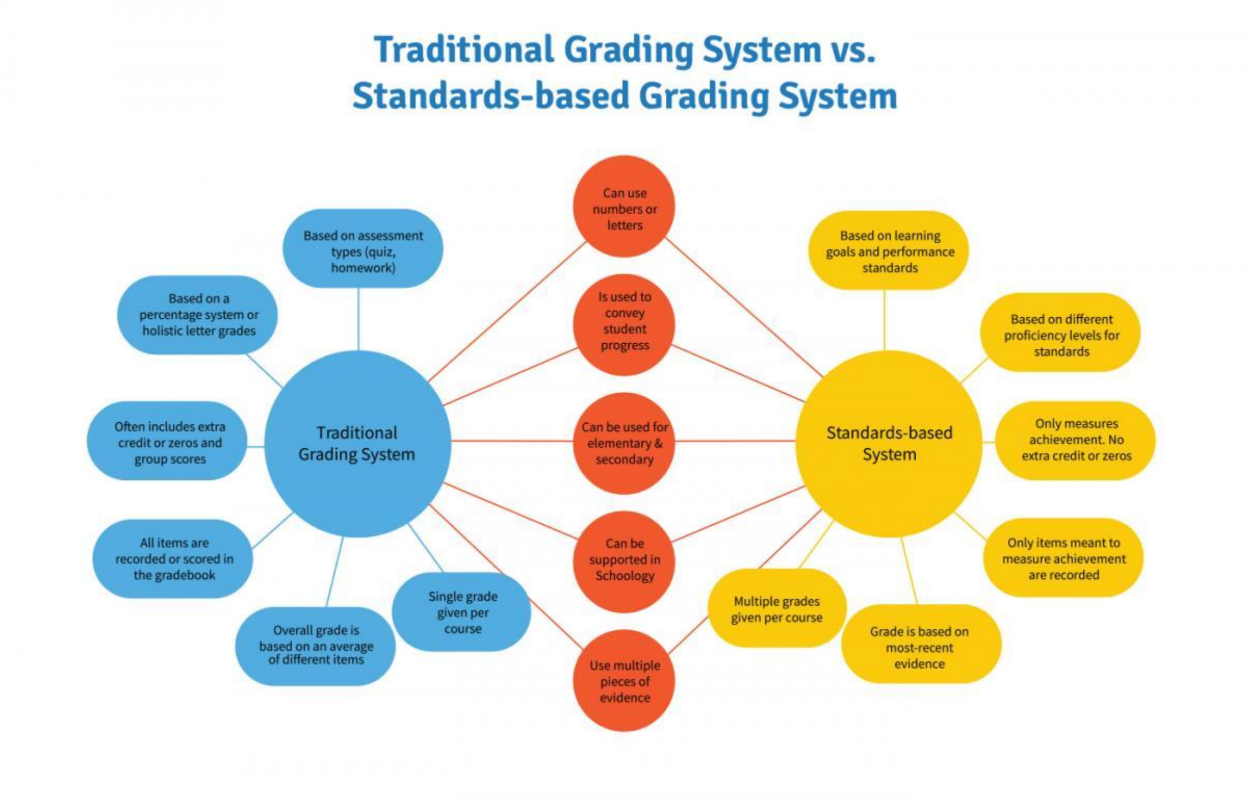

Traditional Grading System vs.
Standards-based Grading System
Traditional Grading System
Standards-based System
Based on assessment types (quiz, homework)
Based on a percentage system or holistic letter grades
Often includes extra credit or zeros and group scores
All items are recorded or scored in the gradebook
Overall grade is based on an average of different items
Single grade given per course
Can use numbers or letters
Is used to convey student progress
Can be used for elementary & secondary
Can be supported in Schoology
Use multiple pieces of evidence
Based on learning goals and performance standards
Based on different proficiency levels for standards
Only measures achievement. No extra credit or zeros
Only items meant to measure achievement are recorded
Grade is based on most-recent evidence
Multiple grades given per course

Figure 2: A second comparison between Standards Based Grading and Traditional Grading Systems [6]

In the engineering field, standards-based grading is a relatively new phenomenon. Summative based approaches have tended to be the norm, and still dominate the field. Multiple papers have been published describing the impacts of Standards Based Grading within the engineering community. [7] [8] [9] However, no specific papers have been found adopting this technique to the teaching of software engineering, though papers have been publishing showing it adopted to other electrical engineering courses, such as Signals and Systems. [10]

Institutional and Course Profile

The Milwaukee School of Engineering offers an accredited Bachelor of Science degree in software engineering and has been accredited since 2001. There is a strong emphasis on small class sizes (13:1 student to faculty ratio) and extensive usage of laboratory learning experiences.

The program offers students several unique learning opportunities. One part of the program is a 10 credit Software Development Laboratory experience where students work on large-scale, industry-sponsored projects. Prior to this, students enroll in a course in software verification, defined in Figure 2. Specifics of this course are described in [11]. The software verification course uses many of the approaches of standards-based grading to assess student performance. However, it cannot be considered to be using pure standards-based grading.

Course Description

This course introduces students to the fundamental concepts of software verification. Topics covered include the activities within testing, coverage criteria, basic testing techniques and types, basic testability metrics, and the application of testing tools. Laboratory assignments provide extensive opportunities to apply software verification techniques and tools. (prereq: <Redacted>)

Course Learning Outcomes

Upon successful completion of this course, the student will be able to:

1. Explain why testing is important to software development
2. Explain the relationship between verification and validation
3. Compose accurate and detailed defect reports and record defects into a defect tracking system
4. Using appropriate coverage criteria and testing theory, design and construct high quality testing approaches and prepare tests in a logical, organized fashion
5. Apply testing theory to design tests based on presented test criteria
6. Analyze the effectiveness of testing using testing metrics, mutation testing, and other techniques
7. Design and implement test cases using mock objects
8. Analyze a given piece of source code for complexity and testability

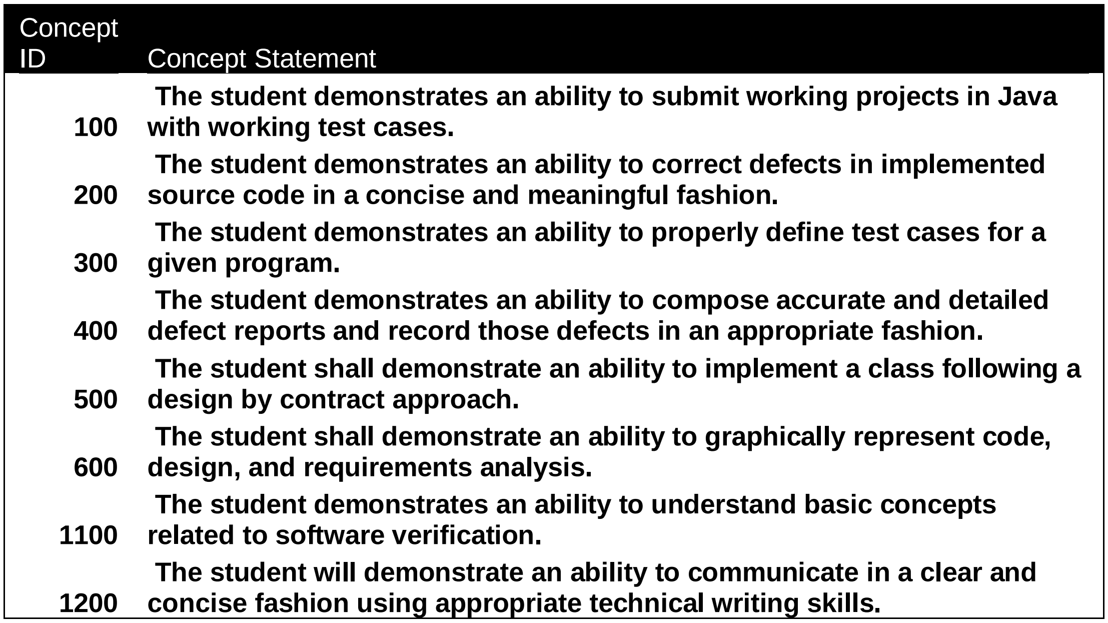
Figure 2: Course Catalog Entry

Developing SBG

The course itself has been taught for multiple years, but only recently has the idea of standards-based grading been considered. It should be noted that while this course currently uses many aspects of standards-based grading, currently, the course is not assessed purely using standards-based grading, as there are several key aspects which must be revised to call the grading system purely standards based.

In deciding to move toward standards-based grading, the first step was to detail the specific daily objectives[1] expected of students. In general, the development of daily outcomes is a best practice for teaching collegiate courses. The course itself had been built using outcomes from the beginning, so this step purely involved organizing the material in a slightly different fashion and into key areas. The lab assignments themselves also were developed with documented outcomes, mapping somewhat indirectly into the course outcomes. By having specific outcomes, the scope of exercises can be easier to controlled, and students can clearly see what they will be learning.

Once the course outcomes were organized, certain key concepts began to emerge which could be readily identified using laboratory exercises. These key concepts are shown in Figure 3. Note that in an ideal implementation of SBG, these concepts would have a closer mapping to course outcomes. However, in this case, the way that the course outcomes have been drafted does not necessarily allow a strong relationship here. There also are a couple of course outcomes that do not readily map into a lab-based environment, namely outcomes 1 and 2.



Concept ID	Concept Statement
100	**The student demonstrates an ability to submit working projects in Java with working test cases.**
200	**The student demonstrates an ability to correct defects in implemented source code in a concise and meaningful fashion.**
300	**The student demonstrates an ability to properly define test cases for a given program.**
400	**The student demonstrates an ability to compose accurate and detailed defect reports and record those defects in an appropriate fashion.**
500	**The student shall demonstrate an ability to implement a class following a design by contract approach.**
600	**The student shall demonstrate an ability to graphically represent code, design, and requirements analysis.**
1100	**The student demonstrates an ability to understand basic concepts related to software verification.**
1200	**The student will demonstrate an ability to communicate in a clear and concise fashion using appropriate technical writing skills.**

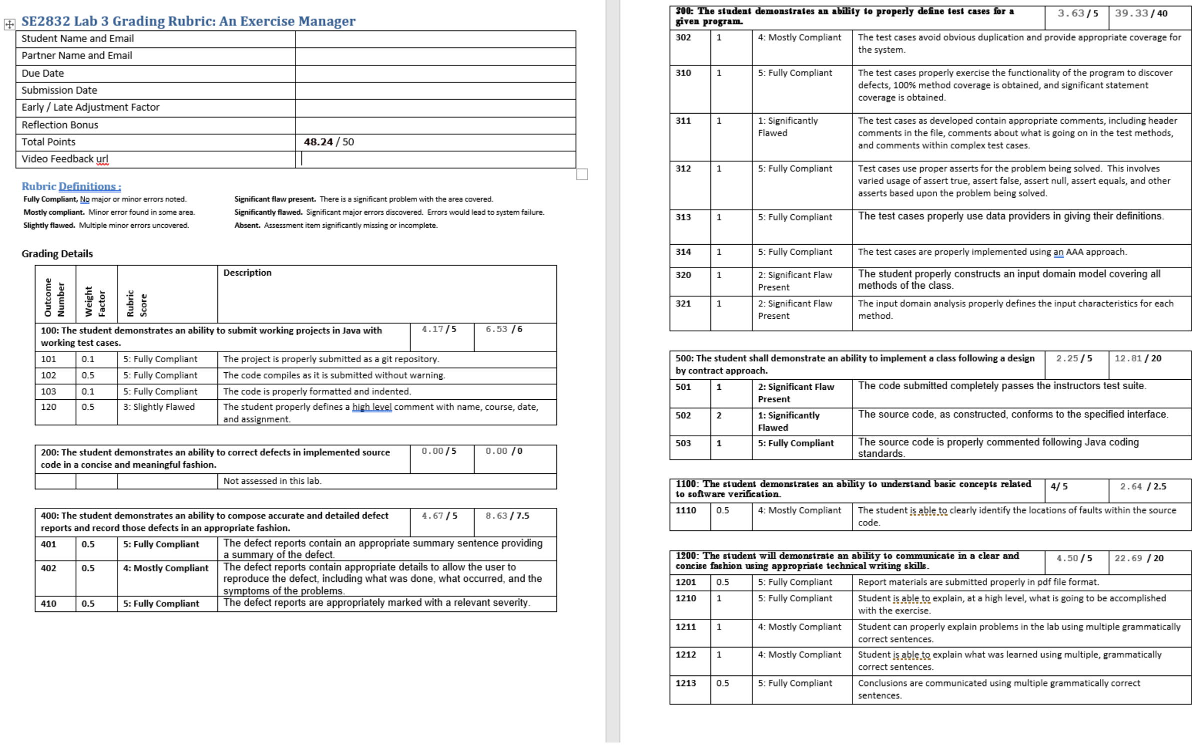
Figure 3: The key measurable concepts for Verification using a Standards Based Grading Approach

Once the key course concepts were identified, more specific sub concepts were developed that would be assessed one or more times throughout the course of the lab sequence. This again was an iterative process, incorporating the overall course outcomes, the lab activities, and other aspects. While ideally this would occur during the design of the course, in this case this process occurred iteratively over two years as the course ran and labs were being revised. Appendix A provides a current snapshot of the concepts and assignments in which they are assessed, as well

[1] or daily outcomes, depending upon terminology being used.

as the weightings placed on each concept in each assignment. Notice that not every concept is assessed in every lab. For maintainability purposes, each concept was given a number which uniquely identifies it. These numbers have no specific meaning, other than they tell which major concept each of the sub concepts is related to in the process.

With this being completed, the focus then moved to the individual feedback to students on each assignment. This was accomplished by generating a custom rubric sheet for each assignment. This rubric sheet lists both general and specific concepts assessed on the given assignment, as is shown in Figure 4. The rubric sheet provides the student with detailed feedback on their performance across each dimension that was assessed. It also provides the faculty member with substantial performance information about all students in the class and can be measured across assignments.



SE2832 Lab 3 Grading Rubric: An Exercise Manager

Student Name and Email	
Partner Name and Email	
Due Date	
Submission Date	
Early / Late Adjustment Factor	
Reflection Bonus	
Total Points	**48.24** / 50
Video Feedback url	

Rubric Definitions :

Fully Compliant, No major or minor errors noted.
Mostly compliant. Minor error found in some area.
Slightly flawed. Multiple minor errors uncovered.
Significant flaw present. There is a significant problem with the area covered.
Significantly flawed. Significant major errors discovered. Errors would lead to system failure.
Absent. Assessment item significantly missing or incomplete.

Grading Details

Outcome Number	Weight Factor	Rubric Score	Description		
100: The student demonstrates an ability to submit working projects in Java with working test cases.				4.17 / 5	6.53 / 6
101	0.1	5: Fully Compliant	The project is properly submitted as a git repository.		
102	0.5	5: Fully Compliant	The code compiles as it is submitted without warning.		
103	0.1	5: Fully Compliant	The code is properly formatted and indented.		
120	0.5	3: Slightly Flawed	The student properly defines a high level comment with name, course, date, and assignment.		
200: The student demonstrates an ability to correct defects in implemented source code in a concise and meaningful fashion.				0.00 / 5	0.00 / 0
			Not assessed in this lab.		
400: The student demonstrates an ability to compose accurate and detailed defect reports and record those defects in an appropriate fashion.				4.67 / 5	8.63 / 7.5
401	**0.5**	**5: Fully Compliant**	The defect reports contain an appropriate summary sentence providing a summary of the defect.		
402	**0.5**	**4: Mostly Compliant**	The defect reports contain appropriate details to allow the user to reproduce the defect, including what was done, what occurred, and the symptoms of the problems.		
410	**0.5**	**5: Fully Compliant**	The defect reports are appropriately marked with a relevant severity.		
300: The student demonstrates an ability to properly define test cases for a given program.				3.63 / 5	39.33 / 40
302	1	4: Mostly Compliant	The test cases avoid obvious duplication and provide appropriate coverage for the system.		
310	1	5: Fully Compliant	The test cases properly exercise the functionality of the program to discover defects, 100% method coverage is obtained, and significant statement coverage is obtained.		
311	1	1: Significantly Flawed	The test cases as developed contain appropriate comments, including header comments in the file, comments about what is going on in the test methods, and comments within complex test cases.		
312	1	5: Fully Compliant	Test cases use proper asserts for the problem being solved. This involves varied usage of assert true, assert false, assert null, assert equals, and other asserts based upon the problem being solved.		
313	1	5: Fully Compliant	The test cases properly use data providers in giving their definitions.		
314	1	5: Fully Compliant	The test cases are properly implemented using an AAA approach.		
320	1	2: Significant Flaw Present	The student properly constructs an input domain model covering all methods of the class.		
321	1	2: Significant Flaw Present	The input domain analysis properly defines the input characteristics for each method.		
500: The student shall demonstrate an ability to implement a class following a design by contract approach.				2.25 / 5	12.81 / 20
501	1	**2: Significant Flaw Present**	The code submitted completely passes the instructors test suite.		
502	2	**1: Significantly Flawed**	The source code, as constructed, conforms to the specified interface.		
503	1	**5: Fully Compliant**	The source code is properly commented following Java coding standards.		
1100: The student demonstrates an ability to understand basic concepts related to software verification.				4/ 5	2.64 / 2.5
1110	0.5	4: Mostly Compliant	The student is able to clearly identify the locations of faults within the source code.		
1200: The student will demonstrate an ability to communicate in a clear and concise fashion using appropriate technical writing skills.				4.50 / 5	22.69 / 20
1201	0.5	5: Fully Compliant	Report materials are submitted properly in pdf file format.		
1210	1	5: Fully Compliant	Student is able to explain, at a high level, what is going to be accomplished with the exercise.		
1211	1	4: Mostly Compliant	Student can properly explain problems in the lab using multiple grammatically correct sentences.		
1212	1	4: Mostly Compliant	Student is able to explain what was learned using multiple, grammatically correct sentences.		
1213	0.5	5: Fully Compliant	Conclusions are communicated using multiple grammatically correct sentences.		

Figure 4: Sample assignment rubric

Limitations of the Current System

At this point, there are several significant limitations to deploying a complete standards-based grading system in this course, even though the individual labs are graded in a manner that would be conducive to fully deploying standards-based grading. First and foremost, the campus LMS is in the process of being upgraded. The current system is not conducive to recording and presenting feedback to students and lacks the ability to weigh concepts in a meaningful fashion.

Thus, while the instructor has gone this far toward adopting standards-based grading, truly assigning final grades based upon individual concepts is not currently feasible. The new LMS system is supposedly more friendly toward this but is not scheduled to be deployed until the 2020-2021 academic year.

Second, in pure standards-based grading, the only measurements made are assessments of achievement. This, however, goes against some of the practices that encourage good student learning and prevent procrastination, such as an early submission bonus which encourages students to start assignments before they are due. [12] [13] The current rubrics still have an early performance bonus because it does encourage students to start earlier and work through assignments, and it also is quite popular with students. But it does go against pure standards-based grading. Another area is a reflection bonus which is used in a similar manner. We know as educators that one way our students learn is by reading our feedback. In the language arts and other fields, it is common for faculty to provide purely constructive feedback on submissions, to which the students make a second submission incorporating that feedback into a better product. This is not feasible in the lab environment, so the next best mechanism is to offer students a small incentive to reflect on their project after receiving comments.

The third issue with pure standards-based grading is that this the grading right now only applies to lab assignments. The concepts documented are only those which apply to lab activities. There is currently no attempt to capture terminology and definitions, concepts which are essential to understanding software verification. It does not make sense to individually list each term or assign an assessment to each term, but the knowledge of terminology must be captured in grading. Similar issues arise with quizzes, whereby quizzes are not always mapped directly to measurable concepts in the same way that the labs have been mapped.

References

[1] C. A. Tomlinson and J. McTighe, Integrating Differentiated Instruction & Understanding by Design: Connecting Content and Kids, Alexandria: Association for Supervision and Curriculum Development (ASCD), 2006.

[2] D. Iamarino, "The Benefits of Standards-Based Grading: A Critical Evaluation of Modern Grading Practices," Current Issues in Education, vol. 17, no. 2, 2014.

[3] "Standards-Based Grading Overview," Active Grade, 2012.

[4] S. Ambrose, M. Bridges, M. DiPietro, M. Lovett and M. Norman, HHow Learning Works: 7 Research Based Principles for Smart Teaching, San Franscisco: Wiley, 2010.

[5] M. Townsley, "What is the Difference between Standards-Based Grading (or Reporting) and Competency-Based Education," Competency Works, 11 November 2014. [Online].

Available: https://www.competencyworks.org/analysis/what-is-the-difference-between-standards-based-grading/. [Accessed 30 12 2019].

[6] L. Davis, "Standards-Based Grading: What to Know in 2019," Schoology Exchange, 13 2 2019. [Online]. Available: https://www.schoology.com/blog/standards-based-grading. [Accessed 15 12 2019].

[7] E. Lee, A. Carberry, H. Diefes-Dux, S. Atwood and M. Siniawski, "Faculty perception before and after implementation of standards-based grading," in Research in Engineering Education Symposium, 2017.

[8] A. Carberry, M. Siniawski, S. Atwood and H. Diefes-Dux, "Best Practices for Using Standards-based Grading in Engineering Courses," in ASEE Annual Conference, New Orleans, 2016.

[9] H. Diefes-Dux and N. Hicks, "Grader Consistency in using Standards-based Rubrics," in ASEE Annual Conference, Columbus, 2017.

[10] J. Wierer, "Standards Based Grading for Signals and Systems," in ASEE Annual Conference, Tampa, 2019.

[11] S. Acharya and W. Schilling, "Effective Active Learning Approaches to Teaching Software Verification," in ASEE Annual Conference, San Antonio, 2012.

[12] W. Schilling, "Using Performance Bonuses To Decrease Procrastination," in ASEE Annual Conference, Louisville, 2010.

[13] R. Bennett, W. Schleter, T. Olsen and S. Guffey, "Effects of an Early Homework Completion Bonus," in ASEE Annual Conference, San Antonio, 2012.