

CSci 780 Advanced Software Engineering

No Silver Bullet: Essence and Accidents of Software Engineering

by Frederick P. Brooks, Jr.

9/8/2003

1

Outline

- The Software Werewolf
- Essential and Accidental Difficulties
- Newest Fads Aren't Silver Bullets
- Is There Hope?
- Things To Try
- Summary
- Discussion

9/8/2003

2

Fred Brooks

- Project manager of IBM OS/360
- 1999 Turing award winner
- Now chair of UNC-Chapel Hill CS dept



9/8/2003

3

The Software Werewolf

- Software is like a werewolf—it looks normal until the moon comes out and it turns into a monster
- Missed deadlines
- Blown budgets
- Buggy software
- We want the silver bullet to kill the monster

9/8/2003

4

There Is No Silver Bullet!

"... no single software engineering development will produce an order of magnitude improvement in programming productivity within ten years"

9/8/2003

5

Essential vs. Accidental Difficulties

- Difficulties either *essential* and *accidental*
- Essential: a characteristic of software
 - *Constituting or being part of the essence of something; inherent.*
- Accidental: a problem in today's production methods
 - *Of or relating to a property, factor, or attribute that is not essential.*
 - Note: This doesn't mean "by chance"!

9/8/2003

6

Accidental Difficulties Already Solved

- High level languages
- Time sharing
- Integrated programming environments
 - Libraries, file formats, pipes & filters
- *What else?*

9/8/2003

7

There is No Silver Bullet

- To get 10x improvement
 - Accidental difficulties would have to be 9/10 of the overall problem
 - Tools and techniques would have to reduce these difficulties to 0
- Brooks: There is no silver bullet
 - Believes first point is not true (probably much less)
 - Believes second point is highly unlikely to happen

9/8/2003

8

Complexity

- No two parts are alike
- Huge number of states
 - "Software systems have orders of magnitude more states than computers do"
- Module interactions scale nonlinearly with size
- Can't know the whole domain, process, or system
- Unlike other disciplines, we can't abstract away the complexity because it is essential

9/8/2003

9

Consequences of Complexity

- Communication overhead: cost overruns, schedule delays
- Large number of states: unreliability
- Complex function: poor usability
- Complex structure: poor maintainability, security risks

9/8/2003

10

Conformity

- Software must conform to arbitrary limitations imposed by humans (e.g. business rules)
- Software arrives late in system design
- Software viewed as most changeable
- It's hard to plan for arbitrary changes that will occur late in development

9/8/2003

11

Changeability

- Software is easier to change than hardware
- Asked to change during development
- Asked to change after deployment
 - New features
 - Software lives longer than hardware
- People underestimate difficulties of change

9/8/2003

12

Invisibility

- The code is invisible and *unvisualizable*
- Structure is terribly complex
- Structure is hidden
- There's only the external input/output view

9/8/2003

13

No "Silver Bullet" in 1987-1997

- Languages like Ada: Remaining benefit small
- OOP: Good, but essential complexity remains
- Artificial intelligence: Not generally applicable
- Automatic programming: Not generally appl.
- Graphical programming: Unvisualizable
- Program verification: Specification still hard
- Environments & tools: Remaining benefit small
- Workstations: Faster compiles, but that's it

9/8/2003

14

Is There Any Hope?

- Compare to medicine
 - Throw out simple, fast fixes for demon-possession and four humours
 - Apply persistent effort to slowly eradicate disease
- Compare to chemistry
 - Throw out alchemy
 - Spend years to understand atoms, then learn to synthesize gold

9/8/2003

15

Yes!

"A disciplined, consistent effort to develop, propagate, and exploit these innovations should indeed yield an order-of-magnitude improvement. There is no royal road, but there is a road."

9/8/2003

16

Things To Try

- Reuse: "buy, don't build"
- Requirements refinement and rapid prototyping
- Incremental development: ("grow don't build")
- Better SE training and nurturing of greatness
- Multiple coordinated techniques

9/8/2003

17

Summary

- No *one advance* will give a 10x improvement
 - All the accidental difficulties have been solved
 - No one advance will address essential difficulties
- But perhaps we can learn to better exploit what we've learned
- Since then (oops) [MMM]
 - Information hiding
 - Incremental development
 - SE specialization
 - Families

9/8/2003

18

Discussion

- Did he predict correctly?
- What other essential properties?
 - Complexity, conformity, invisibility, changeability
- What other "hopes for the silver"?
- Examples of progress?
- Impact of his suggestions?
 - Buy-don't-build, requirements refinement & rapid prototyping, incremental development, nurturing designers, multiple techniques