Test-First Teaching: Extreme Programming Meets Instructional Design in Software Engineering Courses

Mark A. Ardis[1] and Cheryl A. Dugas[2]

Abstract – **Test-first development is a practice of extreme programming designed to produce reliable software quickly. Rather than writing the code first, a software engineer first creates the tests that will demonstrate that the software works correctly. Coding follows and is often guided by the tests. Practitioners of this method claim that the discipline of developing the tests before the code focuses their attention on the right problems and yields cleaner code. Test-First Teaching is a method of course development that incorporates Instructional Design methods to create more effective instruction. The instruments that will be used to test students' day-to-day learning of the course material – assignments and quizzes – are created first, and instruction is developed to meet the students' needs. Components of Test-First Teaching are applied at both course and lecture levels. Test-First Teaching has been used successfully to develop courses for the new Bachelor of Science in Software Engineering program at Rose-Hulman Institute of Technology.**

Index Terms – Extreme Programming, Instructional Design, Software Engineering, Test-First.

INTRODUCTION

In his book *Extreme Programming Explained*, Kent Beck [1] states "Software features that can't be demonstrated by automated tests simply don't exist." Test-first design is one of the 12 practices of extreme programming, and with good reason. What good is developing a software product if you can't demonstrate that it works properly? Test-first design forces the designer to think about what tests must be run to prove that the product does what it was intended to do. By writing tests first, the designer avoids creating a product that can't be proven to work properly. This paper describes a method of course and lecture development which uses the philosophy inherent in test-first design. In our case, the relevant question is "What good is developing instruction if you can't demonstrate that the students have learned what you've taught?" Inspiration for our method comes not only from extreme programming, but also from the field of instructional design.

Instructional design as a systems approach to developing instructional materials has gained in popularity over the years. There are a number of models in use, but each encompasses the same three major events: analysis, strategy, and evaluation. Even those who design these models and advocate their use recognize that it's not always possible to perform each of the steps in a thorough manner. But, being aware of the model and using the steps that you can may greatly increase the effectiveness of your instruction. At Rose-Hulman, the authors have successfully applied these principles in the development of software engineering courses for their new Bachelor of Science in Software Engineering (BSSE) program.

INSTRUCTIONAL DESIGN

We will give a brief introduction to Instructional Design, using the model shown in Figure 1, which is based on the popular Dick & Carey Model [2]. The steps of the process are outlined below:

- **Assess Needs to Identify Goals**
  Determine the need for the instruction.
  Determine what learners should be able to do when they have completed the instruction.
- **Conduct Instructional Analysis**
  Perform a hierarchical analysis of what the learner does when performing the goals.
  Determine prerequisite knowledge.
- **Analyze Learners and Contexts**
  Identify learner characteristics.
  Analyze the instructional setting.
- **Write Performance Objectives**
  Write specific statements of what the learners will be able to do upon completion of the instruction.
- **Develop Assessment Instruments**
  Write assessments that test each of the objectives.
- **Develop Instructional Strategy**
  Based on all of the above, identify an appropriate instructional strategy.
- **Develop and Select Instructional Materials**
  Using the strategy, produce the instruction.
- **Design and Conduct the Formative Evaluation of Instruction**
  Try out the instruction and look for flaws.

[1] Mark Ardis, Professor of Computer Science and Software Engineering, Rose-Hulman Institute of Technology, 5500 Wabash Avenue, Terre Haute, IN 47803 mark.ardis@rose-hulman.edu
[2] Cheryl Dugas, Graduate Student, Department of Curriculum, Instruction and Media Technology, School of Education, Indiana State University, Terre Haute, IN 47809 cdugas@indstate..edu

0-7803-8552-7/04/$20.00 © 2004 IEEE **October 20 – 23, 2004, Savannah, GA**
34th ASEE/IEEE Frontiers in Education Conference

- **Revise Instruction**
  Using results from the formative evaluation, make improvements.
  Repeat the design cycle for the modified instruction.

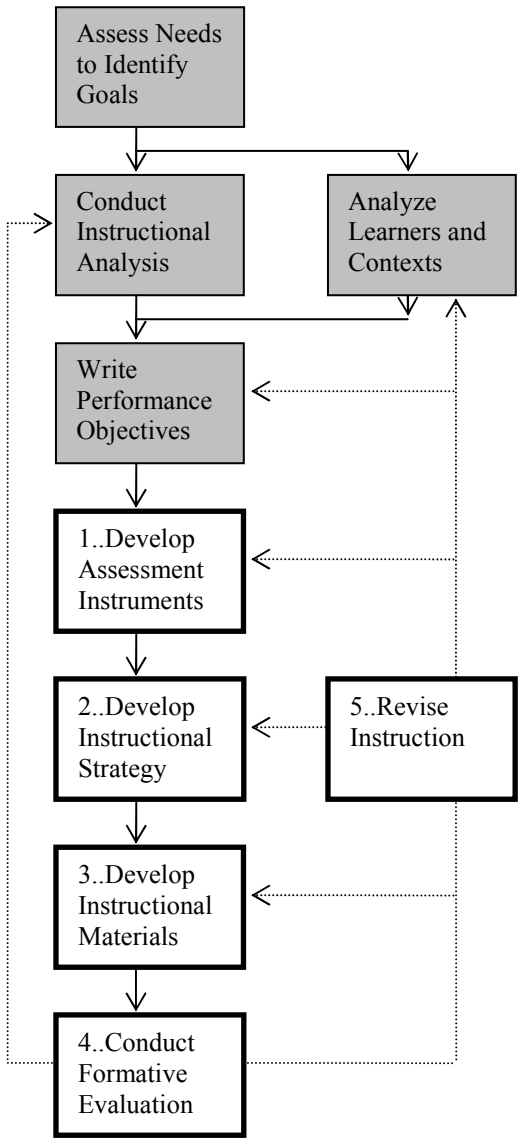

FIGURE 1
INSTRUCTIONAL DESIGN MODEL.

Instructional Design at Rose-Hulman

How does course development at Rose-Hulman relate to the Instructional Design Model? When the BSSE program was first created, the faculty met to decide what courses should be offered, and what should be taught in each of them. They relied upon their collective experience in developing other software engineering programs, and also used the Software Engineering Education Knowledge (SEEK) taxonomy from the software engineering volume of Computing Curriculum 2001[4]. The result was a list of course objectives and topics for each course. In the Instructional Design model, this process is covered in the top three shaded boxes on the left-hand side: Assess Needs to Identify Goals, Conduct Instructional Analysis, and Write Performance Objectives. The other shaded box in the top section – Analyze Learners and Contexts – really doesn't apply here since the learners and setting are always the same within Rose-Hulman, and are well-known by those developing the instruction.

Our attention will be focused on the highlighted boxes – 1 Develop Assessment Instruments, 2 Develop Instructional Strategy, 3 Develop Instructional Materials, 4 Conduct Formative Evaluation, and 5 Revise Instruction. The order of these activities is significant. It is what caused us to recognize the parallels between the type of instructional design that we were using in developing courses at Rose-Hulman and the test-first design practice of extreme programming.

Dick and Carey [2] emphasize the importance of designing assessment instruments before designing instruction:

> You may wonder why test development appears at this point in the instructional design process rather than after instruction has been developed. The major reason is that the test items must correspond one-to-one with the performance objectives. The performance required in the objective must match the performance required in the test item or performance task. Likewise, the nature of the test items that will be given to learners serves as a key to the development of the instructional strategy.

It is efficient and natural to think about what assessment should be used to determine whether an objective is met, at the time the objective is written. It may also save time and trouble later. As we asked in our opening paragraph, what good is developing instruction if you can't demonstrate that the students have learned what you've taught?

How many times have you heard students say "That exam didn't test anything we learned!" or "The homework had nothing to do with what we did in class." An additional benefit of using test-first teaching is that the assessments measure what was actually taught. Some educators may claim that this is "teaching to the test", but we claim that in this type of design, that's exactly what you want to be doing.

The remainder of the paper describes how we have applied the principles of Instructional Design and Test-First Design to both course and lecture development.

THE METHOD: COURSE DEVELOPMENT

Start with Course Objectives and Topics

As was mentioned before, the course objectives and topics for each of the courses in the BSSE program were developed by the faculty, with guidance from SEEK.

1. Develop Assessment Instruments:
Create Homework and Project Assignments

The first step in the test-first process of course development is to create homework and project assignments. The overriding concern here is: What problems must the students work to practice and master the material contained in the course? In instructional design wording, this can be stated as: What assessment instruments must be used to determine whether the course objectives have been met? It is important to be sure that all course objectives are covered.

2. Develop Instructional Strategy:
Determine Order and Method of Instruction

The next step is to put the assignments into order. This is determined by what skills the students need to complete each of them, and which have to precede others. Determine what type of instruction is appropriate for the material covered in each assignment – lecture, demonstration, role-playing, hands-on lab work, etc – and how much time should be allocated to the instruction. Use this to create a course outline.

3. Develop Instructional Materials:
Create Syllabus

Now is the time to select a text that covers the material that will be taught in the course. Using the text, flesh out the course outline into a syllabus. This will contain the usual things – what will be covered each day, reading assignments, project and homework assignments.

4. Conduct Formative Evaluation:
Evaluate Course Results

At the end of the term a review of the course is held with all the instructors in the BSSE program. Student scores on homework assignments and exams, and course evaluations by students are the main source of data for this analysis. Assignments and classes that should be revised are identified. Those assignments and classes that were particularly successful are also noted.

5. Revise Instruction:
Revise the Course

In those areas where students did not meet the objectives, changes are made to the course. This may entail developing different homework assignments, presenting different material in class, selecting a different text, varying the style of instruction for that topic, or some combination of all of these. Care is made to preserve those assignments and classes that were particularly successful in the past.

THE METHOD: LECTURE DEVELOPMENT

Start with Lecture Objectives and Topics

The objectives for a lecture are the subset of the course objectives that cover the material to be presented that day in class. It may be necessary to break these down into sub-objectives to attain the level of detail necessary to organize the instruction.

1. Develop Assessment Instruments:
Create a Daily Quiz

This is the most important part of the test-first teaching method. Create a quiz that covers the material for that day's instruction, and which addresses the objectives for that class. In instructional design phraseology: create an assessment instrument which will test whether that day's class objectives have been met. The quiz is to be given to the students at the beginning of the class period. The students will complete the quiz during the class period, or at the end of the period. The quiz serves two additional important functions beyond its use as an assessment instrument. First, it provides an outline for the development of instructional materials – lecture, demonstration, etc – for that day. Second, it provides the students with an outline of the class objectives.

2. Develop Instructional Strategy:
Determine Method of Instruction

Some of this work may have been done in writing the syllabus. This step may be done concurrently with the previous step, because the type of quiz may be suggested by the type of instruction that is most effective for the day's material. For instance, if the class format is to be a demonstration rather than a lecture, an appropriate quiz might be a checklist based on the demonstration.

3. Develop Instructional Materials:
Plan Class, Create Handouts

As mentioned above, the quiz provides an outline for the development of instructional materials. Using the quiz as a guide, it is usually a straightforward process to create the lecture, demonstration, or other form of instruction, along with necessary handouts.

4. Conduct Formative Evaluation:
Evaluate Class Results

This is perhaps where test-first teaching realizes its greatest benefits, for both instructor and student. A daily quiz provides immediate feedback on the effectiveness of the day's instruction. It is immediately apparent whether the students "got it", or whether there are gaps in their understanding. The student benefits by having a measure of his/her own understanding, and an indicator whether deeper study is necessary to understand the material. It is a win-win situation.

5. Revise Instruction:
Adjust Future Classes

If many students fail to achieve the goals set for the class, the instructor can make adjustments to the remaining classes to fill in those gaps. If it is only one or two students who failed to meet the objectives, they can work with the instructor or an assistant to master the material.

A Note on Exams

Exams to be given in the course are the only assessment instruments that are not created before the instruction. This is so that they can be tailored to the needs of a particular class section of students. The method for creating the exams follows: Begin by checking whether there were any objectives that were not covered by the assignments. If there are, include a test problem that covers the objective, to ensure that all course objectives are assessed. Examine the quizzes and write new versions of the questions. Since the quizzes were designed to cover the objectives, an exam based on the quizzes will be another assessment of the course objectives. Exams also test long-term retention.

TEST-FIRST TEACHING IN PRACTICE

Here are some examples of test-first teaching as it has been employed in CSSE 376 Software Quality Assurance at Rose-Hulman in the winter of 2003-2004. This is a junior-level course required for students in the BSSE program.

Course Development

First we describe the process of creating the course using test-first teaching. As mentioned earlier, lists of objectives and topics had already been drafted for the course. Now it was time for the instructor to elaborate all of the course details, such as the syllabus. Our example will look at only one of four major units in the course.

1. Develop Assessment Instruments:
Create Homework and Project Assignments

This course has several objectives, including "Evaluate a user interface for suitability". In order to accomplish this, students need to learn about usability testing and then practice what they learned. The following sequence of assignments was defined that culminated in a usability test of a simple software system:

10: Create a simple software system to test
11: Prepare a usability test plan
12: Prepare usability test materials
13: Conduct a usability test on the software system

FIGURE 2
USABILITY TEST ASSIGNMENTS

Students were to work in teams of 4 or 5 to write the software, develop the test plan and test materials, and conduct the testing. Testing participants included faculty and staff at Rose-Hulman and students in other courses. The software system was a simple calendar application for recording appointments.

2. Develop Instructional Strategy:
Determine Order and Method of Instruction

A course outline was developed to prepare students for those assignments. It was determined that about one quarter of the course, or 11 out of 40 class periods, could be devoted to this unit. The unit would start with usability test demonstrations. There would be follow-up lectures and practice role-playing the usability test roles and process. The unit would culminate with a usability test designed, planned, and conducted by the teams of students.

3. Develop Instructional Materials:
Create Syllabus

A text [3] was selected and the syllabus was elaborated. The relevant section is shown in Figure 3. Note that assignment 13 is due on a different day for each team.

Date	HW	Read	Topic
1/16		1-3	Overview of Usability Testing
1/19		4	Usability Testing Roles
1/20	10	5	Usability Test Plans
1/22		6	Selecting Test Participants
1/23	11	7	Preparing Test Materials
1/26		8	Conducting the Test
1/27	12	9-10	Debriefing, Analysis
1/29	13		Conduct Usability Test (Team 1)
1/30	13		Conduct Usability Test (Team 2)
2/2	13		Conduct Usability Test (Team 3)
2/3	13		Conduct Usability Test (Team 4)

FIGURE 3
SYLLABUS – USABILITY TEST SECTION.

Since this is the first time that this course has been taught, we cannot yet report the results of the formative evaluation and subsequent revisions. However, we expect to use the same process that has been used in other courses at Rose-Hulman, as described earlier.

Lecture Development

Next we describe how we prepared a single class in the course.

1. Develop Assessment Instruments:
Create a Daily Quiz

The topic for 1/20 was "Usability Test Plans". The objectives for that day were to understand the purpose and contents of a usability test plan. At the end of the class students should be capable of completing homework assignment 11: Prepare a usability test plan.

The first step was to prepare a quiz, shown in Figure 4 below. The quiz asks the students to apply their knowledge of usability test planning to a situation involving the software they created for homework assignment 10.

Your team is preparing to do usability testing of 2 versions of Rosey Calendar: one with a command-line interface and one with a GUI. You need to decide whether to continue development of the GUI, or just support the command-line interface.

1. Write a problem statement for this testing.
2. List the major steps of the method to be used in testing.
3. Write a task list that includes inserting, deleting and printing commands. Write the task list in the form needed for planning and plan review.
4. List 2 evaluation measures that you would use.
5. List 2 things that would be reported at the conclusion of testing.

FIGURE 4
USABILITY TEST QUIZ

2. Develop Instructional Strategy: Determine Method of Instruction

The students had seen several demonstrations of a usability test the previous day, staged by the instructor with a volunteer participant. In this class the students learned how to prepare for a usability test and to document their plans. It was decided that a lecture format would be appropriate.

3. Develop Instructional Materials: Plan Class, Create Handouts

The next step was to develop an outline and lecture notes for the class, as shown in Figure 5.

1. Purpose of Test Plans
2. Suggested Formats
3. Problem Statement
4. User Profiles
5. Example User Profile
6. Testing Methods
7. Task Lists
8. Environment/Equipment
9. Test Monitor Role
10. Evaluation Measures
11. Report Contents

FIGURE 5
USABILITY TEST LECTURE OUTLINE

Note that all of the tasks required on the quiz are covered, and that some additional topics are also included. The quiz does not attempt to cover *all* knowledge and skills required to complete the next homework assignment. Just as any test of software is merely a sample of all possible behaviors, quizzes in class sample the learning of students. Hopefully, the most important samples are chosen in each case.

At the beginning of class the quiz was distributed to all students and read aloud by the instructor. During the class students were given time to complete different sections of the quiz as those topics were covered. For example, the instructor paused to let students answer question 3: "Write a task list..." after finishing topic 7: "Task Lists". This method works because the questions on the quiz and the topics in the lecture are in the same order.

4. Conduct Formative Evaluation: Evaluate Class Results

While grading the quizzes, the instructor examined the pattern and types of errors made, in order to make any adjustments to the upcoming lectures. As was mentioned before, this is a great advantage to the daily quiz format – any material not sufficiently grasped by the students can be presented again at a later class. Weaknesses can be addressed. In addition, changes can be planned for the next time the course is taught, so each offering of the course can be better than the last. For this particular day it was observed that students only had a superficial understanding of evaluation measures.

5. Revise Instruction:
Adjust Future Classes

In order to instill a deeper understanding of evaluation measures, more time was allotted to that topic two days later. During that class students learned how to prepare test materials, including log sheets with evaluation scores.

STUDENT RESPONSE

Students like daily quizzes, many of them indicating that on course evaluations:

- "I like the daily quizzes, makes it easy to follow the lecture, makes it easy to pick up key ideas."
- "The daily quizzes are a great way to do things. With the quizzes I know what to pay close attention to, and what I should take away from the lecture."
- "The daily quizzes force you to think about the material."
- "I really like the style in which this class was taught. I learned something new with every lecture and every homework. That is very rare for me."

This last comment highlights another benefit of the test-first approach: assignments and coursework are tuned for maximum efficiency. When testing software one attempts to find the smallest number of tests that cover the most (and most important) behaviors. So, too, in courses one should strive to find the smallest number of assignments that cover the most (and most important) learning outcomes.

CONCLUSION

Beck [1] argues that programmers and their customers need to run lots of tests in order to get frequent feedback about progress. When the feedback is negative, corrections can be made immediately to keep the project on track. The goal is to reach project completion on time with few surprises.

In the same way, students and instructors need frequent feedback about the learning experience in courses. When the feedback is negative, either the student or the instructor (or both) need to take immediate action to keep the course on track. Test-first teaching is an effective method to use at 2 levels: creation and evaluation of courses, and creation and evaluation of individual classes.

This paper investigated parallels between test-first design and instructional design. We hope to find similar parallels between other extreme programming practices and instructional methods.

REFERENCES

[1] Beck, Kent, *Extreme Programming Explained*, Addison-Wesley, 2000.

[2] Dick, Walter, Carey, Lou, and Carey, James O., *The Systematic Design of Instruction (5th Edition)*, Pearson, Allyn & Bacon, 2000.

[3] Rubin, Jeffrey, *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*, John Wiley and Sons, 1994.

[4] *Second Draft of the Software Engineering Education Knowledge*, December 6, 2002. <sites.computer.org/ccse/know/SecondDraft.pdf>.