CSC 326: Software Engineering
Lab 1 Lesson Plan

Barry Peddycord III

10 January 2013

Overview and Purpose

Lesson Topic

Welcome to Class!

Learning Outcomes

By the end of this lesson, students should be able to...

- Understand the purpose of CSC 326 (not about programming, about people and process)
- Recognize the key concepts covered in lab - requirements, collaboration, communication, and compromise
- Demonstrate their problem solving ability by installing iTrust on their systems
- Understand their expectations and responsibilities in CSC 326 (the TA is not going to hold their hands - they are expected to show initiative by completing pre-labs and solving problems on their own)

Schedule

- Introductions (15 minutes)
- The Drawing Lab Activity (60 minutes)
- Install iTrust (45 minutes)
- Discuss HW1 (10 minutes)

Strategies and Procedures

Focusing activity to begin lesson:

We'll begin with introductions. I'll have each student begin by telling us their name and what they wish to do with their degree in Computer Science when they graduate. I want to make this class a forward-thinking group of students who take what they learn personally with their graduation goals in mind.

Instructional Strategies: What are you/your students going to do for each learning outcome?

The activities are predefined.

The introduction activity, because it focuses on their graduation goals, will help set the stage for students to take the course seriously. This course is not about programming, and they must see this immediately. I can talk about my research if the point isn't made clear or if they don't take the questions seriously.

The drawing activity will convey that there are two ways to look at requirements: as a minimum goal to achieve or as a challenge to be beaten. Some students will see the requirements as constraints, exhibiting creative approaches to the problem. The themes that will be covered are "requirements", "collaboration", "compromise", and "communication".

I expect the students to be capable of following instructions and seeking help when they need it. They will first be asked to install iTrust on the lab machines together, and then - if they have time - be free to install it on their own machines after they see how it works. The students are free to use their laptops if they like, but getting the software installed is their responsibility. I'll encourage students to help one another install iTrust if they get it done themselves.

Evaluation and Feedback

How do you know if they "got it"? What classroom assessment techniques could you use?

Each of the activities are scaffolded. In the drawing exercise, I'll ask them all why they think we do each activity and what each activity means with guiding questions (provided by the activity handout). Their answers should get more accurate with each iteration, converging on notions that requirements are loose, and they will be required to exercise judgment as they complete the exercises.

The installation exercise is also scaffolded. By requiring students to install iTrust on the lab machines before installing on their laptops, they will work in a simple environment before moving to a more complicated one. I don't expect the students to get done in class, as this class involves a significant amount of work outside of the classroom.

Follow-Up and Conclusion

How are you going to end class? Think about the "So What, Now What?" question.

HW1 is going to be tough. You have a week to learn how to use 6 new technologies. I'm leaving it up to you. As computer science students, I think you're capable of handling that.

Reflection

Before Class/Lab

What materials/resources do I need?

What technology/tools will I need to gather?

iTrust v15 should be on SourceForge.

Do I need to test an experiment or practice any activities ahead of time?

I need to make sure that the right versions are still on the lab machines. They shouldn't have changed since Fall, but I need to make sure that iTrust can still be installed.

What other special considerations should I prepare for?

After Class/Lab

How did I deviate from the lesson plan?

I was not as hard on the students as I planned to be. I must not have it in me to be the drill-sergeant type of instructor. For the most part, everything went as planned, but that's because it was heavily scripted.

What worked well in this lesson?

I felt like we stimulated some good discussion by talking about the reason why students are in CSC.

Requiring the students to do the work on the lab machines before their laptops was an excellent way to scaffold the assignment. They were able to solve the installation of iTrust mostly independently, so this should help prop them up for success when it comes to installing it on their laptops and doing HW2.1.

What didn't work well?

There was a bug in the presentation animation where the "secret" to the collaboration exercises was displayed with the instructions, kind of ruining how the exercises were

supposed to work. I tried to connect the concepts of requirements and teamwork in spite of that, but it was difficult.

What should I continue to do when I teach?

I liked the food for thought approach for starting off class. I think what I'm going to do is before each lecture, post a brief snippet of a faculty member's research and encourage the students to think about how they would respond to the challenges that they face.

What might I do differently next time?

Next lab, I'm going to focus substantially on proper test case and data development. Students, far too often, don't seem to grasp the importance of testing, so I may give them a pop quiz if Dr. Xie doesn't beat me to it in the lecture segment. Last semester, I feel like I didn't cover regression tests early enough to make a difference... we'll see if we can fix that this semester.

Notes

I would like to thank Dr. Barbi Honeycutt for the great lesson plan template! For anyone who wants the LaTeXSource, please contact me at *bwpeddyc [at] ncsu [dot] edu*.

CSC 326: Software Engineering
Lab 2 Lesson Plan

Barry Peddycord III

17 January 2013

Overview and Purpose

Lesson Topic

Bug Hunt!

Learning Outcomes

By the end of this lesson, students should be able to...

- Express importance of regression testing
- Write an acceptance test plan
- Differentiate between good and bad tests
- Identify bugs in software

Schedule

- Meet your Partner (5 minutes)
- Testing Discussion (10 minutes)
- Bug Hunt (55 minutes)
- Trading Test Plans (30 minutes)
- Homework 2 pt 2 (10 minutes)

Strategies and Procedures

Focusing activity to begin lesson:

We will begin by looking at a test plan with obvious errors in it. This will help the students realize what NOT to do and identify major problems with the text.

Instructional Strategies: What are you/your students going to do for each learning outcome?

Student will begin by looking at a ”bad” test plan and then have to write their own for Use Cases 1, 18, and 46. Students will then share test plans with one another and assess them based on their repeatability and specificity.

Evaluation and Feedback

How do you know if they “got it”? What classroom assessment techniques could you use?

The ”spot the error” exercise will be a think-pair-share. I want to make sure that students know what a good and bad test plan should look like before making their own, and then students can catch one another when they evaluate each other.

Follow-Up and Conclusion

How are you going to end class? Think about the “So What, Now What?” question.

Why are test plans important? Because if they aren’t specific and rigorous, how do you know your code is good? If your bug reports are not specific, how do you expect the TAs to be able to help you?

Reflection

Before Class/Lab

What materials/resources do I need?

What technology/tools will I need to gather?

I need to make sure all the teams are correctly set up in Paireval.

Do I need to test an experiment or practice any activities ahead of time?

I’ve already found the bugs in UC 18 and 46. I’m hoping that a bug in UC 1 will manifest itself.

What other special considerations should I prepare for?

After Class/Lab

How did I deviate from the lesson plan?

I stuck with the lesson plan to the letter.

What worked well in this lesson?

Students found all of the problems in the example - the think-pair-share worked well to get them to investigate the problems with the test plan, including its lack of specificity, missing columns, and so forth. I haven't looked through the submitted black box test plans yet, but I think I can wait until I see HW2.2 to see how well it all sunk in.

I'm glad that all of the students were able to use their own version of iTrust - last semester, students were still having trouble starting iTrust and had to rely on the demo version. Nobody was using the demo version this time, meaning that my approach for getting them to use the lab machines before setting things up on their laptops provided some great scaffolding for getting up to speed with the software.

What didn't work well?

Students had some problems finding bugs in the code base. Most teams found at least one bug, but only a few managed to find one for each use case.

The class still ended up finishing about 20 minutes early. I need to learn to let students work without getting so antsy about moving along.

What should I continue to do when I teach?

Think-pair-share works very well with very little overhead. I'll definitely make use of it as I move forward.

What might I do differently next time?

While the focus for this exercise was on testing, I wish I had taken the time to go over systematic bug finding. We'll get a chance to do that in the security lecture, though.

Notes

I would like to thank Dr. Barbi Honeycutt for the great lesson plan template! For anyone who wants the LaTeXSource, please contact me at *bwpeddyc [at] ncsu [dot] edu*.

CSC 326: Software Engineering
Lab 3 Lesson Plan

Barry Peddycord III

24 January 2013

Overview and Purpose

Lesson Topic

Homework 2: Bugzilla, Black Box Tests

Learning Outcomes

By the end of this lesson, students should be able to...

- Write a bug report in bugzilla
- Write a black box test plan
- Create test data for their test plan without breaking prior tests
- Estimate the time it will take to complete bugs
- Complete Homework 2

Schedule

- Test Plan Peer Review (15 minutes)
- Bugzilla (30 minutes)
- Discuss the more well-defined requirements for the bugs (30 minutes)
- Planning poker (30 minutes)
- Work on HW2.2 (rest of class)

Strategies and Procedures

Focusing activity to begin lesson:

I'll begin by having the teams peer review each others' test plans. Look for the problems in the test plan we looked at yesterday. And come up with the grade that they would give them.

Instructional Strategies: What are you/your students going to do for each learning outcome?

I want the teams to look at each others' test plans to ensure that they know what good tests look like.

We'll then put it to the test by creating the bug reports in bugzilla.

Afterwards, we will approach the estimation component by playing planning poker. Students need to recognize the importance of estimation since they will be the "code experts" in their organization when they graduate - they will be expected to provide strong estimates.

Evaluation and Feedback

How do you know if they "got it"? What classroom assessment techniques could you use?

Looking at the exercises today will be the way I can gauge what I expect from students in HW2.2. HW2's performance will reveal a lot about how students treat the testing problem.

Follow-Up and Conclusion

How are you going to end class? Think about the "So What, Now What?" question.

Since we're ending with planning poker, this is my chance to bring out the higher-level questions. We are not raising programmers, but project managers! Their estimates should get more accurate as time moves on.

Reflection

Before Class/Lab

What materials/resources do I need?

What technology/tools will I need to gather?

Make sure Bugzilla works. I'll have the students send a password reset request to get their passwords set as they want them.

Do I need to test an experiment or practice any activities ahead of time?

What other special considerations should I prepare for?

After Class/Lab

How did I deviate from the lesson plan?

The students did not complete their black box test plans and bring them to class, so I wasn't able to do the BBTP peer review. As such, I deviated by having the students dive right into Bugzilla and then peer review each others bug reports. Despite the hiccup, things proceeded smoothly after that.

I also spent the last portion of class - rather than letting students work on the HW, setting up their SVN repositories, which causes major issues each semester.

What worked well in this lesson?

Having the students set up their SVN repositories before leaving as a "ticket out the door" works very well, ensuring that major roadblocks to collaboration are taken care of before they have to work for a week on their own.

What didn't work well?

As I can tell from last lesson, I wasn't very clear with the expected deliverables.

What should I continue to do when I teach?

The best focus for lab time is on clearing up misconceptions and dealing with technology issues (SVN, iTrust, etc). By focusing on those, I make the best use of the time.

The second best usage is to discuss lab technologies and principles.

What might I do differently next time?

Notes

I would like to thank Dr. Barbi Honeycutt for the great lesson plan template! For anyone who wants the LaTeXSource, please contact me at *bwpeddyc [at] ncsu [dot] edu*.

CSC 326: Software Engineering
Lab 4 Lesson Plan

Barry Peddycord III

31 January 2013

Overview and Purpose

Lesson Topic

Homework 3: Requirements Inspection

Learning Outcomes

By the end of this lesson, students should be able to...

- Conduct a requirements inspection
- Create CRC Cards

Schedule

- HW2 Discussion (10 minutes)
- HW3 Requirements Inspection (30 minutes)
- CRC Cards (30 minutes)
- Planning Poker (30 minutes)
- Set up SVN for HW3 (rest of class)

Strategies and Procedures

Focusing activity to begin lesson:

Write down one thing that *you* could have done to make HW2 part 2 easier on yourselves. After that, write one thing that we as the TAs could have done.

Instructional Strategies: What are you/your students going to do for each learning outcome?

We are going to conduct a requirements inspection. In the new teams, students will read UC46 and critically identify what questions that they have to ask the TAs to make sure that they code the right things.

Afterwards, we'll have a CRC card session to ground the ideas of modularity and a planning poker session to refine estimation skills.

Evaluation and Feedback

How do you know if they "got it"? What classroom assessment techniques could you use?

The class will share their questions, showing that they thought about the gaps in the requirements.

Follow-Up and Conclusion

How are you going to end class? Think about the "So What, Now What?" question.

I want to end class making sure students have the technology to complete the homework, so the ticket out the door will be having SVN working.

Reflection

Before Class/Lab

What materials/resources do I need?

What technology/tools will I need to gather?

Do I need to test an experiment or practice any activities ahead of time?

What other special considerations should I prepare for?

After Class/Lab

How did I deviate from the lesson plan?

What worked well in this lesson?

What didn't work well?

I ran out of time in the first lab, taking far too long with the requirements inspection and not having enough time for planning poker.

What should I continue to do when I teach?

What might I do differently next time?

Notes

I would like to thank Dr. Barbi Honeycutt for the great lesson plan template! For anyone who wants the LaTeXSource, please contact me at *bwpeddyc [at] ncsu [dot] edu*.

CSC 326: Software Engineering
Lab 6 Lesson Plan

Barry Peddycord III

14 February 2013

Overview and Purpose

Lesson Topic

Homework 4: Requirements Inspection

Learning Outcomes

By the end of this lesson, students should be able to...

- Conduct a requirements inspection
- Complete HW4 part 1

Schedule

- HW3 Discussion (10 minutes)
- Requirements Inspection (40 minutes)
- Scenarios, Test Data, Planning (rest of class)

Strategies and Procedures

Focusing activity to begin lesson:

Get together to comment on HW3 and reflect on what has been learned thus far.

Instructional Strategies: What are you/your students going to do for each learning outcome?

We are going to conduct another requirements inspection.

Evaluation and Feedback

How do you know if they "got it"? What classroom assessment techniques could you use?

Students will share their concerns again.

Follow-Up and Conclusion

How are you going to end class? Think about the "So What, Now What?" question.

Once again, make sure students can commit to SVN before leaving.

Reflection

Before Class/Lab

What materials/resources do I need?

What technology/tools will I need to gather?

Do I need to test an experiment or practice any activities ahead of time?

What other special considerations should I prepare for?

After Class/Lab

How did I deviate from the lesson plan?

What worked well in this lesson?

What didn't work well?

I ran out of time in the first lab, taking far too long with the requirements inspection and not having enough time for planning poker.

What should I continue to do when I teach?

What might I do differently next time?

Notes

I would like to thank Dr. Barbi Honeycutt for the great lesson plan template! For anyone who wants the LaTeXSource, please contact me at *bwpeddyc [at] ncsu [dot] edu*.

CSC 326: Software Engineering
Lab 7 Lesson Plan

Barry Peddycord III

21 February 2013

Overview and Purpose

Lesson Topic

Homework 4: Peer Review and Intro to SQL

Learning Outcomes

By the end of this lesson, students should be able to...

- Use JOIN statements
- begin coding HW4
- Use static analysis tools to find bugs

Schedule

- SQL Mini-lecture (20 minutes)
- HW4 Peer Review (30 minutes)
- HW4 Test Evaluation (30 minutes)
- Static Analysis (30 minutes)

Strategies and Procedures

Focusing activity to begin lesson:

Discuss SQL.

Instructional Strategies: What are you/your students going to do for each learning outcome?

I'll give a mini-lecture and explain the theory of Relational Databases. This is verbal information, so lecture is reasonable.

Students will dive in and use Findbugs to learn about static analysis tools.

Evaluation and Feedback

How do you know if they "got it"? What classroom assessment techniques could you use?

Think-pair-share on the various ways to handle relations in SQL, one-to-one and many-to-one.

Follow-Up and Conclusion

How are you going to end class? Think about the "So What, Now What?" question.

Going to let class end early so that students can work on HW4 and/or study for the midterm next week.

Reflection

Before Class/Lab

What materials/resources do I need?

What technology/tools will I need to gather?

Do I need to test an experiment or practice any activities ahead of time?

What other special considerations should I prepare for?

After Class/Lab

How did I deviate from the lesson plan?

An hour for the peer review was too long. We ended up not taking nearly as much time as I thought we would.

What worked well in this lesson?

The lecture went smoothly.

What didn't work well?

I should have given students an opportunity to demonstrate understanding of the join statement. I was afraid that there was too much in the lab session, but an hour for the peer evaluation was a great overestimate of how much time they needed.

Also, I think I might have given away too much information, but I'm more concerned with mastery of the SQL than discovery - at least this time around.

What should I continue to do when I teach?

Mini-lectures on technical subjects.

What might I do differently next time?

I should try to make worksheets. That will give students something to work on while I teach.

Notes

I would like to thank Dr. Barbi Honeycutt for the great lesson plan template! For anyone who wants the LaTeXSource, please contact me at *bwpeddyc [at] ncsu [dot] edu*.