Lab 2: Intro Material

Software Engineering and Pair Programming

Notes for CSC 100 - The Beauty and Joy of Computing
The University of North Carolina at Greensboro

Question to think about….

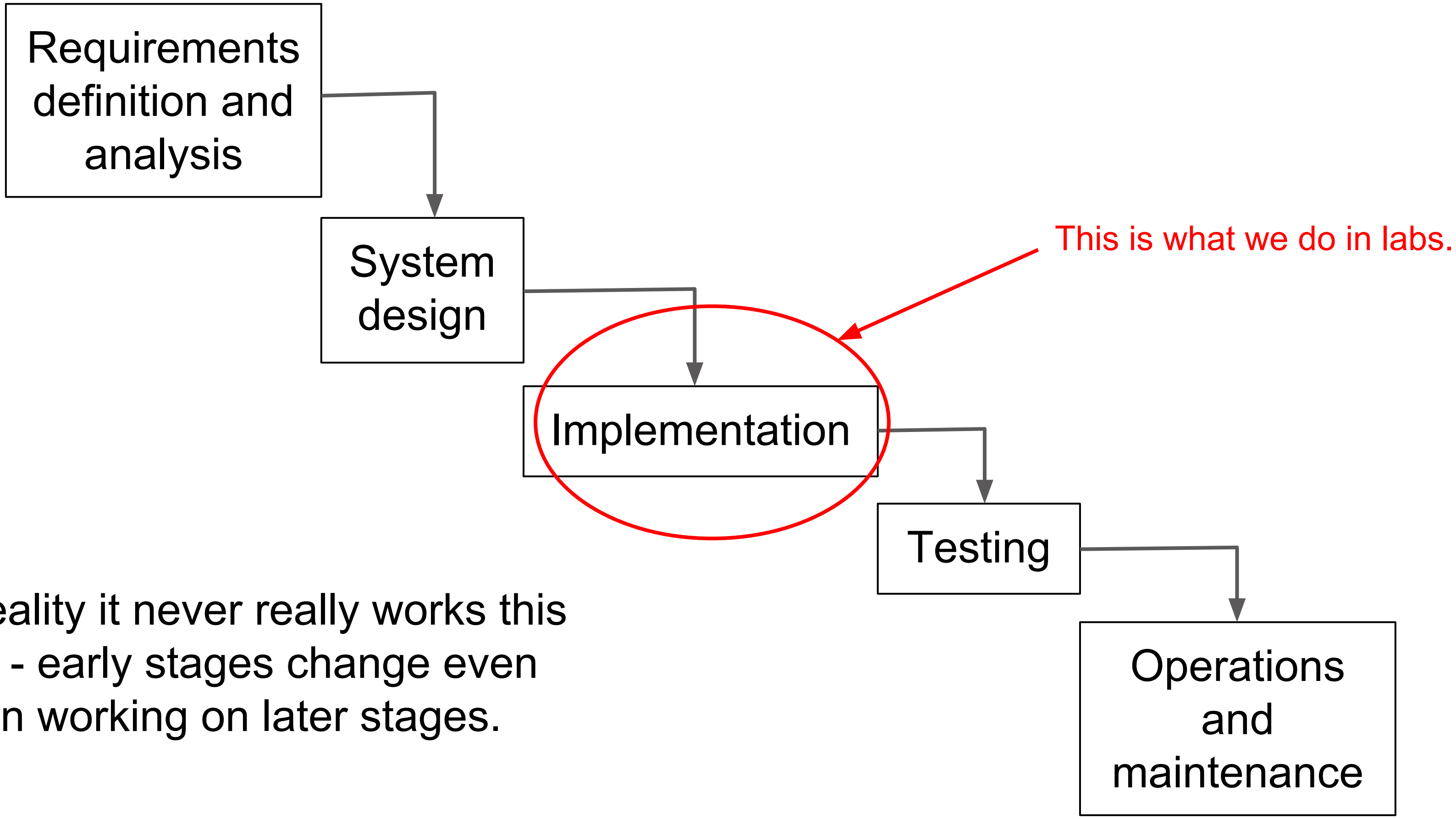
In these labs you are writing programs

How does this relate to professional software development?

Software Engineering

How to control the complex process of creating software

Traditional "software lifecycle" has well defined phases that feed into each other one-way - called the "Waterfall Model"


Requirements definition and analysis
System design
Implementation
Testing
Operations and maintenance

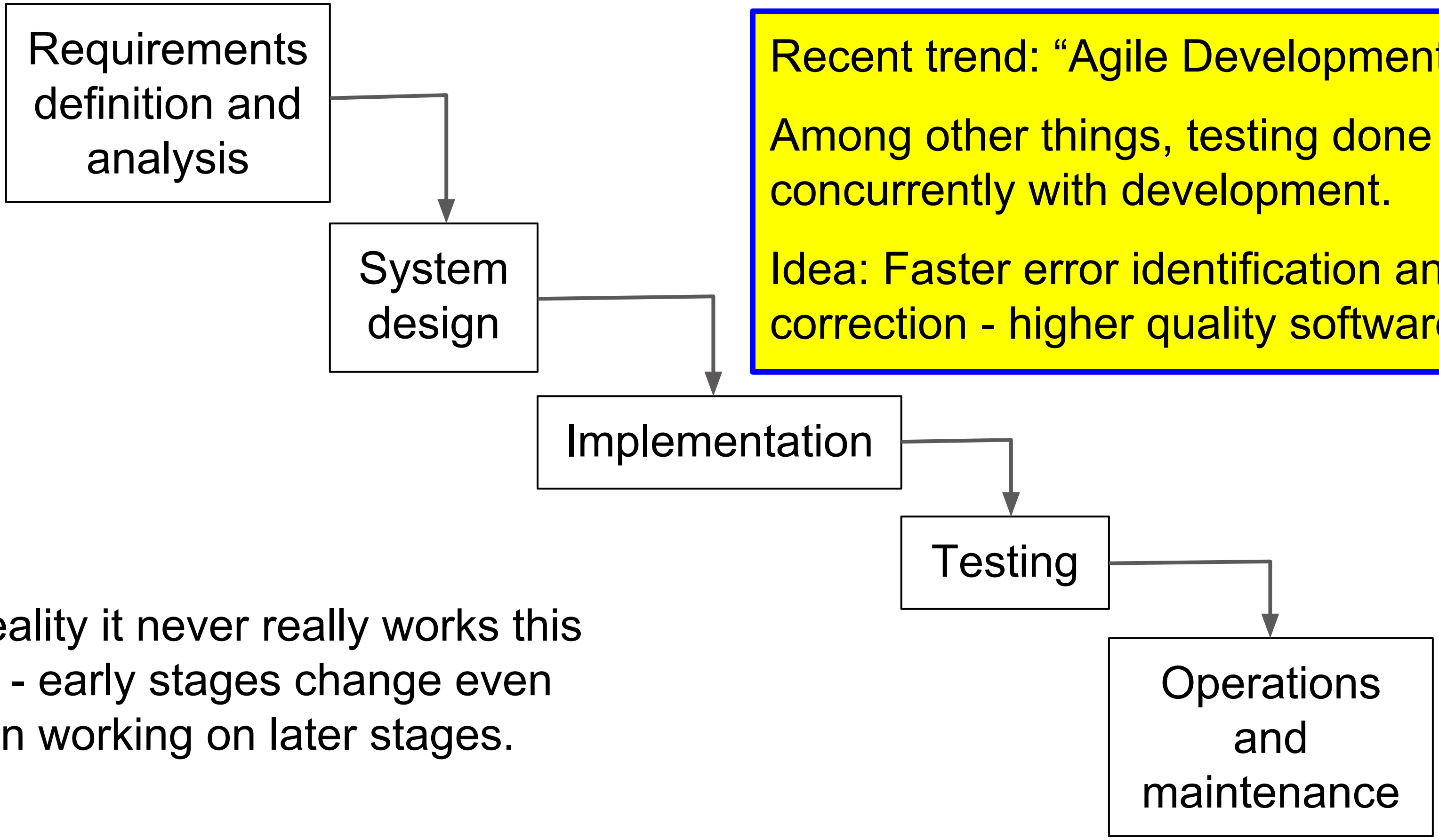
This is what we do in labs.

In reality it never really works this way - early stages change even when working on later stages.

Software Engineering

How to control the complex process of creating software

Traditional “software lifecycle” has well defined phases that feed into each other one-way - called the “Waterfall Model”



Requirements definition and analysis

System design

Implementation

Testing

Operations and maintenance

Recent trend: “Agile Development”

Among other things, testing done concurrently with development.

Idea: Faster error identification and correction - higher quality software?

In reality it never really works this way - early stages change even when working on later stages.

Peer Reviews and Pair Programming

Two ideas for software development...

Peer Reviews: At regular intervals, a developer presents and explains their code to co-workers, who critically review code. Similar to a writer going over drafts with an editor.

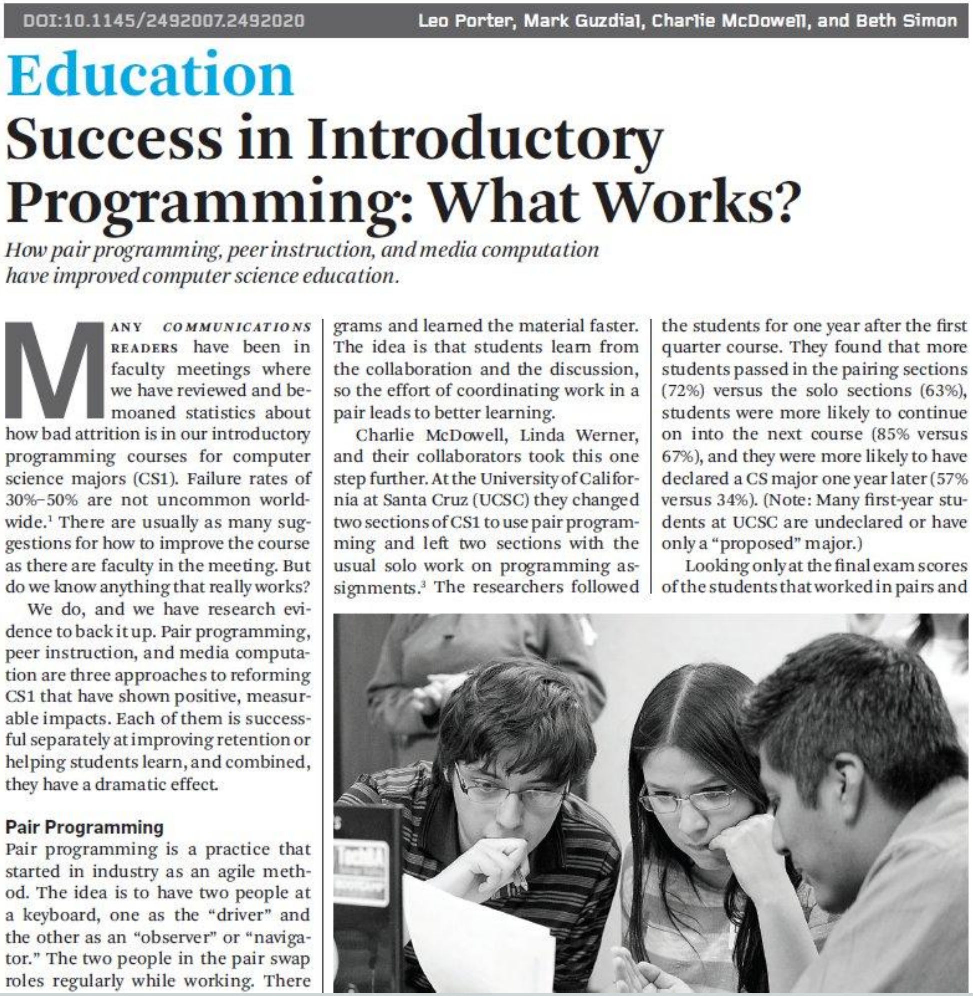
ALL OF THESE PARENTHESES AND SEMICOLONS TOTALLY RUIN THE FENG SHUI OF THE CODE.
SALLY LEARNS AN IMPORTANT LESSON ABOUT INVITING THE RIGHT PEOPLE TO HER CODE REVIEWS.

Pair Programming: Development is actually a collaborative activity - pushes “review” so far back that it is simultaneous with development!

Pair Programming for Learning

From August 2013 *Communications of the ACM*:

Plug: Don't forget our UNCG ACM chapter!



DOI:10.1145/2492007.2492020 Leo Porter, Mark Guzdial, Charlie McDowell, and Beth Simon

Education

Success in Introductory Programming: What Works?

How pair programming, peer instruction, and media computation have improved computer science education.

MANY COMMUNICATIONS READERS have been in faculty meetings where we have reviewed and bemoaned statistics about how bad attrition is in our introductory programming courses for computer science majors (CS1). Failure rates of 30%–50% are not uncommon worldwide.[1] There are usually as many suggestions for how to improve the course as there are faculty in the meeting. But do we know anything that really works?

We do, and we have research evidence to back it up. Pair programming, peer instruction, and media computation are three approaches to reforming CS1 that have shown positive, measurable impacts. Each of them is successful separately at improving retention or helping students learn, and combined, they have a dramatic effect.

Pair Programming

Pair programming is a practice that started in industry as an agile method. The idea is to have two people at a keyboard, one as the "driver" and the other as an "observer" or "navigator." The two people in the pair swap roles regularly while working. There

grams and learned the material faster. The idea is that students learn from the collaboration and the discussion, so the effort of coordinating work in a pair leads to better learning.

Charlie McDowell, Linda Werner, and their collaborators took this one step further. At the University of California at Santa Cruz (UCSC) they changed two sections of CS1 to use pair programming and left two sections with the usual solo work on programming assignments.[3] The researchers followed the students for one year after the first quarter course. They found that more students passed in the pairing sections (72%) versus the solo sections (63%), students were more likely to continue on into the next course (85% versus 67%), and they were more likely to have declared a CS major one year later (57% versus 34%). (Note: Many first-year students at UCSC are undeclared or have only a "proposed" major.)

Looking only at the final exam scores of the students that worked in pairs and

Surprise!

We don't do pair programming because we like to torture you and stick you with terrible partners.

We do it because it helps people learn!

Pair Programming: What to do

Really simple concept:

- Two students, one computer
  - Roles: “Driver” and “Navigator”
  - Driver has keyboard/mouse, but navigator describes how to build solution
- Both students always active
  - *Not “Driver” and “Sleepy passenger”*
- Switch roles regularly
  - At least once per lab activity, if not more often
- Be open and respectful
  - If you don’t like a proposed solution, your job is to either explain why it’s not a good solution or to make the case for a better one - don’t just dismiss it!

Handout: “Fun with Pair Programming!”

Details of Pair Programming in CSC 100

At the beginning of lab:

- Check partner/workstation information sheet posted on lab door
- Find your workstation
- If you don't know your partner, introduce yourself!
- Only one of you will log in to the workstation - decide which one (maybe first to sit down?) and log in to Snap! Cloud account
- If your partner is a no-show by 10:02, let the instructor or lab assistant know

Submitting your work:

- Submit activities as usual through "The G"
- One of you logs in to Canvas and enters names that worked together
- Everyone still does the quiz individually (before the next class)