

CSci 780 Advanced Software Engineering

No Silver Bullet: Essence and Accidents of Software Engineering

by Frederick P. Brooks, Jr.

9/10/04

1

Outline

- No Silver Bullet for the Software Werewolf
- Essential vs Accidental Difficulties
- Discussion #1
- Newest Fads Aren't Silver Bullets
- Is There Hope?
- "No Silver Bullet" Refired
- Summary
- Discussion #2

9/10/04

2

Fred Brooks

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



9/10/04

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

9/10/04

4

There Is No Silver Bullet!

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

9/10/04

5

Essential vs. Accidental Difficulties

- Difficulties either *essential* and *accidental*
- Essential: a fundamental quality 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"!

9/10/04

6

Accidental Difficulties Already Solved

- Raw materials: High level languages
- Computer access: Time sharing
- Using multiple tools: Integrated programming environments
 - Libraries, file formats, pipes & filters

9/10/04

7

Analysis of Difficulties

- 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
 - Believes first point is not true (probably much less)
 - Believes second point is highly unlikely to happen

9/10/04

8

Concept versus Representation

"I believe the hard part of building software to be the specification, design, and testing of this conceptual construct, not the labor of representing it and testing the fidelity of the representation."

9/10/04

9

Characteristics of Software

9/10/04

10

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

9/10/04

11

Consequences of Complexity

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

9/10/04

12

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

9/10/04

13

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

9/10/04

14

Invisibility

- The code is invisible and *unvisualizable*

- Structure is terribly complex

- Structure is hidden

- There's only the external input/output view

9/10/04

15

Discussion #1

- Is Brooks' claim true for 40 years? 100?
- Are these *really* unique to software?
 - Complexity, conformity, invisibility, changeability

- What other essential properties?
- What other accidental difficulties solved?
- What other "hopes for the silver"?

- Have we had significant progress since 1987?

9/10/04

16



The "Silver Bullets," and Real Hope

9/10/04

17

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

9/10/04

18

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

9/10/04

19

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."
- NSB is not pessimistic. But so what if it is!
 - Einstein said we can't go faster than speed of light
 - Don't waste time searching for quick solutions

9/10/04

20

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

9/10/04

21



Reflections on "No Silver Bullet"

9/10/04

22

"No Silver Bullet" Refired [MMM 1995]

- Lots of controversy and rebuttals in 1987
 - (But no dramatic effects by 1995!)
- Clarifies "accidental" vs. "essential" for software
 - Essential: mental crafting of the conceptual construct
 - Accidental: implementing that concept
- He believes that rendering the code is down to half or less of total effort
 - (And no one has claimed it to be 9/10!)

9/10/04

23

Recap: Essential Difficulties not Hopeless

- Example: complexity
 - Can we create order from chaos? e.g. complex molecules replaced with 3 laws of thermodynamics
 - Simplify the implementation: hierarchical abstraction, incremental construction
- Example: invisibility
 - "... one needs multiple diagrams, each covering some distinct aspect..."

9/10/04

24

Brooks on Brooks

- “Buy don’t build”
 - Underestimated customizability
- Object-orientation
 - Very promising, but we need to scale up from stacks
 - Need to pay large up-front training costs
- Reuse
 - Need organizational support
 - Need good design and good documentation
 - Must address the “vocabulary problem”

9/10/04

25

Summary

- No *one advance* will give a 10x improvement
 - All the accidental difficulties have been solved
 - No one advance will address essential difficulties
- But maybe better exploit what we’ve learned
- ...or just roll up our sleeves and get to work
- Refired: We are making incremental progress, and that’s still all we can hope for

9/10/04

26

Discussion #2

- Did he predict the technologies correctly?
- Impact of his suggestions?
 - Buy-don’t-build, requirements refinement & rapid prototyping, incremental development, nurturing designers, multiple techniques
- Which definition do you prefer?
 - Mental construction vs. inherent property
 - Tangible expression vs. non-inherent property
- Are there “laws of software?”
- What is the best combination for domain X?

9/10/04

27