Programming

When in danger or in doubt, run in circles, scream and shout.

Robert Heinlein, "The Cat Who Walks Through Walls"

A human being should be able to change a diaper, plan an invasion, butcher a hog, conn a ship, design a building, write a sonnet, balance accounts, build a wall, set a bone, comfort the dying, take orders, give orders, cooperate, act alone, solve equations, analyze a new problem, pitch manure, program a computer, cook a tasty meal, fight efficiently, die gallantly. Specialization is for insects.

Robert Heinlein, "The Notebooks of Lazarus Long"

Problem Solving and Programming

Design

Requires intense concentration

When is the best time to fix bugs?

Implementation

Requires mapping design into a programming language

Must chose among alternatives

Testing

Requires a lot of skill, practice

How does problem solving relate to testing?

Debugging

Requires discipline and often benefits from tools

More difficult than testing

Debugging Example 1

A man who has had a heart attack goes every evening to a supervised exercise program. He handles the exercise well during the first 15 sessions, maintaining a heart rate at about 100 beats/minute. In the middle of the 16th session, however, his heart rate suddenly shoots up to 130 beats/minutes.

Although this may not be dangerous, nevertheless, the attendant has him stop exercising and calls the supervising doctor. The man is short of breath but otherwise feels fine. The change in heart rate appears to be his only symptom. What question(s) should the doctor ask?

Debugging Example 2

A man went to wash his face on awakening and found that there was no hot water.

He knew to look for a special feature. He asked his wife whether she had done anything the day before near the boiler.

Her response was in the negative. She added, however, “I didn’t have a chance to tell you, but the oil company sent a man yesterday to clean the furnace.” That certainly looked like a promising hint. A call to the oil company led to the solution of the problem.

Debugging

One of the hardest parts of programming

Strategy 1: Avoid bugs in the first place

- Careful design (clean decomposition)
- Care with syntactic issues (layout, commenting)

Strategy 2: Implement in a series of small steps, and test along the way

- This localizes new bugs to what changed in the program to introduce the bug.

Finding bugs requires a disciplined, deductive approach

Scheduling

Managing large-scale projects involves significant efforts to plan and schedule activities

- It is human nature to work better toward intermediate milestones.

The same concepts can/should be applied to mid-sized projects encountered in class.

- For any project that needs more than a week of active work to complete, break into parts and design a schedule with milestones and deliverables.

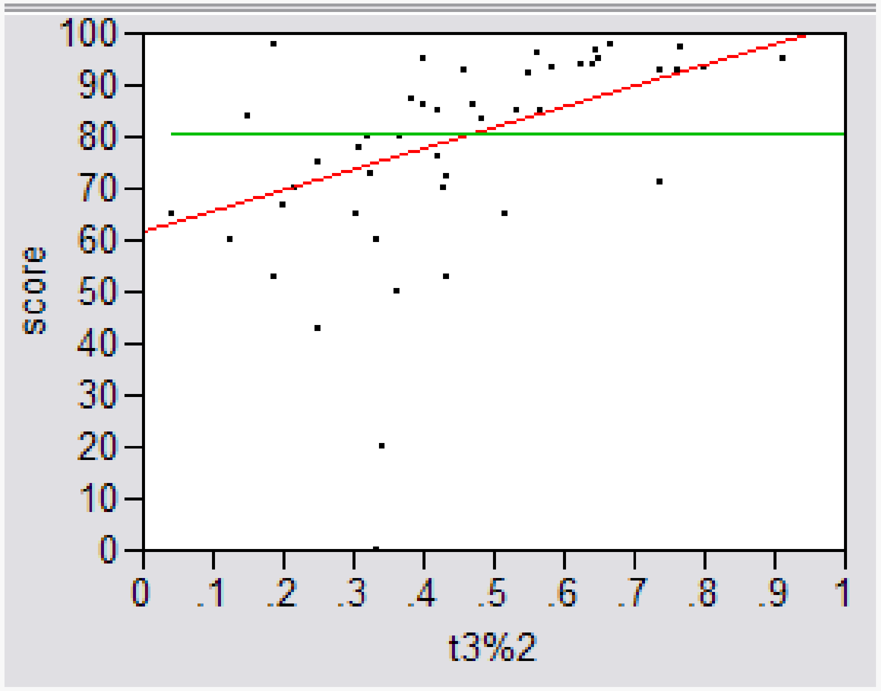
Real Results

CS2606, Fall 2006

3-4 week project cycles

Kept schedule information:

- Estimated time required
- Milestones, estimated times for each
- Weekly estimates of time spent.



T3%2= percent of project completed before last week

Real Results

Results were significant:

- 90% of scores below median involved students who did less than 50% of the project prior to the last week.
- Few did poorly who put in > 50% time early
- Some did well who didn't put in >50% time early, but most who did well put in the early time

Correlations:

- Strong correlation between early time and high score
- No correlation between time spent and score
- No correlation between % early time and total time

What is the mechanism?

Correlations do not necessarily imply a causal relationship

- Do they behave that way because they are good, or does behaving that way make them good?

Spreading projects over time allow the “sleep on it” heuristic to operate

Avoiding the “zombie” effect makes people more productive (and cuts time requirements)

7 Habits of Highly **In**effective Programmers

1. Fail to understand/absorb requirements
2. Design with less than total focus
3. Write disorganized code
   - Style, comments, design
4. Don't program defensively
   - Embed error-checking, state pre- and post-conditions and follow them
5. Bite off more than you can chew
   - During implementation
6. Debug in a random walk
7. Program/debug in zombie mode
   - Don’t start early enough, don’t pace

7 Habits of Highly Effective People

(Covey)

1. Be Proactive: Take initiative, seek new ideas
2. Begin with the end in mind: Have a goal
3. Put first things first: Prioritize, organize
4. Think win/win: Seek mutual benefits
5. Seek first to understand, then to be understood: Learn first, be adaptable
6. Synergize: Make the whole greater than the parts
7. Renewal: Physical, mental, spiritual, emotional