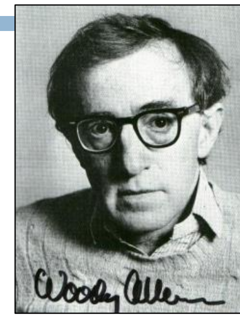
[Michael Jackson]

CSE503: SOFTWARE ENGINEERING

INTRODUCTION

David Notkin
Spring 2011

My favorite software pundit



- Lady #1: "The food in this place is terrible."
- Lady #2: "Yes, and in such small portions."

- This captures much of the confusion about software: it's broadly believed to be of low quality, but there is a voracious appetite for it
- Software engineering R&D must consider both dimensions

When I say "software engineering"

- …what do you think of?
- Shout it out! I'll write them on the board…
- (Can somebody write these down and mail them to me?)

In the beginning

A Story: How much does software weigh?

5

503 11sp © UW CSE • D. Notkin

Back to the beginning

6

- Computers were a more precious resource than were people – "it's the money, honey"
- Working in ways that optimized the use of the computer, even at the cost of significant human effort, was sensible

Aside: What was the most precious computing resource – cycles, memory, bandwidth, …?

503 11sp © UW CSE • D. Notkin

Towards the present

7

- 1968 and 1969 NATO conferences on software engineering
  - Friedrich Bauer chaired it in 1968, with about 50 attendees including Turing Award winners Alan Perlis, Edsger Dijkstra and Peter Naur
- There were increasing difficulties and costs in developing software – the "human" vs. "computer" tradeoff had to be reconsidered

Perlis epigrams (http://www.cs.yale.edu/quotes.html)
- **"Syntactic sugar causes cancer of the semicolon."**

503 11sp © UW CSE • D. Notkin

Quotations from the 1968 "Highlights"

8

The discussions cover all aspects of software including
- relation of software to the hardware of computers
- design of software
- production, or implementation of software
- distribution of software
- service on software.

[T]he report also contains sections on …
- the problems of achieving sufficient reliability in the data systems which are becoming increasingly integrated into the central activities of modern society
- the difficulties of meeting schedules and specifications on large software projects
- the education of software (or data systems) engineers
- the highly controversial question of whether software should be priced separately from hardware

503 11sp © UW CSE • D. Notkin

Quotations on growth rate of software

9

- Helms: In Europe alone there are about 10,000 installed computers — this number is increasing at a rate of anywhere from 25 per cent to 50 per cent per year. The quality of software provided for these computers will soon affect more than a quarter of a million analysts and programmers.
- David: …OS/360 cost IBM over $50 million dollars a year during its preparation, and at least 5000 man-years' investment. TSS/360 is said to be in the 1000 man-year category. It has been said, too, that development costs for software equal the development costs for hardware in establishing a new machine line.
- d'Agapeyeff: In 1958 a European general purpose computer manufacturer often had less than 50 software programmers, now they probably number 1,000-2,000 people; what will be needed in 1978?

- **[This] growth rate was viewed with more alarm than pride**

503 11sp © UW CSE • D. Notkin

The "usual" questions…

10

…that drive software engineering research

- Why does software cost so much?
- Why does software [testing | maintenance | …] cost so much?
- Why are there so many errors in software?
- Why do so many soft… …cts fail?
- Why can't software b… …e hardware or cars or buildings or bridges or …?
- Why can't software engineering be more like *real* engineering?
- Where's Moore's Law for software?
- …

Why does software suck?

503 11sp © UW CSE • D. Notkin

Standish Report 1995

http://www.spinroot.com/spin/Doc/course/Standish_Survey.htm

11

- U.S. spends more than $250 billion annually on IT application development
- The average cost of a devel… project ranges from $434K (… small) to $2.3M (for large) p…
- 31.1% of projects will be c… before completion
- 52.7% of projects will cost 1… their original estimates
- The failure to produce reliable software to handle baggage at the new Denver airport [cost] the city $1.1 million per day
- "A great many of these projects will fail. Software development projects … chaos, and we can no longer … the three monkeys -- hear no …, see no failures, speak no …"
- "… of these failures and … are just the tip of the … iceberg. The lost … unity costs are not measurable, but could easily be in the trillions of dollars."
- "One just has to look to the City of Denver to realize the extent of this problem."

503 11sp © UW CSE • D. Notkin

"Software's Chronic Crisis" by Gibbs

Scientific American September 1994

12

"To veteran software developers, the Denver debacle is notable only for i… …ies have shown that for every six ne… …oftware systems that are put into op… …ers are canceled. The average softwa… project overshoots its schedule by half; … generally do worse. And some thr… all large systems are 'operating failures' that either do not function as intended or are not used at all."

503 11sp © UW CSE • D. Notkin

Standish Group redux

http://www.infoq.com/articles/Interview-Johnson-Standish-CHAOS

13

- More recent Standish Group reports show some improvement in
- The, however, r lear about the continuing pres ware crisis.
- Jim Johnson, th hairman of the Standish Grou "People know that the more common scenario in our industry is still: over budget, over time, and with fewer features than planned."

503 11sp © UW CSE • D. Notkin

Software lifecycle costs

14

Evolution/maintenance ≅ 90%

http://www.cs.jyu.fi/~koskinen/sm

- "The relative cost fc maintaining softwar managing its evoluti represents more tha its total cost"
- "[A]lthough there hc much empirical rese this particular area, the magnitude of the maintenance cost effects is clearly identifiable."

Testing/verification ≅ 50-75%

Hailpern & Santhanam, IBM Sys. Journal 2002

a typical commercial velopment organization, the it of providing [assurances functional and non- ictional performance] via propriate debugging, ting, and verification tivities can easily range from 50 to 75 percent of the total development cost."

503 11sp © UW CSE • D. Notkin

So, these data show that…

15

- …software costs too much on an absolute basis
- …software lifecycle phases cost too much on a relative basis
- …software projects are cancelled too often

"When you can measure what you are speaking about and express it in numbers, ng about it; but when you , when you cannot express knowledge is of a tisfactory kind: it may be knowledge but you have houghts, advanced to the stage of science." —*Lord Kelvin*

In a nutshell: qualitative assessment is an "absolute zero"

503 11sp © UW CSE • D. Notkin

But how do we measure "too"?

16

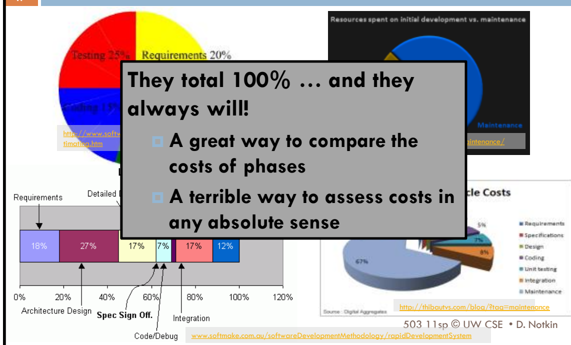
…software costs *too* much…, software lifecycle phases cost *too* much …, software projects are cancelled *too* often…

- We don't – we accept that we're just too *X* and we should just get better at *X*
- What would be ideal or even acceptable absolute costs? Relative lifecycle costs? Project cancellation rates?
- If you believe firmly in measurement, then this should be as unsatisfactory as any other kind of lack of measurement

503 11sp © UW CSE • D. Notkin

Per-phase "pie" charts: what do they have in common?

17



They total 100% … and they always will!

- **A great way to compare the costs of phases**
- **A terrible way to assess costs in any absolute sense**

Testing and evolution

18

- Why might we care about what "too" means?
- Is 50-75% too high for testing? What would be acceptable? Why? Is 0% a good goal? Are there benchmarks from other engineering disciplines and, if so, should we believe they may be analogous?
- Is 90% for evolution and maintenance too high? What would be a good goal? 50%? 10%? 0%? Or is it too low, and 99% would be a better goal? Even the desired directionality for this is not entirely clear.

Cyber-physical systems

19

- Dizzying increase in physical systems that have a significant softw... medical devices, spacecraft and ... nces, automobiles, bridges and bui... s, and *many* more.
- When these syst... er the Denver airport, the Mars Polar L... ne V, and *many* more – they are genera... oftware failures
- In an important sense, this is accurate – specifically, in most cases, a variant on the software could have eliminated the failure.

Mars Polar Lander:

$100M lost (plus opportunity costs)

20

- "…the most likely cause of the failure of the mission was a software ... akenly identified the vibration cause ... ment of the lander's legs as being c... hicle touching down on the Martian ... g in the vehicle's descent engine ... while it was still 40 meters above the surface, rather than on touchdown as planned." [Wikipedia]

Co-design decisions

21

- Allocation of function to physical vs. software components is critically important
- In some domains these decisions come from those with more know-how on the physical side
- Even more commonly, these decisions are made with a clear view that much complexity can and should be pushed into the software
  - Thus, it is tautological that software would cause more problems in cyber-physical systems simply because it is "assigned" greater complexity.
- That is, Increasing the complexity of the software is (surely at times) a fine decision – but one should not then later be surprised at increased risks and costs

Lead time for physical manufacturing

22

- Physical components generally require a long lead time for design and manufacture; for practical reasons, this is done concurrently with software production
- The physical components and their means of production necessarily and practically become more stable and more costly to change over time
  - Changes made at later stages tend to be much more costly to fix
  - Just like software ☺ but even *more* costly!
  - If you really think software is too hard to change, try changing the physical components instead!

Changes to software requirements

23

- So unexpected shortcomings on the physical side are often handled by changing the software requirements
- This adds complexity and cost to the software because numerous design and implementation decisions have already been made during the concurrent development
- To accommodate flaws in the engineering of the physical components, even more complexity is injected into the software
- "Better" software can generally overcome these flaws, but the need to do so is induced by weaknesses on the physical side

Software is last

24

- Testing software on the physical system instead of on simulators, mockups, etc. may be cheaper and easier
- When software is changed to overcome physical flaws, the software is necessarily later
- There is, quite reasonably, a perception that software is indeed "soft" compare and thus it seems to be able to withstand changes until (and often after) the last moment
- But just because it is last doesn't mean it is (entirely) at fault

Software: breaking [Moore's] law" [Wikipedia]

25

"… exponentially improved hardware does not necessarily imply exponentially improved software performance to go with it. The productivit
softw
assu
incr
fitfully ove
"

- The performance of software and software developers is compared to transistors on an integrated circuit
- Wha human activity has matched growth of Moore's Law?
  - Do we (or should we) compare the erformance of trains to their tracks? f train designers to their trains?
- t other technology has ned the growth of Moore's Law? Batteries, displays, ??? IC circuits are a (wonderful and probably) singular technology

❖ Would you rather take the bus to work or your lunch?
❖ Would you rather be in love or in Tucson?

Blame isn't the goal

26

- Simply blaming software for the problems because it *could* fix the system and because it *was* (naturally) last to be stabilized ***cannot*** easily lead us to better solutions to costly fiascos
- Of course we as software engineering researchers and engineers must work hard to do better – indeed, much better
- We must not, however, let the playing ground be set in a way that is not helpful towards achieving critical goals

Value: missing from most discussions

27

- Value is definitely hard to measure – but the world has surely agreed that software has value, or else companies that produce and sell it would not exist!
- We need much more work in this area
  - Barry Boehm, Kevin Sullivan, Mary Shaw, and others have worked on software engineering economics – this is crucial but very difficult
- But we have to remember that the reason software is important is because it provides value – real value to society, to the economy, to people – and if it didn't, nobody would care about cost, dependability, etc.

Reprise: Standish '95: U.S. spends > US$250B annually on IT application development

28

- Software industry (2008, worldwide) US$304B
  DataMonitor via Wikipedia
- Advertising industry (2009, Worldwide) US$445B
  http://www.plunkettresearch.com/advertising%20branding%20market%20research/industry%20statistics
- Travel industry (2008, Worldwide) US$944B Wikipedia
- Porn industry (2004, Worldwide) US$57B
  http://www.toptenreviews.com/2-6-04.html
- Size is an inherently limited way to assess how well an industry is doing…

Different kinds of questions…

29

…that could and should drive software engineering research

- What <u>should</u> software systems cost to design, build, maintain? Can we find a useful lower bound?
- If we had infinite cycles to help software engineers, what problems would <u>still</u> exist?
- When changing software, we assume that new behavior can be arbitrarily far from old behavior. What if we instead focused on the <u>common-case</u> – a small Δ?
- Under what conditions is it reasonable/unreasonable to characterize a class of software systems as <u>similar/dissimilar</u>?
- How should we <u>legitimately</u> assess and achieve important properties that are – even if we dislike it – not binary, not efficiently computable, not even precisely defined, etc.?
- …

Coarse course expectations…

30

Overall objective – allow you to focus on the subareas and dimensions of software engineering that you find most interesting and/or most beneficial to you

- "History" assignments (#1 and #2)
- Project #1: Tool use and evaluation (research) **or** software building (development)
- Project #2: Primary research project **or** secondary research project
- Some assigned work TBA
- Course participation
- No examinations

Just to show some awareness…

31

http://news.softpedia.com/news/SCADA-Software-Increasingly-Under-Scrutiny-by-Security-Researchers-191525.shtml

"Supervisory control and data acquisition (SCADA) software is responsible for monitoring and controlling equipment in industrial facilities, including oil and gas refineries, power and water processing plants, factories, etc.

Attacks against SCADA software moved from theoretical to practical last year with the discovery of Stuxnet, a highly sophisticated industrial espionage malware whose purpose was to destroy uranium enrichment centrifuges at the Iran's Natanz nuclear plant. …

[Researcher] Rubén Santamarta released an exploit for a remote code execution vulnerability affecting a Web-based SCADA product called BroadWin WebAccess.

His decision to go public was the result of the vendor denying the existence of a problem. *'I contacted ICS-CERT [Industrial Control Systems Cyber Emergency Response Team] to coordinate with Advantech but the vendor denied having a security flaw. So guys, the exploit I'm releasing does not exist. All is product of your mind,'* the researcher says ironically. …"