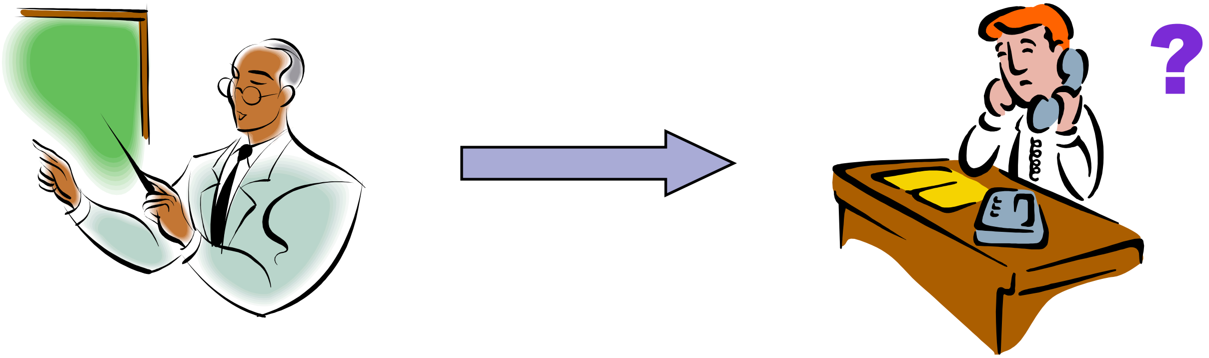
Production Programming in the Classroom

Eric Allen, Robert Cartwright, and Charles Reis

Rice University

{eallen, cork, creis}@rice.edu

Missing Part of CS Education



- Students are not taught to program in a production environment
  - Projects written from scratch, then discarded
  - No project maintenance
  - No real users to consider

Teach Production Programming

- Students should:
  - Maintain an existing codebase
  - Support *real* customers
  - Learn effective methodologies

But…

- Where does the project come from?
- Who are the customers?
- How can an instructor manage all this?

Selecting a Project

- On-campus customers
  - Ideally, the students themselves
  -
- *Open Source*
  - Free, high quality tools
  - Easy to gain customers
  -
- At least one year old, sustainable

Selecting a Methodology

- Extreme Programming! (XP)
-
- Effective methodology in industry
  - Leading edge practices
  - Rapid development, reliable products
  -
- Translates well to classroom

XP: Pair Programming

- Better Design, Fewer Bugs
  - Quick development pace
  -
- Knowledge Transfer
  - Students pair with experienced developers
  - Effective against very high turnover

XP: Unit Testing

- Unit tests for every non-trivial method!
  - Enforce tests are run before committing
  -
- Confidence to make changes
  - Won’t break old functionality
  -
- “Executable Documentation”
  - Quickly learn the code

XP: Continuous Refactoring

- Let students improve *any* part of the code
  - Collective ownership: sense of pride
  - Prevents fragile code
  -
- Unit tests provide safeguard
  - Always safe to refactor!

XP: Incremental Development

- Break down into small tasks
  - Estimate time-to-completion
  - Keeps codebase stable
  -
- Release frequently
  - Students get feedback from real users
  - Much more powerful than just grades

But… XP in the classroom?

- Difficult to apply XP in a course:
  - Scheduling pair programming time?
  - Maintaining unit test coverage?
  - Finding on-site customers?
  - Managing a work force?

Applying Pair Programming

- Schedule time to pair program in class
  - Two lectures, one closed lab per week
  -
- Allow students to choose pairs
  - Avoid scheduling conflicts

Ensure Unit Tests are Written

- *Hard* to get students in right mindset
  - Early assignments for writing tests
  - Emphasize the importance of tests!
  -
- Test-Driven Development
  - Write the test *first*, then the code

Providing On-site Customers

- Students themselves should be customers
  - Careful selection of project
  -
- Discuss new features, specs in class
-
- Also support off-campus customers

Course Management

- Experienced TA’s as Project Managers
  - Pair program with new students
  - Sustainable
  -
- Supervise progress on tasks
  - Monitor students like employees

SourceForge.net

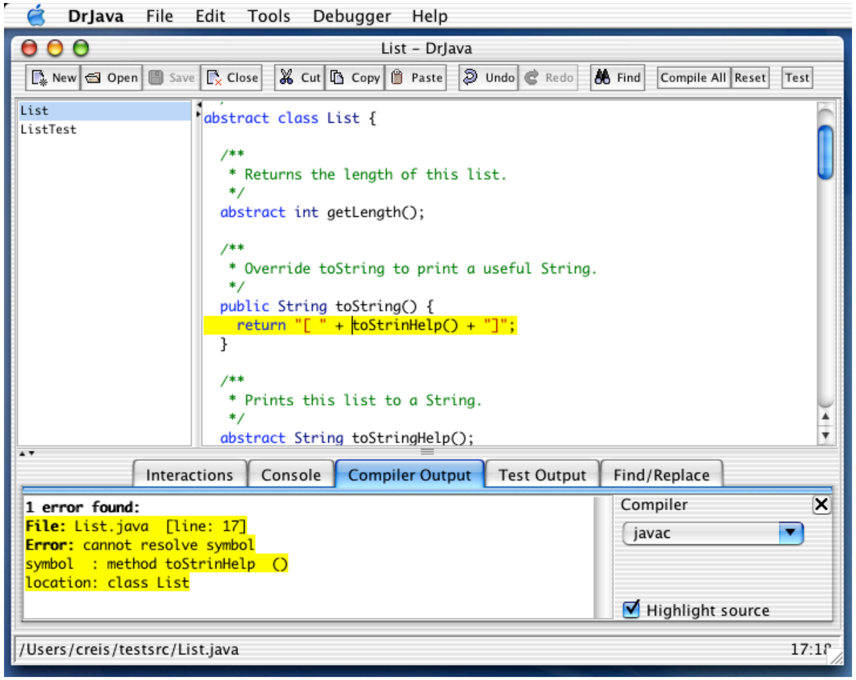
- Free open source project hosting
-
- Professional Management Tools
  - Track features/bugs online
  - Task management
  - Respond to customers

Open Source Tools

- JUnit
  - Framework for writing unit tests
  -
- Ant
  - Build tool with XML build scripts
  - Enforce all tests pass before committing code

Case Study: DrJava

- Pedagogic IDE developed at Rice University
  - Used in intro Java courses
  - Also useful for advanced developers



```
abstract class List {

  /**
   * Returns the length of this list.
   */
  abstract int getLength();

  /**
   * Override toString to print a useful String.
   */
  public String toString() {
    return "[ " + toStrinHelp() + "]";
  }

  /**
   * Prints this list to a String.
   */
  abstract String toStringHelp();
```

```
1 error found:
File: List.java  [line: 17]
Error: cannot resolve symbol
symbol  : method toStrinHelp  ()
location: class List
```

DrJava: Ideal Candidate

- Open Source
-
- Full unit test coverage
  - >35% of codebase is test code
  -
- Students can be customers!
- Other customers around the world
  - 14+ schools, 10+ countries

DrJava as Course Project

- Many small projects in pairs
  - Bug fixes, feature requests from customers
-
- Three major projects
  - JUnit Integration
  - Configurability
  - Integrated Debugger

Results, Spring 2002

- Mostly successful
  - High quality code
  -
- Large projects not completed
  - Two unfinished, one had a bug
  - Difficult to estimate
  - No fixed deadlines: course became low priority

Changes for Spring 2003

- Students must log 10 hours per week
-
- *Emphasize* test-first programming!

Conclusion

- You CAN Teach Production Programming
  - Extreme Programming works in classroom
  - Unit tests are a safeguard
  - Open Source tools