

Instructional Design + Knowledge Components: A Systematic Method for Refining Instruction

Author 1
Department
Institution
City, State
email

Author 2
Department
Institution
City, State
email

Author 3
Department
Institution
City, State
email

Author 4
Department
Institution
City, State
email

Author 5
Department
Institution
City, State
email

ABSTRACT

This paper reports on a systematic method used to improve an existing unit of instruction. The method is distinctive in combining steps of instructional design with "knowledge components" from a cognitively-based framework of learning. Instructional design is used to develop assessment instruments that incorporate information about student misconceptions. The method uses the assessment instruments to evaluate student performance and learning gains, while statistical analysis evaluates the quality of the instruments themselves using measures of difficulty and discrimination. Fine-grain insight into possible improvements is enabled by the knowledge components implicated by the assessment. The method is illustrated and evaluated by applying it to a unit of instruction on collection-based iteration in a computational thinking class. Data gathered during this evaluation highlights a number of opportunities within the unit to refine the instruction.

CCS CONCEPTS

• **Applied computing** → **Education**; Learning management systems;

KEYWORDS

Knowledge Components, CS1, CS Education, Assessment, Instructional Design

ACM Reference format:

Author 1, Author 2, Author 3, Author 4, and Author 5. 2018. Instructional Design + Knowledge Components: A Systematic Method for Refining Instruction. In *Proceedings of ACM SIGCSE conference, Baltimore, Maryland USA, February 2018 (SIGCSE'18)*, 6 pages.
https://doi.org/10.475/123_4

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGCSE'18, February 2018, Baltimore, Maryland USA

© 2018 Copyright held by the owner/author(s).

ACM ISBN 123-4567-24-567/08/06.

https://doi.org/10.475/123_4

1 INTRODUCTION

Revision is a crucial component of course design. Frequently, however, the original instruction and accompanying assessments have not been developed using any kind of formal process, but instead were developed from informed experience while using informal means. In other cases, the instruction was "inherited", often with overlapping contributions and viewpoints from various instructors. Of course, experienced instructors often have valuable insight into the misconceptions held by students. When instructors have the opportunity to refine a curriculum, or even rebuild it from the ground up, the process can benefit from this prior knowledge. Still, using ad-hoc processes can limit the effectiveness of a revision opportunity in the same way that large software projects suffer without guidance from formal design methodologies.

Course revision and development can benefit from using formal methods of Instructional Design to guide development, although in our experience many computing instructors are unaware of these methodologies. The field of Instructional Design has been around for many decades, and a variety of models and frameworks exist that systematize the design of learning experiences [2, 4, 7]. In a few cases, researchers in computing education have used elements of formal ID models in their work, such as Needs Assessments [9] or comparing instructional strategies [3]. Some works even note the parallels of Software Engineering and Instructional Design [5, 6], yet few suggest targeted ways for computing educators to incorporate ID into their curriculum design efforts [1], suggesting a need within the literature for a well-described model suitable for computing education researchers.

In this paper, we describe our activities in the practical improvement of a targeted unit of existing instruction. This paper makes two contributions:

- The development of a method for improving a targeted unit of instruction that combines steps of instructional design with elements of a cognitively-based framework of learning and is particularly suited for the design of computing instruction, and
- The illustration and evaluation of the method as applied it to a unit of instruction on collection-based iteration in a introductory computing class.

The audience for this paper are instructors seeking an approachable and effort-conscious way of improving a specific part of their course. We believe the method is approachable; while it does require some reading and preparation, it does not require extraordinary training or overly-sophisticated statistical analysis. The method is effort-conscious, requiring a modest time investment. We estimate that creating the assessment instruments (the first step of the method) involved three of the co-authors working intermittently over several weeks, so conservatively two person-weeks of effort. The analysis step took four of the co-authors less than a week to conduct. Finally, the method is (at least in our view and use) focused on a “unit” of instruction (i.e., a single topic) which in our case was a unit lasting four class days on the topic of collection-based iteration. While the method can be used to incrementally improve an entire course, one unit at a time, we realize that in practice not all units in a course may require improvement. Rather, especially critical or under-performing units can be targeted. This work also makes a partial contribution toward the development of a concept inventory [10, 11], specifically for the topic of iteration.

We give an outline of our method and use this outline as a guide to the organization of the paper. The method has four steps:

- (1) Design assessment
 - identify instructional goals (Sec 2.1)
 - identify misconceptions (Sec 2.2)
 - conduct instructional analysis (Sec 2.3)
 - create assessment instruments (Sec 2.4)
- (2) Collect assessment data
 - administer assessment instruments (Sec 3.1)
 - tabulate student performance and learning gains (Sec 3.2 and Table 1 and Figure 2)
 - tabulate item analysis to identify item difficulty and discrimination (Sec 3.2 and Table 2)
- (3) Analyze student learning and assessment instruments
 - identify items with weak learning gains and valid assessments (Sec 4.1 and 4.2)
 - identify implicated knowledge components (Sec 4.2)
- (4) Improvement (Sec 4.1 and 4.2)
 - correct suspect items in assessment instruments
 - improve instruction (presentation, problems, feedback)

Roughly, the steps in the design assessment step are a subset of the Dick and Carey instructional design process [4], though specialized. For example, in our method, the Dick and Carey step of identifying instructional goals is limited to identifying the goal of the unit of instruction selected for improvement. The identifying misconceptions step is the Dick and Carey step of Learner and Context Analysis, focused on the misconceptions of the learners and parts of the context relevant to misconceptions. The instructional analysis and assessment instrument steps are the same as in the Dick and Carey model. We do not claim that the method is unique in conception, but a selection and integration of existing techniques.

A deficiency in many instructional design processes is a lack of a model of the learners’ understanding. To overcome this, the Instructional Design model that we describe leverages the cognitive modeling aspects of the Knowledge Learning Instruction Framework (KLI) [8] within a sub-set of a formalized Instructional Design

process [4], with an operational definition that a Knowledge Component (KC) is “an acquired unit of cognitive function or structure that can be inferred from performance on a set of related tasks” [8]. Examples of such cognitive functions/structures include, but are not limited to, concepts, skills, misconceptions, and facts [8]. KCs can be things that we want our students to know or misconceptions that students have in their head. For example KC020 (see Table 3) “You can’t add the list to a number.” Another possible KC a student could have (a misconception) is “You can filter lists using an assignment and a boolean operator.”

The steps to analyze student learning and the assessment instruments use standard statistical techniques to study the assessment data collected by administering the assessment instruments. Through this analysis, specific KCs are implicated by assessment items with reasonable difficulty and discrimination and with less than desired student performance or learning gains. Comparison across different assessment items refine the list of implicated KCs.

2 DESIGN ASSESSMENT

Assessment-driven improvement of instruction relies on assessment instruments that are directly relatable to the instruction. Some systematic instructional design process must be used to achieve this linkage. In our case, a subset of the Dick and Cary model [4] was used where some steps were narrowed or refined for the task.

2.1 Identifying Instructional Goals

We targeted one of the most difficult and important parts of our curriculum, collection-based iteration. Not only is collection-based iteration difficult for our learners, it is also central to our curriculum, which revolves around data science. Specifically, students must be able to use iteration to compute simple quantitative measures of list-oriented data (e.g., averages) and generate visualizations (e.g. histograms). Two instructional goals emerged: “Construct an algorithm that outputs a quantitative measure of the values in a given list” and “Construct an algorithm that produces a visualization based on the values in a given list”. Each goal expresses an observable skill that we expect our students to learn from the instruction. For brevity, only the first is considered in this paper.

2.2 Identify Misconceptions

An initial list of student misconceptions about collection-based iteration was developed through personal observation and semi-automated analysis. Personal observations in the classroom were done informally. In our course format, instructors walk around the classroom and help students who are engaged in active learning exercises with other students. The interaction with students helped build our intuition regarding misconceptions and student knowledge. Semi-automated inspection of previous student code supported our personal observations. The tools used in the inspection allowed the replay of the students’ steps in writing their code, comparison of their code to a model answer, and k-means clustering of similar student programs. For brevity, we do not discuss these tools. The misconceptions played a role both in identifying KCs (a misconception suggests one or more partially understood or misunderstood KCs) and in developing distractors on the multiple choice assessment questions. While we developed tools for the

semi-automated analysis of student code, this is not a requirement of using the method. Any information regarding student misconceptions, from formal or informal data sources, is useful.

2.3 Instructional Analysis

The next step is to perform Instructional Analysis whose major purpose is “...to identify the skills and knowledge that should be included in our instruction ...” [4, page 41] that leads students to achieve the identified instructional goal. This process involves identifying the major components of instruction, subordinate skills, and their relationships. For our instructional goal we determined the baseline knowledge we expected students to have at the beginning of the iteration unit. The units prior to iteration provide instruction on basic initialization, Boolean logic, if-else constructs, as well as a knowledge of the basic data types (string, integer, and float). Determining the baseline knowledge of the students is important to having well defined boundaries about what was, and was not, out of the scope of our instructional analysis.

Using our instructional goal, Bloom’s taxonomy, our list of misconceptions, the baseline knowledge, and previous course material as a guides, we created a rough task analysis diagram of skills we wanted our students to be able to perform. To break the instructional goal down into subtasks and steps, we went through the process of discerning what skills were being exercised and used to produce our learning goal. We used Bloom’s taxonomy as a guide to make sure we included items from lower cognitive levels, to higher cognitive levels. The list of misconceptions helped us identify items in the diagram. For example, we included “Identify a subtype of list” as a task because our observations showed that students would occasionally use strings instead of numbers when constructing a list. After establishing a hierarchy of tasks, we formalized our task analysis into an instructional design diagram. The diagram shown in Figure 1 is a final iteration of the diagram after multiple iterations of refining our performance objectives and tasks.

We also generated performance objectives for each task in the diagram. A performance objective identifies for a given problem what the student will be able to accomplish and with what accuracy. For example, for Task 1.1.1 “Construct an iteration property”, two performance objectives are:

- (1) Given a list explain what type of value the iteration property will have.
- (2) Given a list select the type of the iteration property from among several choices.

Many other performance objectives are possible for this task. The value of stating these performance objectives is that they lead directly to assessment questions. They are the linkage between the assessment data gathered later and the instructional steps.

2.4 Assessment Instruments

Our assessment instruments (the pretest and post-tests) was a multiple choice test that assessed the individual subtasks in the instructional analysis diagram. To generate the multiple choice questions, we drew upon the performance objectives defined earlier, using those that could be cast into a multiple choice form. The misconceptions that we found in our earlier analysis (semi-automated

inspection and observation) and refined in our instructional analysis drove the creation of the distractors we used for the test.

After creating the test, we used the answer choices to generate a formalized list of KCs as the student’s selection of an answer choice represents knowledge we can infer the student to have (or not have). Table 3 has a small sample list of KCs (full list provided on request). It is important to keep in mind that because KCs are inferred from student performance, different instructors will come up with different KCs due to both teaching differences and student population differences. Note that while the KCs were derived from the assessments we made, the distractors themselves were built upon the misconceptions that we observed and inferred our students to have, and as such are biased towards our student population. That being said, a large number of these KCs are likely transferable to other student populations, or may serve as a starting point for other instructors.

3 COLLECT AND TABULATE DATA

The data was collected using parallel pre- and post-tests administered before, during, and after the unit of instruction. The tabulated data allows both an analysis of student learning and an evaluation of the instruments used to measure that learning (i.e., the pre- and post- tests). Both analyses are needed because gains or deficiencies in student learning should not be ascribed on the basis of a suspect instrument.

3.1 Administer Assessment

The primary unit of instruction on iteration that we sought to improve took place over three consecutive class days. During this time, the students used a block-based programming language. Following this unit, the students transitioned to using a text language (Python) where they learned a more complex data structure (lists of dictionaries) and used iteration to process such data. We were interested in assessing the students learning within the block-based portion and also the transfer of this learning to the text-based portion. Therefore, multiple tests were constructed to assess student learning at different stages.

Each test has nine multiple choice questions. The ordering and content of the multiple choice questions remained the same, but the details of the questions (e.g. summing distances vs. summing prices) and the ordering of the answer choices changed between tests. The tests were administered in an online format via a learning management system where each question was presented one at a time, with students not being able to go back to a previous question. This format was used because later questions in the test can give away answers to previous questions in the test (e.g., the code in a later question could give the answer to an earlier recall question).

The assessment consisted of one pretest and three post-tests. A pretest was given at the end of the class prior to the onset of unit instruction. The pretest established each student’s baseline knowledge on the material being taught. The first post-test, sometimes called an embedded post-test, was given at the beginning of the third day of instruction. The second post-test was given two class days after the first post-test. The first two post-tests assessed the student learning within the block-based language portion and used a block-based representation of code for the questions and answer

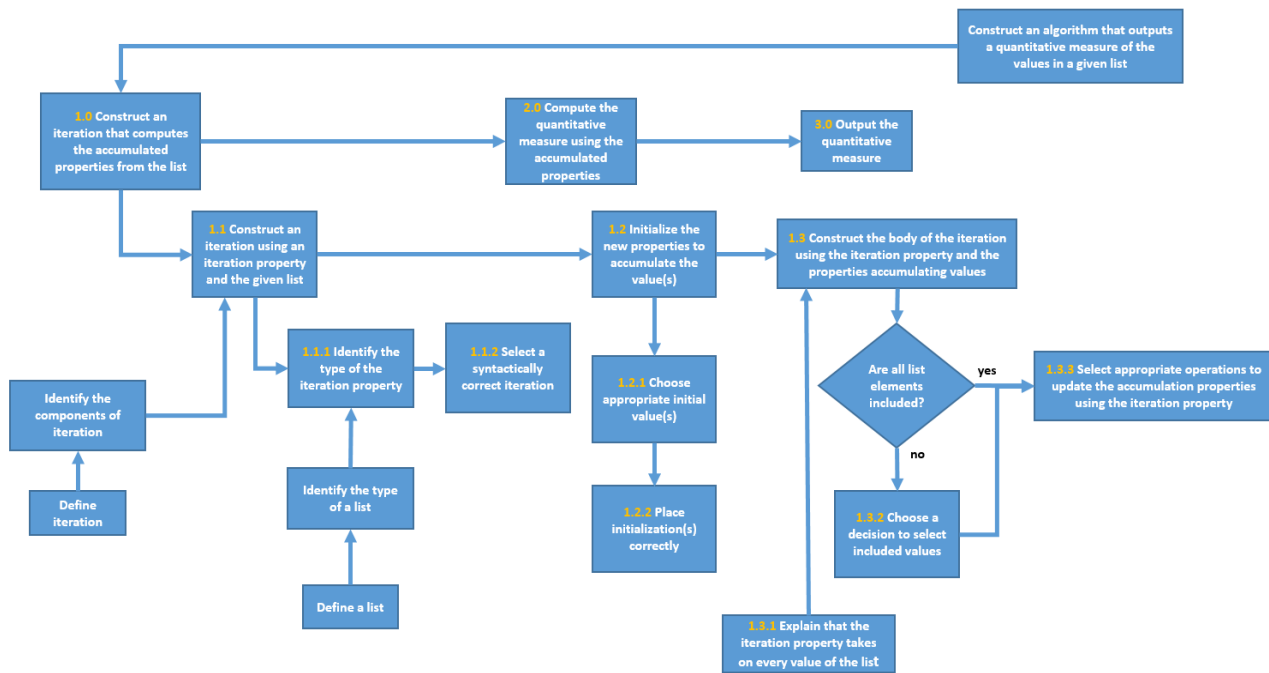


Figure 1: Final iteration of Instructional Design Diagram

choices. The third post-test, sometimes called a far transfer test, was given eight class days after the second post-test. This post-test was administered after the students had completed the Python coding portion of the class and was administered using the text representation for questions and answer choices. This post-test assessed the transfer of the student learning from the earlier instruction in a book-based form to a text-based form.

The tests were administered over two sections of the class that each met twice a week. The number of consenting participants were 67 for the pretest, 75 for post-test 1, 72 for post-test 2, and 69 for post-test 3 with 55 students participating in all three tests.

All tests were planned to be administered in class. However, an issue with the learning management system for one section necessitated the pretest for that section to be administered as a take home test. To insure that the difference in pretest administration did not invalidate the outcome measure we conducted an independent 2-group Mann-Whitney U test between the two sections. The p-value of 0.8157 suggests that there were no significant differences caused by taking the test at home instead of in the classroom.

3.2 Tabulate Data

Table 1 presents the data on student learning showing the percent of student answering each question correctly on each test. Learning gains were calculated as $\frac{post-pre}{1-pre}$ where pre and post represent the percentage of students who correctly answered each assessment question. The percent correct is an absolute measure of student performance while the learning gains is a measure relative to the students' starting (pretest) point. The learning gains are presented graphically in Figure 2.

Table 2 presents the data for the assessment of the three post-test instruments on an item (i.e., question) by item basis. The item difficulty is a proportion of collective correct responses. The discrimination ability of post-test items is characterized through summary correlation between a continuous and binary variable (overall outcome in relation to item-level association).

The assessment data can be described as follows. Based on response items of the post-test, question 1 showed a proportional decline in post-test 1 and post-test 3 with a slight proportional increase in post-test 2 in relation to the pretest. Questions 2-9 have notable increases in learning outcome while there was no detectable learning gain for question 1 across the 3 tests. Based on item difficulty, questions 2 and 5 had the highest proportional outcome on the post-test, reflected in the student learning gains on post-test 1, post-test 2 and post-test 3. Questions 3 and 6 had very little change in learning gains (average change of less than 0.03), and were identified by the portion of correct responses as items of lower difficulty, but with stronger discriminative power. Question 4 has a comparatively higher level of post-test difficulty in relation to questions 2, 3, 5, 6, and 9 and has moderate discriminative power (summary correlation of 0.599). Question 7 shows reasonably high learning gains on post-tests 2 and 3 (0.834 and 0.914 respectively), but shows much smaller progressions of learning gains in the first post-test. The resulting difference in learning gains is highlighted in the discussion section. Questions 8 and 9 had strikingly similar behaviors in their learning gains curves (see Figure 2) as well as their interesting learning gain drops on post-test 3 and are also highlighted.

Table 1: Learning Outcomes

Question	Percent of students correct				Learning Gains		
	pre	post1	post2	post 3	post1	post2	post 3
q1	0.646	0.547	0.708	0.594	-0.279	0.177	-0.145
q2	0.633	0.947	1.000	1.000	0.855	1.000	1.000
q3	0.329	0.947	0.931	0.942	0.921	0.896	0.914
q4	0.241	0.693	0.708	0.797	0.596	0.616	0.733
q5	0.709	1.000	1.000	1.000	1.000	1.000	1.000
q6	0.519	0.907	0.875	0.899	0.806	0.740	0.789
q7	0.329	0.467	0.889	0.942	0.205	0.834	0.914
q8	0.367	0.507	0.819	0.725	0.221	0.715	0.565
q9	0.443	0.773	0.931	0.812	0.593	0.875	0.662

Table 2: Item Analysis

	Average		Standard Deviation	
	Proportion Correct	Discrimination	Proportion Correct	Discrimination
Q1	0.616	0.588	0.068	0.040
Q2	0.982	-0	0.025	-0
Q3	0.940	0.222	0.007	0.066
Q4	0.733	0.599	0.046	0.105
Q5	1	-0	0	-0
Q6	0.893	0.546	0.013	0.111
Q7	0.766	0.398	0.213	0.127
Q8	0.684	0.659	0.131	0.114
Q9	0.838	0.356	0.067	0.044

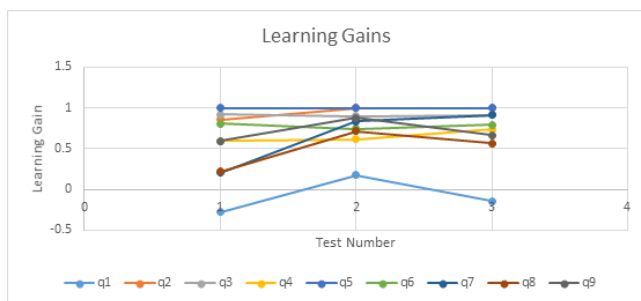


Figure 2: Learning Gains by Question

4 ANALYZE DATA

The tabulated assessment item data is analyzed in two steps. First, areas of possible improvement are identified. Assessment instrument improvements are indicated by low quality questions (i.e., those with problematic difficulty or discrimination). Improving these questions leads to a better assessment instrument. Instructional improvements are indicated by high quality questions where materially better student performance is possible. Second, for instructional improvements the related KCs are identified.

4.1 Identify Possible Improvements

Question 1 appears to be of low quality. The level of correct responses is inconsistent and low and the learning gains are very poor (Table 1). The overall level of performance is the lowest of all items (Table 2). We are not sure if there is some ordering effect that affected this question and distorted the learning outcomes. Additionally, the content for this question was based on identifying data types which limited developing parallel questions on each test as the only suitable types for the distractors were float, integer, and string. It is also possible that our instruction on types was insufficient. Analysis of this question highlights a need for item revision.

Questions 2, 3, and 5 (all with learning gains consistently above 0.855 and never dropping) leave little room for improvement. Question 6 has acceptable learning gains by post-test 3 (0.789). Questions 7 through 9 have low learning gains on the first post-test because the material for those questions was not yet taught. Question 7 showing good learning gains on post-tests 2 and 3 (0.834 and 0.914). Questions 8 and 9 have significant drops in learning gains between post-tests 2 and 3 (0.715 to 0.565 and 0.875 to 0.662 respectively). One hypothesis is that the introduction of dictionaries between post-tests 2 and 3 resulted in confusion on those questions. This suggests that materials involving dictionaries should be a focus for future improvement.

Question 4 is a high quality questions indicating an area of improvement in student performance. The students showed lower learning gains than we would have liked; learning gains were 0.596, 0.616, and 0.733 across the three post-tests respectively. The low learning gains flags the particular question as containing concepts that our students are not mastering. The fact that 73% of students get this question correct, and the fact that it's correlation to test score is 0.599, we can tell that the question itself is a fairly balanced discrimination value appropriate to its difficulty, meaning that the content of the question itself, and not the clarity of it, is something we need to address in our instructional material.

In summary, we have identified an assessment item to improve (Question 1), a topic for the future application of our method (dictionaries as indicated by Questions 8 and 9), and a current topic study further (Question 4).

4.2 Identify Knowledge Components

Exploring the KCs of a question, like Question 4, that points to possible instructional improvements allows for a fine-grain understanding of precisely what in the instruction needs to be improved. For brevity, we'll focus on answer choice "a" for Question 4 (see Figure 3) that was one of the two most commonly selected distractors. We want to know what KCs led students to select this distractor. From answer choice "a", we can infer that the students have the following KCs (see Table 3): 6, parts of 18, and to some confidence, 23. The missing pieces of knowledge are other subparts of KC 18 as well as KCs 12 and 22.

By identifying the KCs, we have a list of specific ideas against which we can review our instruction. For instance, we know the student has at least some partial competency with regards to KC 18. So an instructional improvement from the syntactic perspective

When using the iteration shown below to compute the sum of the numbers in the distance_list, which of the following is the correct statement to be in the body of the iteration?

```
distance_sum = 0
for distance in distance_list:
    pass

a: distance_sum = distance_list + distance
b: distance = distance_sum + distance
c: distance_sum = distance_sum + distance_list
d: distance_sum = distance_sum + distance
e: distance_list = distance
f: distance_sum = distance_sum + 1
g: distance_sum = distance_list
```

Figure 3: Question 4 template

Table 3: Knowledge Components for Question 4

KC Number	Description
KC006	The iteration property takes on each value of the list
KC012	The sum and the list are two different items
KC018	Update for sum is sum = sum + iteration_ property (has 6 different sub parts)
KC020	You can't add the list to a number
KC022	The list is not used in accumulation
KC023	The list is not used as an accumulator

is to clarify the basic accumulation pattern for summing a list using iteration. However, another approach, using KC 12, is to make clearer how a list is distinctly different from a sum, or a way of summarizing a list. A third approach would be to use direct instruction: “don’t use the list in for loop”. A fourth approach is to include in our programming environment tests for this misconception and provide immediate feedback linked to “just-in-time” instruction.

In summary, we have identified four different approaches to address the misconceptions revealed by the KCs in Question 4. Other guidance could be gained by examining other associated KCs in this question and studying how these same KCs appear in other questions. For example, Question 3 also contains KC 12 but performance on Question 3 question is on the upper end (learning gains averaging .9). Does this mean that students generally have KC 12 and the problem is elsewhere? Or does it mean that KC 12 in the context of accumulation is problematic? Further analysis can reveal the answer.

5 CONCLUSION

In this paper we have presented a pragmatic method for improving instruction by combining KCs from a cognitively-based framework with selected steps from a commonly referenced model of instructional design.

This four-step method begins by developing an assessment of the instruction targeted for improvement. In this step, a limited number of formal and informal methods are combined to define the high-level goal of the instruction, break-down the instruction into a set of related steps (visualized in an instructional diagram), identify student’s misconceptions using whatever means are at hand

- including experienced observation, and use all of this information in a deliberate way to create assessments that can be directly related to the instruction. We have reported our own experience on using this method to develop an assessment for a unit of instruction on collection-based iteration.

The second step applies the assessment during the teaching of the material and tabulating this data to derive measures of performance, learning gain, difficulty and discrimination. Calculating these measures is straightforward and does not require statistical sophistication. We have reported our experience explaining how we structured our assessment test to gain measures during the primary class days when the material was presented in a block-based language and also a later test to gauge the transfer of this learning to a text-based language. We encountered the usual situation where the administration of one of the tests did not go as planned and we discussed how we recovered from this problem.

The third step analyzes the assessment data to identify weakness in the instruction and specific “knowledge components” that students have not sufficiently mastered. We reported our efforts to relate the assessment to improving the questions on the assessment itself and how one question pointed us to an area where we could improve our instruction. We are now in the process of performing the fourth step - improving the instruction and the assessment.

We have shown how we have applied this method to our instruction of collection based iteration in a computational thinking course. We hope that this work better informs the community about the possibility of using principled methods to assess and revise critical topics with reasonable level of effort.

ACKNOWLEDGMENTS

This work is supported in part by National Science Foundation grants DUE 1624320, DUE 1444094, and DGE 0822220.

REFERENCES

- [1] Austin Cory Bart and Clifford A. Shaffer. 2016. Instructional Design is to Teaching As Software Engineering is to Programming (*SIGCSE '16*). 240–241.
- [2] Robert K Branson, Gail T Rayner, J Lamarr Cox, John P Furman, and FJ King. 1975. *Interservice procedures for instructional systems development. executive summary and model*. Technical Report. DTIC Document.
- [3] Michael E. Caspersen and Jens Bennedsen. 2007. Instructional Design of a Programming Course: A Learning Theoretic Approach (*ICER '07*). 111–122.
- [4] Walter Dick, Lou Carey, James O Carey, and others. 2001. *The systematic design of instruction*. Vol. 5. Longman New York.
- [5] Ian Douglas. 2006. Issues in software engineering of relevance to instructional design. *TechTrends* 50, 5 (2006), 28–35.
- [6] J. Philip East. 2004. Applying Software Design Methodology to Instructional Design. *Computer Science Education* 14, 4 (2004), 257–276.
- [7] Peter J Esseff and Mary Sullivan Esseff. 1998. Instructional Development Learning System (IDLS). (1998).
- [8] Kenneth Koedinger, Albert T Corbett, and Charles Perfetti. 2012. The Knowledge-Learning-Instruction Framework: Bridging the Science-Practice Chasm to Enhance Robust Student Learning. *Cognitive science* 36.5, Article 5 (2012), 757-798 pages.
- [9] Nasir I. Mohd, Azilah N. Nor, and Naufal U. Irfan. 2010. Instructional Strategy in the Teaching of Computer Programming: A Need Assessment Analyses. *TOJET : The Turkish Online Journal of Educational Technology* 9, 2 (2010).
- [10] Cynthia. Taylor, Daniel Zingaro, Leo Porter, Kevin C. Webb, Cynthia Bailey Lee, and M. Clancy. 2014. Computer science concept inventories: past and future. *Computer Science Education* 24, 4 (2014), 253-276.
- [11] Allison Elliott Tew and Mark Guzdial. 2011. The FCS1: a language independent assessment of CS1 knowledge. In *Proceedings of the 42th ACM Technical Symposium on Computer Science Education*. ACM, 111–116.