CSE503: Software Engineering

David Notkin
University of Washington
Department of Computer Science & Engineering
Winter 2002

1

What is software engineering?

- In groups of two or three people, take three minutes to write down a definition

2

Key points for definitions

- Full-lifecycle: "womb-to-tomb"
- Multiple people are necessary
  - Many people
  - And people with different skill sets and job descriptions
- Multiple versions of software will be developed
- Economics plays a key role: resources are constrained (cost, time-to-market, etc.)

3

Highest level goal of 503

- Develop a deep understanding of the fact that *software engineering is not a mere matter of programming*

4

What is software engineering *research*?

- Finding ways to identify and better understand problems that are faced in effectively engineering software
- Finding ways to solve these problems
- Neither the problems nor the solutions are cut-and-dried in software engineering research
- Both are much more contextual than in many other areas of computer science research

5

People play a key role

- Many aspects of software engineering focus on how to make the humans involved in engineering software more effective (as opposed to the computer itself)
- People use the software systems (even if indirectly) and this places pressures on the software itself

6

Assessment is complicated

- The contextual and human-oriented nature of software engineering makes it hard to assess proposed improvements
- Some graduate students view this characteristic of software engineering research as sufficiently disturbing to cause them to work in other areas

7

My view is different

- The problems of software engineering are real: really, really real!
- The "softness" of the problems and the solutions make it more interesting, challenging and exciting
  - In understanding the problems, in determining potential solutions, and in assessing their value
- This does cause the answer to many (perhaps most or even all) questions to include the phrase, "Well, it depends…"

8

CSE503: Technical focus

- Much work in software engineering touches on managerial issues: this is essential, since coordinating groups of people over time clearly relates to management
- In this course, we'll focus on technical aspects of software engineering
- That said, it's impossible to draw this line firmly and clearly
  - Near the end of the course, I may cover a few of the more managerial aspects of software engineering

9

CSE503: Two primary objectives

- Provide an overview of some of the most important techniques and approaches that can help in producing better software at more predictable costs
  - Understanding state-of-the-art, which may help you in your own system building
- To lay a foundation for performing research in software engineering
  - Even though not all of you actually will!
  - This means that we will often discuss the intention and effectiveness of techniques and approaches, as well as the techniques and approaches themselves

10

What is your background?

- What's the largest software system you have ever worked on?
- Original developer or maintainer?
- Any products with significant user bases?
- What were the most difficult software engineering problems you faced?

11

What do you want from CSE503?

- Other than "it's a quals course"

12

"Not a mere matter of programming": an example

- Proving programs correct has a 30+ year history
  - Given a *specification* (in a formal logic) and
  - an *implementation* (in a programming language),
  - prove that the implementation *satisfies* the specification
- This has often been considered to be *the* key problem in software engineering

```
{ true }
x: int;
read(x);
if (mod(x,2) = 1) then
  x := x + 1;
fi
{ even(x) }
```

13

But it leaves open key software engineering problems

- Requirements engineering
  - Where did the specification come from? Does it satisfy the needs of the customer?
- Design
  - How does it interact with other parts of the program?
- Evolution
  - What happens if the specification is changed?
- Economics
  - What is the cost of proving correctness?
- Testing
  - Should we rely entirely on the proof?
- …

14

Proving programs correct: redux

- None of these issues eliminate the value of proving programs correct, but they show some of the limitations with respect to engineering large software systems
- Barry Boehm
  - Verification: "Did you build the system right?"
  - Validation: "Did you build the right system?"

15

CSE503: basics

- Reading
  - I'll make papers available in advance
  - give a quick quiz of some sort in class on the contents of the readings
- Discussions on the cse503 email list
  - Sign up using majordomo
  - Address both topics in the course and other software engineering topics that interest you
  - You drive!
- Lectures
  - Specifications, design and architecture, analysis and tools, testing and quality assurance, etc.
  - Tentative schedule on the web

16

CSE503: assigned work

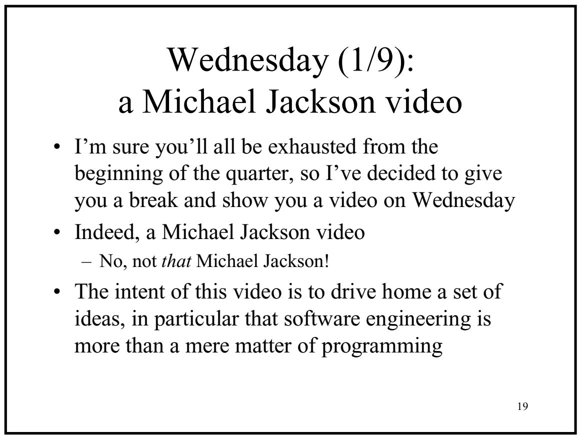
- Two homework assignments [40% total]
  - May be done in pairs
- (1 and 2) or 3 [60% total]
  - A 10-15 page term paper on one of the subjects found on the web page (or by negotiation with me) [30%]
    - Done alone
  - A tool evaluation [30%]
    - May be done in pairs
  - An extensive project
    - Requires my permission and agreement upon a topic
    - Only recommended for those with significant background
    - Done alone

17

No midterm or final exam

- I've never been able to write a good exam on software engineering

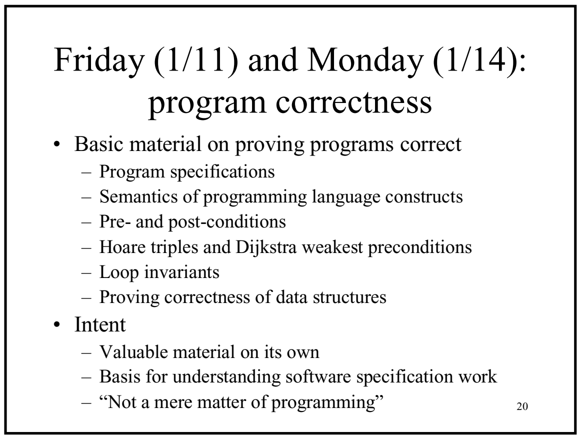
18



Wednesday (1/9): a Michael Jackson video

- I’m sure you’ll all be exhausted from the beginning of the quarter, so I’ve decided to give you a break and show you a video on Wednesday
- Indeed, a Michael Jackson video
  - No, not *that* Michael Jackson!
- The intent of this video is to drive home a set of ideas, in particular that software engineering is more than a mere matter of programming

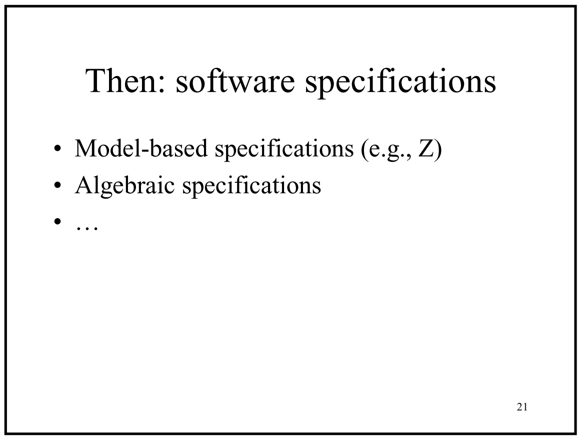
19



Friday (1/11) and Monday (1/14): program correctness

- Basic material on proving programs correct
  - Program specifications
  - Semantics of programming language constructs
  - Pre- and post-conditions
  - Hoare triples and Dijkstra weakest preconditions
  - Loop invariants
  - Proving correctness of data structures
- Intent
  - Valuable material on its own
  - Basis for understanding software specification work
  - “Not a mere matter of programming”

20



Then: software specifications

- Model-based specifications (e.g., Z)
- Algebraic specifications
- …

21