Types of Important Observations to Support Project Assessment – A Case Study

Svetlana V. Drchova, Joseph E. Hollingsworth, and Murali Sitaraman

Technical Report RSRG-13-07
School of Computing
100 McAdams
Clemson University
Clemson, SC 29634-0974 USA

September 2013

Types of Important Observations to Support Project Assessment – A Case Study

Svetlana V. Drachova
Limestone College
Computer Science
Gaffney, SC 29634
1-864-656-3444
sdrachova@limestone.edu

Joseph E. Hollingsworth
Indiana University Southeast
Computer Science
New Albany, IN
1-812-941-2425
jholly@ius.edu

Murali Sitaraman
Clemson University
School of Computing
Clemson, SC 29634
1-864-656-3444
murali@clemson.edu

ABSTRACT
Evaluation is critical component of all successful CS educational projects. One key benefit of evaluation is that it facilitates continuous improvement by helping pinpoint areas where there is room for improvement. Such pinpointing is difficult even in the case of projects where evaluation assumes a central role because it is often hard to factor out the impact of various elements that confound the results. This paper is a multi-year case study of our own experiences in evaluation. It identifies a variety of important observations that we have made in the context of our project that we believe will be useful in guiding other educational project assessment efforts. Clarifying the impact of various factors on project results also makes it easier for others interested in project results to replicate them elsewhere.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education—*computer science education*

General Terms
Management, Measurement

Keywords
Assessment, attitudes, concept inventory, performance-based learning outcomes, project evaluation

1. INTRODUCTION
The focus of this paper is the assessment process of a project that has as one of its goals that undergraduate computing students will learn specific concepts or skills. Often these concepts or skills are novel to the undergraduate curriculum by either being relatively new to the computing field or heretofore by having been relegated solely to graduate level education. Assessment of such a project is paramount for a number of reasons. High among these reasons includes being able to use the assessment results as feedback into the project for continuous improvement over time. Furthermore, if a project has received funding its principal investigators must be able to provide quantitative evidence that the funding provided to the project has produced positive results [1]. The main purpose of this paper is to identify various types of data supported observations that can be used by a project director—funded or not—for driving improvement and for justifying the project's benefits. We will list a number of such general observations and illustrate them with specific examples.

As a topic in itself assessment in education has received considerable attention, including journals devoted to presenting techniques for evaluation, their effectiveness, and the impact of various confounding factors. Whereas the goal of these efforts is general applicability, assessment results to show the benefits of specific techniques and tools in CS education are often the focus of presentations at SIGCSE. This paper makes CS educational assessment itself as its topic and it is based on our own experience over a 5-year period. It contains a variety of useful observations for CS educators who are infusing novel ideas in their classrooms.

Section 2 provides foundational material for one approach to systematically collecting data to support the observations. Section 3 discusses data collection details. Section 4 lists the general kinds of observations that a project director may wish to make along with specific examples to illustrate these observations. Section 5 concludes with some discussion.

2. FOUNDATIONS
In general assessment needs to be data driven. That includes data for measuring student learning as well as for measuring students' attitudes. With respect to collecting data for a particular subject area, one way to support the systematic collection of direct evidence of student learning is to base this data collection on a reasonably comprehensive inventory that lists the concepts and skills required of anyone working in that subject area. The examples provided in this paper are based on an inventory called the reasoning concept inventory (or RCI), which captures much of what needs to be taught to permit students to mathematically reason about the correctness of a piece of software so as to support the development of high quality software [2]. Technical results of the project are reported elsewhere [3,4,5,6,7] and are not directly relevant here.

Building on this approach with the inventory at the foundation, next comes performance-based learning outcomes. The learning outcomes refine the more general ideas captured in the inventory and more precisely identify the concepts and skills expected to be exhibited by the learner after instruction. The "performance-based" part of a performance-based learning outcome, has to do with choosing action verbs that describe the "performance" expected of a learner when asked to demonstrate that the concept or skill has been learned. These action verbs are key to helping instructors write quality assessments because these verbs will frequently form the backbone of a particular assessment question.

Furthermore, to support collecting data at the appropriate level of difficulty, Bloom's taxonomy is employed. Within the cognitive domain, the difficulty of a particular skill is considered to fall within one of the following six levels listed from lowest level of difficulty to the highest: knowledge, comprehension, application, analysis, synthesis, and evaluation. We found that having three levels satisfied our assessment needs and also reduced some of the complexity surrounding assessment, so our learning outcomes appear at one of the three levels: Knowledge-Comprehension (or KC—this combines the two lowest levels), Application-Analysis (or AA), and Synthesis-Evaluation (or SE). Educational researchers have developed comprehensive lists of action verbs that correspond to these cognitive levels and these verbs can then be employed in writing the performance-based learning outcomes. For example, by choosing a verb from the AA level, one can construct a learning outcome that expects the learner to demonstrate the ability to apply a particular concept or skill or to perform an analysis of a given situation or problem.

In Section 4, we introduce a number of observations about student learning that are supported by the analysis of the project data that was collected. This collected data came from assessment instruments written from performance-based learning outcomes, where those outcomes utilized action verbs at various levels of difficulty. All of the outcomes were based on our concept inventory.

3. DATA COLLECTION

This section explains how the Reasoning Concept Inventory, along with the learning outcomes and methods for their instruction, forms the basis for experimentation, data collection, evaluation, and improvement. Complete details may be found elsewhere [2]. The experimentation involves multiple undergraduate courses at 11 universities. The required IRB procedures were in place at two of the 11 universities but not in place at the other nine. However, first-hand reports from the adopting instructors at these nine universities indicate that students had a positive outcome learning the reasoning topics.

At Clemson where the appropriate IRB procedures had been executed, CPSC215 and CPSC372 were the targeted courses for adoption. CPSC215 is a sophomore-level Software Development Foundations course and CPSC372 is a follow-on junior-level course titled Introduction to Software Engineering. In both classes in the semesters prior to full-blown adoption of a portion of the RCI reasoning principles, small pilots were run to gain some initial experience. In CPSC215 adoption encompassed four weeks of the semester, and data was collected over six semesters of eight different sections. In CPSC372, approximately one third of the semester covered the reasoning principles and data was collected over four semesters in four different sections. For the remaining two-thirds of CPSC372 traditional software engineering topics were taught, which also factored into the data analysis (see Section 4.4). At Alabama, the IRB procedures were also executed, and CS315 (a junior-level software engineering course) was targeted. In CS315, three class periods per semester covered the reasoning principles, and data was collected over a three semester period.

While the collected data consists of student midterm and final examinations, and select assignments and quizzes that incorporated the RCI (reasoning) principles, all the data used for illustration in this paper come from final examination questions. Also note that the data of interest was gathered from specific questions based on specific learning outcomes, which in turn were based on specific RCI reasoning principles, and was not gathered from an entire test or quiz score.

4. TYPES OF OBSERVATIONS TO SUPPORT PROJECT ASSESSMENT

The data analysis provides evidence for various types of observations, ranging from ones that show the ability of students to learn new (reasoning) principles to ones that show positive attitudes to the new material. The observations are grouped into seven sections according to their relevance, and each is followed by a brief discussion. Before proceeding to the discussion of the observations, it needs to be emphasized that in our case the concept inventory and the learning outcomes are paramount to making conclusions about experimental data. Because the inventory of (reasoning) principles are divided into five areas, each of which is further subdivided into several levels, and learning outcomes on the appropriate level of difficulty, learning of each principle can be assessed with a high degree of precision. Being able to exactly pinpoint the deficiencies in particular areas of student learning guides the development of an effective intervention for the area in need. Table 1 provides a quick overview of the observations discussed in detail in subsections 4.1 through 4.7.

In the table below we identify a number of types of observations that a project manager may wish or need to make as part of a comprehensive assessment process. Each of these types of observations is illustrated by a specific observation that we were able to make about our project.

Table 1. An Overview of Observation Types

1. Observations related to students' learning of the principles
2. Observations related to pinpointing and conducting interventions
3. Observations related to difficulty of assessment questions
4. Observations comparing learning of new vs. traditional principles
5. Observations related to instructors teaching reasoning principles
6. Attitudinal observations to supplement direct evidence
7. Focus group observations to supplement direct evidence

4.1 Observation Type #1: Students Are Capable of Learning What Is Being Taught

Central to most undergraduate education projects is the idea that by some means, undergraduate students can learn material that has either never been systematically taught at the undergraduate level, or if it has been taught, that the students can learn the material more efficiently or at a deeper level than before. Consequently, it is imperative that such a project demonstrates that students are capable of learning what is being taught.

One method for demonstrating this is through the development of assessment instruments in the manner described in Section 2.

Illustrative example:

One of the learning goals of our project is to demonstrate that undergraduate CS are capable of learning the mathematics required to formally reason about the correctness of a piece of software, and to apply what they have learned to reason about software artifacts of the appropriate instructional level.

We provided instruction on these topics in CPSC372. The data was collected from four semesters: Fall 2010 to Spring 2012. Four

principles from our RCI inventory were assessed on the AA and SE levels of difficulty (from Bloom's taxonomy). Table 2 contains the collected data.

Table 2. Data Supporting Observation Type #1

	Difficulty Level	Semester: Fa10	Sp11	Fa11	Sp12
RCI #3.4.3	AA	94%	78%	89%	84%
RCI #4.1.1.2	SE	93%	79%	86%	84%
RCI #5.2.2	SE	61%	73%	76%	71%
RCI #5.3	SE	88%	88%	46%	86%

Discussion of Data:

The percentages in the rightmost column of Table 2 represent the class average for a particular assessment question. The project team members must identify a cutoff percentage for their project (or department/institution) with respect to making the claim that the students have learned the material. Seventy percent might serve as the cutoff if for example a department requires for passing that a student earns a 'C' or better (where a 'C' cutoff is at the 70% level). In Table 2, our project failed to meet the 70% cutoff for RCI #5.2.2 (row 3) for Fall 2010 and RCI #5.3 (row 4) for Fall 2011. Our project utilized this data to pinpoint what was believed to be causing problems and based on that analysis to subsequently develop interventions for process improvement (see Section 4.2).

A complementary metric that can also be used, illustrated in row 2 of Table 3 (in a subsequent section), is to look at the percentage of students who score at a certain level or better. For example, a project might aim for 80% of the students scoring at the 70% level or better on each of the assessment questions. This metric is an approach for tracking that a project-chosen percentage of the students (80% in this example) can demonstrate that they have learned the material at a reasonable level.

Finally, the RCI items that appear in the Table 2 are:

- RCI #3.4.3 – applying operation pre/post-conditions in the reasoning process
- RCI #4.1.1.2 – evaluating code by tracing/inspection utilizing pre/post-conditions
- RCI #5.2.2 – evaluating code for correctness by utilizing assumptions and obligations
- RCI #5.3 – synthesizing verification conditions (VCs) and applying proof techniques that utilize VCs

An example of a specific learning outcome related to RCI #3.4.3 at the SE (Synthesis-Evaluation) level in Bloom's taxonomy might be: *Write an ensures clause that precisely captures the behavior of an operation*. The verb "write" in this learning outcome describes the performance expected of the student.

4.2 Observation Type #2: Assessments Must Aid in Pinpointing Difficulties

To support continuous process improvement, project assessment must provide feedback to the PI as to where improvement of instruction can be made. Two aspects where assessment can support improving instruction include helping to pinpoint where improvements need to be made, and less directly what type of intervention might be suitable.

Illustrative example – Where to make an intervention?

In a CPSC215 we provided instruction on how to utilize pre/post conditions of called operations to create a reasoning table of assumptions and obligations in the client operation. Table 3 contains data (based on assessment instruments developed in the manner described in Section 2) from two back-to-back semesters of the same course, the first semester's data helped to pinpoint where an intervention was needed and the second semester's data helped to verify that the intervention made a positive impact.

It is important to note that all the data pertaining to the CPSC215 course used in this paper are (underperforming) students who are not exempt from the final. About a third of the students with 'A' grades just prior to the final were exempt.

Table 3. Data Supporting Observation Type #2

	Difficulty Level	Semester: Fa11	Sp12
RCI #4.1.1.2 RCI #5.2.2 *Class average*	AA	64%	79%
RCI #4.1.1.2 RCI #5.2.2 *% of students w/ ≥ 70%*	AA	50%	71%

Discussion of Data:

In the table the first row uses the class average metric in row one, while in row two the metric shows the percentage of students that scored 70% or better on the particular assessment (either multiple questions and/or partial credit were given). Since during Fall 2011 both of these metrics indicated student performance was not as high as desired, we decided to develop and apply an intervention. Sometimes, more longitudinal data is desired prior to taking action, especially when only one of the metrics is below standard, or when the metric is slightly below the desired success rate.

The intervention taken was the development of three short educational videos (approximately five to eight minutes each) that provide a step-by-step guide in the construction of a reasoning table for a simple code example [8]. In Spring 2012, these videos provided supplemental instruction outside of class to the instruction provided during class. We recognize that due to the many variables that cannot be controlled from one semester to another (e.g., student makeup of each section), there is by no means a guarantee that our intervention was solely responsible for an improvement of student behavior. However, achieving random assignment and controlling all variables other than the treatment condition is quite difficult in the educational setting, so we must often resort to a quasi-experimental design.

Discussion on determining the type of intervention:

Collecting data on student performance and its subsequent analysis is at the beginning of the process when making an intervention. The next step requires considering many of the non-treatment variables that cannot usually be held constant and that confound the analysis. These variables are related to aspects of the following: types of assessments, class periods, instructors, materials covered, and students. We developed a number of diagnostic questions to help determine if there might be a problem with the actual assessment, such as "Is this the first time the question is used?" or "Does the question correspond to the level of difficulty at which the concept was taught?" among others;

Table 13 in [1] lists each of these variables and corresponding questions for aid in determining the type of intervention.

Without a reasonable assessment process in place, at best it will be difficult determine where an intervention is needed, and at worst it will not even be known that an intervention is required.

4.3 Observation Type #3: Assessments Must Be at the Appropriate Level of Difficulty

Just having an assessment process in place is not sufficient. The members of the project must continue to monitor the assessments for being at the appropriate level of difficulty. Inappropriate assessments are not good for students or for meaningful evaluation. If an assessment is too simple it can lead to overconfidence, or simply be a waste of time for the student. Assessments that are too difficult can discourage students, or cause resentment. Furthermore, data collected and analyzed by the project based on inappropriate assessments can lead the team to take unnecessary or unwarranted interventions, or possibly to make inaccurate claims.

Performance-based learning outcomes based on Bloom's taxonomy (discussed in Section 2) then become a tool to be used by the project to initially cause to raise questions concerning at what level we expect students to perform, and to then provide the forum for discussing the level of difficulty of an assessment. If it is determined that the performance-based learning outcome and its assessment are both at acceptable levels of difficulty, but the performance measured is not, then assessment is appropriately leading us toward making an intervention as was discussed in Section 4.2.

Illustrative example

In CPSC215 we asked two questions on the final exam based on a KC-level of difficulty (lowest level). The questions asked students to identify the correct definition for "contract programming" and "loop invariant". Table 4 shows the result. There is nothing wrong with students scoring 100% on a particular question, however, when discussing this assessment data we realized that at the KC-level we could ask the students to perform other tasks that would possibly better reinforce these concepts. For example, we could ask a student to *defend* that a particular client has been engineered using contract programming principles, or to *summarize* the basic ideas surrounding loop invariants, etc. These two italicized verbs were selected from the many that appear in tables of "Bloom's verbs".

Table 4. Data Supporting Observation Type #3

	Difficulty Level	Semester: Sp12
RCI #4.2	KC	100%
RCI #4.3.2.1	KC	100%

At the other end of the spectrum, in the same course in Fall 2011, CPSC215 students were given a final examination which contained a question about using mathematical models for conceptualizing objects. The question dealt with RCI#3.3.1 (mathematical modeling for conceptualizing objects), and inadvertently involved an idea beyond the knowledge of the students. The material was taught on the KC-level of Bloom's taxonomy, but the assessment question was asked on the SE level. Almost none of the students got the answer right (see Table 5). (A teaching assistant who was teaching the course for the first time set this particular question.) The instruction was improved to the AA level and the question was changed from being at SE level to AA level. This adjustment helped, and next semester the difficulty level of the question was more appropriate, with 43% of students scoring 70% or higher. Though this is only the average performance of non-exempt students, it is still not an ideal situation and requires more investigation into additional interventions.

Table 5. Data Supporting Observation Type #3

RCI #3.4.3	Difficulty Level	Class Average	% students with 70% or higher
Fall 2011	SE	4%	0%
Spring 2012	AA	43%	43%

4.4 Observation Type #4: Assessments Must Show New Can Be Learned As Well As Old

In the Computer Science Curricula 2013 Ironman Draft Version 1 [9], it is stated, "… in several places we expect many curricula to integrate material from multiple Knowledge Areas", with examples given for introductory, systems, and parallel programming courses. If a project is attempting to integrate material into an existing course's curriculum then assessments that can measure how well students learn the new material with respect to the already existing, or traditional material will be of value.

An analysis of assessment data that shows that students learn the new material on par with the traditional material would suggest that the new material is appropriate for the targeted audience. On the other hand if the initial analysis indicates that the new material is not being learned as well, that might lead to a number of additional questions to be investigated as part of the current project or a future project. One example question is: Would instruction of the new material be received better if it were in its own course rather than being integrated?

Illustrative example

In CPSC372 we integrated instruction of reasoning concepts into a software engineering course with two thirds of the course devoted to traditional topics such as requirements analysis and design. In our final exam we assessed the important parts of both of these areas, the reasoning concepts and the traditional software engineering concepts. Table 6 shows that for two separate semesters class averages on these two different areas almost mirroring each other. Without such data, for example, one could reach conclusions that a new approach is working better or worse, when in fact the result might be affected by the student population.

Table 6. Data Supporting Observation Type #4

	Semester: Fa10	Sp11
Class average on reasoning concepts	85%	79%
Class average on traditional SE concepts	85%	78%

4.5 Observation Type #5: Assessments Must Account for Instructor-Related Variations

Dissemination and adoption are often important goals of CS education projects. So if new teaching techniques, or techniques for teaching new conceptual material are developed by the project, then dissemination through adoption by other instructors of the new conceptual material as well as the teaching techniques will almost certainly be a project goal. In this situation, results from assessment will add value if the results can demonstrate how well students perform when learning from an instructor who has little prior experience with the new techniques or with the new material.

To support gathering of data in this situation, the project needs to recruit a second group of instructors who are less familiar with the new concepts and/or new teaching techniques and then arrange for this group of instructors to teach the new concepts/techniques. The project at a minimum also needs to provide this group of instructors supporting instructional material, already written assessments, possibly some face-to-face training with the material, and then some support during the semester, e.g., answering instructor questions, etc. The data gathered by the project-provided assessments can then be used to analyze how well the students were able to learn from this group of instructors. Finally, if the group is permitted to teach the material over multiple terms, then longitudinal data can be gathered and utilized.

Illustrative example

Table 7 presents the assessment data from CPSC215 collected in Fall 2012 semester. Instructor 1 had been teaching RCI reasoning principles since 2008. Instructor 2 was an experienced computer science instructor who taught undergraduate CS courses for a number of years. Instructor 2 taught the RCI reasoning principles for the first time in Fall 2012. The six reasoning principles listed were assessed through various questions at the end of the semester in both sections. Students in the section taught by Instructor 2 scored comparable to those in the section taught by Instructor 1, and in some instances scored higher. We believe that this success is at least due in part to the fact that experienced educators are already familiar with good foundational teaching methods and when provided with good instructional and assessment materials, that they too can achieve satisfactory results with new material and techniques.

Table 7. Data Supporting Observation Type #5

	Difficulty Level	Fall '12 Class Average: Instructor 1	Instructor 2
RCI #3.4.3.2	AA	44%	60%
RCI #4.1.1.3	KC	71%	75%
RCI #4.2.2.1	KC	95%	88%
RCI #5.2.2.1	KC	81%	100%
RCI #5.3.2	AA	48%	63%
RCI #5.2.2	AA	81%	85%

If, however, this type of assessment seems to point in the other direction, i.e., toward students having a harder time learning from instructors who are unfamiliar with the material, then this might help the project to find ways to modify existing instructional materials or develop additional supplemental instructional materials to either be a direct aid to the student or to the instructors themselves. Furthermore, if longitudinal data can be collected, then that data can be analyzed to determine if learning improves as the unfamiliar instructor becomes more familiar with the material.

4.6 Observation Type #6: Attitudinal Assessments Are a Must

Attitude measurement is important because it is well known in social psychology that attitudes not only affect behavior (i.e., they are predictive of future behavior), but that behavior can affect attitudes. The attitudes of students will be affected by the project's instruction and interventions, so it is important to assess the attitudes of students along with how well they have learned the material.

Illustrative example

We conducted attitudinal surveys in both the sophomore-level and junior-level courses mentioned in this paper. A questionnaire was administered to students in each section at the beginning and end of the semesters for these courses. This summative survey data and the full version of the survey, along with the consent form that students receive prior to their participation, can be found in [2].

The questionnaire assesses the student attitudes on software engineering topics. Statistical tests were used to compare students' attitudes before and after taking the class in which the new topics were taught. The results from CPSC215 showed a significant positive change in students' conception of how to build high quality software after taking the class. Results from CPSC372 showed a significant positive change in students' view of precise mathematical descriptions for developing correct software.

The results of the attitudinal surveys indicate that the attitudinal changes occurred exactly in the areas emphasized in each course. The sophomore-level CPSC215 course taught basic concepts of software design, and the junior-level follow on course CPSC372 taught more advanced software engineering skills, including specifications, contracts, etc. Such significant attitude changes cannot be taken for granted as noted earlier; students may "learn" a topic to achieve better grades without necessarily changing their attitudes towards the importance of those topics.

4.7 Observation Type #7: Focus Group Studies Are a Must

While quantitative evidence is central for assessment, ideally, it must be supported with qualitative evidence as well. Student and instructor focus groups can both be useful, depending on the size and scope of the project. Here, we report on results from a focus group meeting that was conducted with the graduate teaching assistants who taught CPSC215, following suitable IRB procedures. The goal of this meeting was to understand what new principles were actually introduced in the course, and to take inventory of the successes and challenges.

The instructors provided useful feedback, and the most relevant items are discussed below. The full transcription of the meeting is available in [2]. It was discovered that a large number of RCI principles were covered in the course. Each instructor spent about 3 weeks of the course teaching the new principles. Though all the instructors covered an almost identical set of new principles, each instructor introduced the new topics where they thought it logically fit within the traditional material. When asked to place a mark in an inventory table of principles next to the items that they have taught/tested and taught/not tested, the marked sets from

each instructor were almost identical. Some of the principles were covered at the KC level of the Bloom's taxonomy, and others at more advanced levels.

Though teaching the topics was a new endeavor to these instructors, they only experienced minor difficulties teaching them. They found the guidelines, instructional materials, and assessment questions that were provided to them useful. We can conclude with confidence that even a novice instructor can be successful in teaching these new topics. With minimal guidance, instructors can tailor assessment questions to meet their individual teaching styles and the material coverage level.

The difficulties of incorporating new topics into the existing course material were also discussed. Though one of the instructors indicated that introduction of reasoning topics initially seemed like a "hard left turn", the others did not see a challenge in incorporating the topics into the existing curriculum.

Another important conclusion was that the students were able to learn new topics, and performed comparably to the traditional topics. This qualitative evidence collected at the group meeting correlates with the research data discussed in the earlier sections. For example, both indicate that students are capable of learning reasoning topics just as well as traditional ones. Some challenges were noted in the focus group meeting as well, such as a lack of prerequisite knowledge of mathematics for some students.

5. DISCUSSION

While there is much discussion in the general education literature on aspects of assessment (e.g., [10]), few detailed studies on the topic itself are available in a computer science context. Moskal, Lurie, and Cooper [10], for example note the improvement in performance and attitudes for a curriculum designed to benefit "at risk" majors. Dorn and Elliot [12] have found using a pre/post survey that student attitudes can change throughout the course of a semester and that the impact of pedagogy on student attitudes can be measured. A panel discussion has raised various issues in assessing performance at different colleges [13].

The goal of this paper is to help educators confirm through assessments the benefits of teaching new principles or using new approaches, and in the process consider a host of factors that affect the results of such assessments. We have provided a detailed case study with actual examples and supporting data, collected over a period of five years as a guide for other CS educational efforts. Ultimately, no matter how good the new approaches might be, without proper validation through critical evaluation, they are unlikely to be replicated elsewhere.

6. ACKNOWLEDGMENTS

We thank members of our research groups for their contributions to the contents of this paper. This research is funded in part by the NSF grants CCF-1161916, DUE-1022191, and DUE-1022941.

7. REFERENCES

[1] Stevens, F., Lawrenz, F., and Sharp, L. 1993. *User-Friendly Handbook for Project Evaluation - Science, Mathematics, Engineering and Technology Education*, Ed. J. Frechtling. NSF 93-152, available at: http://www.nsf.gov/pubs/2002/nsf02057/start.htm

[2] Drachova, S.V., 2013. Teaching and Assessment of Mathematical Principles for Software Correctness Using a Reasoning Concept Inventory, Ph.D. Dissertation, Clemson University.

[3] Cook, C.T., Drachova, S.V., Hallstrom, J.O., Hollingsworth, J.E., Jacobs, D.P., Krone, J., and Sitaraman, M., 2012. A systematic approach to teaching abstraction and mathematical modeling. In *Proceedings of the Seventeenth ACM annual conference on Innovation and Technology in Computer Science Education*, Haifa, Israel 2012, ACM, 357-362. DOI= http://dx.doi.org/10.1145/2325296.2325378.

[4] Cook, C.T., Drachova, S. V., Sun, Y-S., Sitaraman, M., Carver, J., and Hollingsworth, K. E., "Specification and Reasoning in SE Projects Using a Web-IDE," *Proc. 26th Conference on Software Engineering Education and Training*, IEEE, 2013.

[5] Krone, J., Baldwin, D., Carver, J.C., Hollingsworth, J.E., Kumar, A., and Sitaraman, M., "Teaching Mathematical Reasoning Across the Curriculum", *Proc. 43rd ACM Technical Symposium on Computer Science Education*, ACM, 2012, 241-242.

[6] Leonard, D.P., Hallstrom, J.O., and Sitaraman, M. 2009. Injecting rapid feedback and collaborative reasoning in teaching specifications. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, 524-528. DOI=10.1145/1508865.1509046 http://doi.acm.org.oberon.ius.edu/10.1145/1508865.1509046

[7] Sitaraman, M., Hallstrom, J.O., White, J., Drachova-Strang, S.V., Harton, H.K., Leonard, D., Krone, J., and Pak, R. 2009. Engaging students in specification and reasoning: "hands-on" experimentation and evaluation. In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*. ACM, New York, NY, USA, 50-54. DOI=10.1145/1562877.1562899 http://doi.acm.org.oberon.ius.edu/10.1145/1562877.1562899

[8] Hollingsworth, J.H., 2012. SIGCSE Workshop 2012 Instructional Video Series. http://www.cs.clemson.edu/group/resolve/teaching/ed_ws/sigcse2012/index.html

[9] IEEE/ACM Computer Science Curricula 2013, Ironman Draft V1, 2013. http://ai.stanford.edu/users/sahami/CS2013/ironman-draft/cs2013-ironman-v1.0.pdf

[10] Dunn, K. E. and Mulvenon, S. W. 2009. A Critical Review of Research on Formative Assessment: The Limited Scientific Evidence of the Impact of Formative Assessment in Education. In *Practical Assessment, Research, and Evaluation*, Volume 17, Number 7, 1-11; http://pareonline.net/pdf/v14n7.pdf

[11] Moskal, B., Lurie, D., and Cooper, S. 2004. Evaluating the effectiveness of a new instructional approach. In *Proceedings of the 35th ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, 75-79.

[12] Dorn, B. and Elliott A.T. 2013. Becoming experts: measuring attitude development in introductory computer science. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, 183-188. DOI=10.1145/2445196.2445252 http://doi.acm.org.oberon.ius.edu/10.1145/2445196.2445252

[13] Sazawal, V., Schwarm, S., Goldner, B., Gellenbeck, E., and Zander, C. 2003. Assessment of Student Learning in Computer Science Education, The Journal of Computing Sciences in Colleges, Volume 19, Number 2, 39-42.