New Course Development: 461L Software Engineering and Design Laboratory

Prof. Miryung Kim
Electrical and Computer Engineering
The University of Texas at Austin
A new junior level, software engineering & design laboratory class was created to meet the needs of our ECE undergraduates.

Class activities and self-paced tool tutorials helped students to engage in highly abstract subject matter and gain confidence in working with large software.
Outline

- Motivation
- Key Objectives
- Course Structure
- Example Instruction Materials and Methods
- Lessons Learned
- Conclusions
Among ECE undergraduates, Software Engineering and Design Core (SE) has become the most popular technical area.
Motivation 2.
Lack of Core SE Laboratory Class

- SE tech area did not have its own core laboratory class in the old 2008 catalog.
- A lack of emphasis on hands-on experience in 422C, 360F, and 360C

<table>
<thead>
<tr>
<th></th>
<th>ComNets</th>
<th>445S Digital Signal Processing Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ICS</td>
<td>438 Electronic Circuits</td>
</tr>
<tr>
<td>EE</td>
<td>Energy</td>
<td>462L Power Electronics</td>
</tr>
<tr>
<td></td>
<td>EM</td>
<td>462L or 438</td>
</tr>
<tr>
<td></td>
<td>Nanotechnology</td>
<td>440L Micro Elec Fabrication</td>
</tr>
<tr>
<td>CE</td>
<td>CAEP</td>
<td>445L Microprocessor Lab</td>
</tr>
<tr>
<td></td>
<td>Soft. Engineering</td>
<td>No lab course. Take 445L Instead</td>
</tr>
</tbody>
</table>
**Motivation 3. Prepare our students for professional careers in SE**

### Example Career Paths

<table>
<thead>
<tr>
<th>Process Control Engineer</th>
<th>Software Engineer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tech Program Manager =&gt; Senior PM</td>
<td>Manager =&gt; Senior Manager =&gt; Director</td>
</tr>
<tr>
<td>Senior Product Manager =&gt; Senior Manager</td>
<td>CTO of his own start-up</td>
</tr>
<tr>
<td>Founder of his own start-up =&gt; Director</td>
<td>Senior Vice President &amp; Chief Digital Officer</td>
</tr>
</tbody>
</table>

- BS in Applied Physics and Electrical Eng.
- BS in Systems Engineering

---

**Art** produced by virtuosos with years of experience?

**Business management** (organization & planning)?

**VS.**

**Science & Engineering**
Outline

- Motivation
- **Key Objectives**
- Course Structure
- Example Instruction Materials and Methods
- Lessons Learned
- Conclusions
Key Objectives

- **Hands on experience**
- Teach tools required by industry
- Systematic engineering methods
- Realistic project tasks
Key Objectives

- Hands on experience
- **Teach tools required by industry**
- Systematic engineering methods
- Realistic project tasks

Version Control Tools, Unix Utilities & Shell Scripting, UML Modeling Tool, Build Mgmt Tool, Unit Testing Tool, Debugger, Profiler, etc.
Key Objectives

- Hands on experience
- Teach tools required by industry
- **Systematic engineering methods**
- Realistic project tasks

Design Patterns, Unit Testing, Regression Testing, Formal Methods, Static and Dynamic Program Analysis
Key Objectives

- Hands on experience
- Teach tools
- Systematic engineering methods
- **Realistic project tasks**

Building a **small** project from scratch **X**

VS.

Evolving a **large system** through **feature additions**
Course Structure

Lectures
(3 hours per week)
Concepts, principles, & methods
Engaging students through class activities, discussions, and demos

Tool Tutorials and Exercises
(lab: 3 hours)
Tool installation & following self-paced tutorials
exercise tasks at each milestone

Team Projects +
New in Fall 2012
Homework +
Paper Reviews +
Quizzes + Exams
<table>
<thead>
<tr>
<th>Week 1 (8/30)</th>
<th>Lectures (T/TH)</th>
<th>Laboratory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lecture 1: Overview</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tutorial 1a: Subversion Version Control System Tutorial 1b: Project - Saros (Distributed Pair Programming)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tutorial 2: UML</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tutorial 3: Unix Environment and Command-line Utilities and Shell Scripting</td>
<td></td>
</tr>
<tr>
<td>Week 5 (9/25, 9/27)</td>
<td>Class Presentations.</td>
<td>Quiz 2. Unix Commands and UML (Thursday)</td>
</tr>
<tr>
<td>Week 6 (10/2, 10/4)</td>
<td>Lecture 5: Information Hiding Principle Lecture 6: Design Patterns Part 1: Abstract Factory, Factory Method</td>
<td>Project Part A. New Feature Proposal. (Due: Tuesday, 12:29PM)</td>
</tr>
<tr>
<td></td>
<td>• Motivation and User Benefits, Feature Description and Requirements, Identification of Relevant Classes, Mock-Up Screenshots. • Use Case Diagram in UML • Preliminary Class Diagram in UML</td>
<td></td>
</tr>
<tr>
<td>Week 7 (10/9, 10/11)</td>
<td>Lecture 6: Design Patterns Part 2. Singleton, Adapter, Flyweight, Bridge Lecture 6: Design Patterns Part 3. Observer, Mediator, Strategy, Visitor</td>
<td>Reading Assignment 1: Paper Review Report Due in class on Tuesday 12:29PM. On the criteria to be used in decomposing systems into modules, DL Parnas</td>
</tr>
<tr>
<td></td>
<td>Tutorial 5: Improving Design Design Pattern and Refactoring</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Quiz 3. Information Hiding Principle and Design Patterns (Thursday)</td>
<td></td>
</tr>
</tbody>
</table>
Outline

- Motivation
- Key Objectives
- Course Structure
- **Example Instruction Materials and Methods**
- Lessons Learned
- Conclusions
Extension Scenario 1. How about adding a different look and feel such as `MacWindowKit`?

VS.

Extension Scenario 2. How about adding a new type of an object, `Button`?
1. Write tests for this program.
2. How do you know the adequacy of your tests?
Control Flow Graph

Fill out the following code coverage table by running the program with the following inputs

<table>
<thead>
<tr>
<th>input</th>
<th>exercised statements</th>
<th>exercised branches</th>
<th>exercised paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>(cond1=true, cond2=true, cond3=true)</td>
<td>s1, s2, s3, s4, s5, s6, s7</td>
<td>b1, b3, b5</td>
<td>[b1, b3, b5]</td>
</tr>
<tr>
<td>Coverage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(cond1=false, cond2=false, cond3=false)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coverage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(cond1=false, cond2=true, cond3=true)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coverage</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Is this program correct?  
2. Which pre-condition does this program need to satisfy?
Logical reasoning of weakest pre-conditions and loop invariant
Self-paced Tutorial Example 1: UML Design and Modeling Tool
Self-paced Tutorial Example 2
Test Coverage with JUnit

```java
@Test
public void testAllTrue() {
    assertEquals(0, CoverageExample.identity(0, true, true, true));
}

// Branch Coverage, all of the above plus the following

@Test
public void testAllFalse() {
    assertEquals(0, CoverageExample.identity(0, false, false, false));
}

// Path Coverage, all of the above plus the following

@Test
public void testTrueFalseFalse() {
    assertEquals(0, CoverageExample.identity(0, true, false, false));
}

@Test
public void testTrueFalseTrue() {
    assertEquals(0, CoverageExample.identity(0, true, true, true));
}
```
Team Project

- 4 to 5 person team
- Adding a feature to an open source project
- SAROS is an Eclipse plug-in for distributed collaborative programming, developed by Lutz Prechelt at Freie Univ. Berlin in Germany
Team Project

Part A: New Feature Proposal
- Motivation
- User Benefits
- Feature Descriptions
- Mock-up Screenshot
- Preliminary Design in UML

Part B: Implementation Progress
- Design Extension in UML
- API Descriptions
- User Interfaces
- Test Scenarios and Test Cases in JUnit

Part C Final Feature Demonstration
- Design Extension in UML
- API Descriptions
- User Interfaces
- Test Scenarios and Test Cases in JUnit
- User Manuals
Example: SAROS User Statistics with Logging & Tweeting Features
Questions about Hoare Triples

To answer John's question in class, we can check whether a HOARE TRIPLE \((P \ S \ Q)\) is true or false by computing a weakest precondition \((S, Q)\) and show that \(P\) implies \(wp(S, Q)\).

For example,
- \(\{true\}\): \(x' = x' \land y' = 0\)
  - The answer is \(true\) because \(wp(y = x' \land y = 0)\) is \(true\) and \(true\) implies \(true\)
- \(x = y'\) \(\land y = y'\)
  - The answer is \(true\) because \(wp((x = x \land y) \land y = y)\) is \((x = X \land y = Y)\) and \((x = X \land y = Y)\) implies \(x = x\) \(\land y = y\)
- \(x > 0\) \(\land x > 0\)
  - \(wp(x > 0, x > 0)\) is \((x > 0)\) and \((x > 0)\) implies \((x > 0)\) so it is true.
- \(x > 0\) \(\land x > 0\)
  - \(wp(x > 0, x > 0)\) is \((x > 0)\) and \((x > 0)\) implies \((x > 0)\) so it is true. The other way of thinking about this is to compute the strongest post condition, \((x > 0)\) and show that \((x > 0)\) implies \((x > 0)\)
- \(\{true\}\): \(x' = x' \land y' = 0\)
  - \(wp(y = x' \land y = 0)\) is \((x > 0)\) and \((x > 0)\) implies \((x > 0)\) so it is true.
- \(x = y'\) \(\land y = y'\)
  - \(wp(x = x \land y = y)\) is \((x = X \land y = Y)\) and \((x = X \land y = Y)\) does not imply \((y = Y \land y = Y)\), so the answer is false.

The following is from Question \(\#3\) in the class activity.

\(\{x' < 0\}\)
- If \((x > 0)\) then \(x := x + 1\) else \(x := -x\)
- \(wp(1)\)
  - \((x > 0)\) then \(x := x + 1\) else \(x := -x\)
  - \((x > 0)\) and \(wp(x := x + 1, x > 0)\) OR \((x < 0)\) and \(wp(x := -x, x > 0)\)
  - \((x > 0)\) and \(wp(x := x + 1, x > 0)\) OR \((x < 0)\) and \(wp(x := -x, x > 0)\)
  - \((x > 0)\) and \(wp(x := x + 1, x > 0)\) OR \((x < 0)\) and \(wp(x := -x, x > 0)\)
  - \((x > 0)\) or \((x < 0)\) OR \((x > 0)\) OR \((x < 0)\)
  - \(wp(x := x + 1, x > 0)\) so the above triple is true.
Welcome, miryungkim!

Spaces you collaborate with

- **Chime Report** (Free/Private | Member)
- **Chime-FSE-New-Idea** (Free/Private | Member)
- **ChimeExample** (Free/Private | Owner)
- **ChimeGraph** (Free/Private | Owner)
- **ChimeMediumSizeSubjectSystem** (Free/Private | Member)
- **LASE GUI** (Free/Private | Owner)

Stream

- **12:32** Mehmet E Yesin @ ChimeGraph accepted invitation mehmeterolyesin
  - **2012-09-28**
- **16:28** ruiqiu @ ChimeGraph invited mehmeterolyesin
  - **2012-09-19**
- **21:01** jwacobellis @ LASE GUI invited allisonsksullivan@utexas.edu
  - **2012-09-18**
- **11:38** jwacobellis @ LASE GUI committed [6]: version in development, adding save/load edit script feature
  - **2012-09-14**
- **11:09** jwacobellis @ LASE GUI committed [7]: minor UI changes
  - **2012-09-13**
- **21:02** jwacobellis @ LASE GUI invited allisonsksullivan@utexas.edu
  - **2012-09-13**
- **15:24** smcho @ ChimeGraph accepted invitation smcho

Create another Space
“I think (in class) activities are great, they help me a lot to understand the concepts that are taught in class.”
“Dr. Kim has designed a highly interactive course. The skills we learn during labs have helped me become stronger in my software skills.”
“It is useful to learn different tools. The self-paced tutorial is a nice way to do it.” “It is also good to feel like I am in an open forum and can ask for help at any time”

“It should be a sophomore level class because this is material you need before internships! I had a hard time learning this on the job.”
“I think this course can benefit from having weekly or biweekly homework.”
“Reading academic papers would be cool.”
Lessons Learned

- Provide early and frequent feedback
- Incentivize rather than offer unsolicited advice
- Clear communication on course management / expectations
- Bring out creativity and ownership for class projects
Conclusions

- 461L intends to provide hands-on experience in working with large software systems and to prepare for professional careers in SE.
- In class activities and self-paced tutorials helped students to engage in highly abstract subject matters.
- Early and frequent feedback through tests and assignments and clear communications on course expectation are needed.
Acknowledgment

- Software Engineering Area Faculty in ECE
- Academic Development Fund from Cockrell School of Engineering
- ECE Department
- TAs: Evan Grim, Rui Qui, and Kevin Boos