The Billion-Dollar Lesson

Datto Infrastructure

Presented By: Fred Mora

Agenda

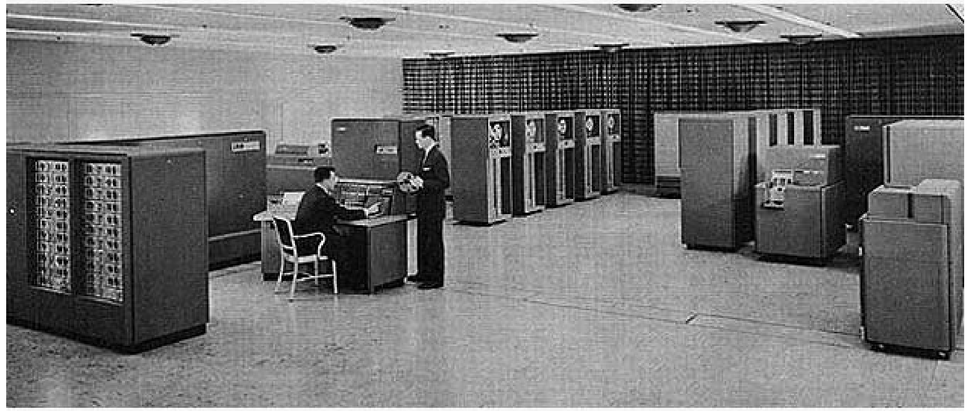
- The lesson: Complexity kills
- The classroom: IBM's OS/360 project
- The student: Fred Brooks
- The textbook: The Mythical Man-Month
- How software developers integrated the lesson: OOP

Complexity kills

- A limit of human intellectual performance is short-term memory
- In software, this means "what do I need to know in order to make use of something"
  - Very often, an existing tool or library is so complex we are compelled to write our own
  - And thus, we constantly reinvent the wheel
- But how bad is large scale complexity, really?
- Surely, a high-level team can deal with it?
- Well... Not really.

IBM's OS/360 project

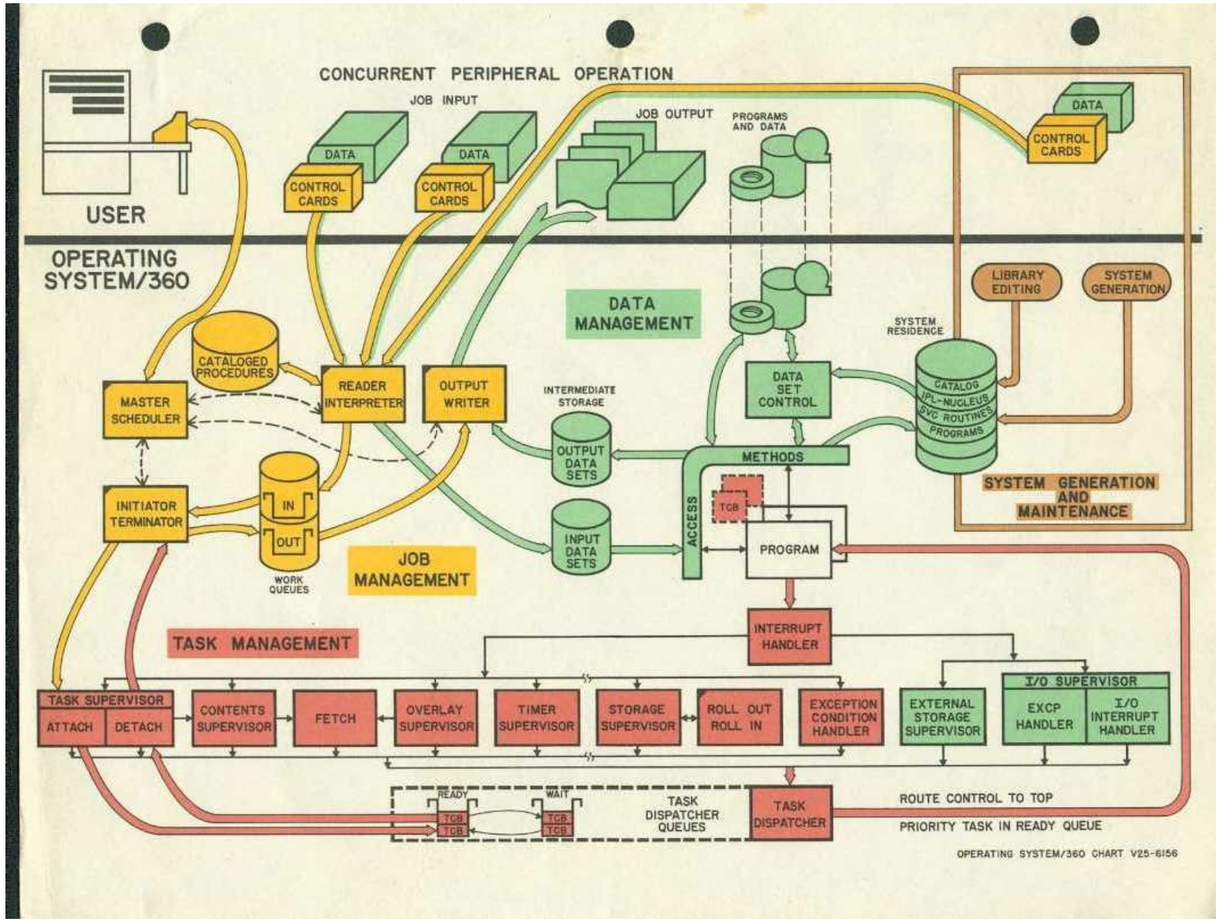
- In the 60s, each new mainframe model was completely different from all others
- New architecture, instruction set, programming language, OS, tools...
  - Huge costly problem
  - IBM formed a task group to create a solution
  - They were told to stay in Stanford, CT, motel for 8 weeks to come up with a recommendation
  - Their report outlined a future general, evolutive architecture
- From the task group report, IBM decided to create a new family of machines called IBM 360.
- Mainframes of this family would range from very low to very high end yet be compatible.
- The common OS was called OS/360.



http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/system360/words/

OS/360 early problems

- The hand-picked team of OS developers was very large
- The hardware was simulated (at slower speeds) on existing mainframe
- System 360 assembly code could be run and tested
- Early on, communication problems appeared:
  - Initial tests showed memory management problem
  - Coders used the worst possible way to manage memory
  - This was because management discouraged use of precious core memory
  - But disk paging is even worse.



OS/360: The "fix"

- Clearly, problems were due to incomplete understanding of memory alloc
- Solution: more doc!
- Each subsystem was therefore documented in often-updated doc formatted on mainframes and printed in 3-ring binders
- Deltas were distributed every day
- A vacation or absence meant hundreds of pages of delta to remove and insert.
- The doc provided low-level details of each subsystem
- Predictable result: TMI

OS/360: The outcome

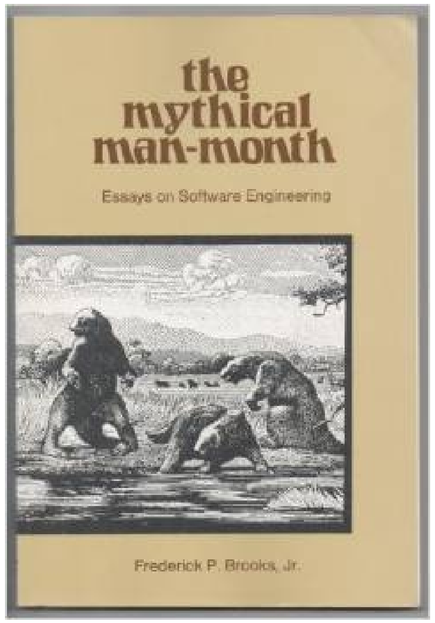
- OS/360 ended up being extremely costly.
- It was considered a success, but remained buggy
- It reached a point where each fix introduced another bug
- The hardware was remarkable, though.
- S/360 and OS/360 were introduced in 1964
- IBM hired tens of thousands of new employees and sold thousands of S/360 (1000 per month as early as 1966)

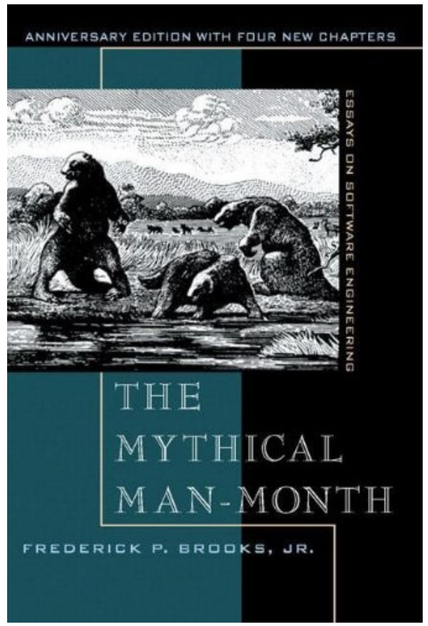
Fred Brooks

- Fred Brooks was the manager of the OS/360 project
- He was interested in software architecture and methods
- He considered that OS/360 was a costly lesson for all developers
  - The lesson is applicable to all software projects
  - Tuition paid by IBM – enjoy.
- Brooks wrote a book summarizing the teachings of OS/360: *The Mythical Man-Month*

The Mythical Man-Month

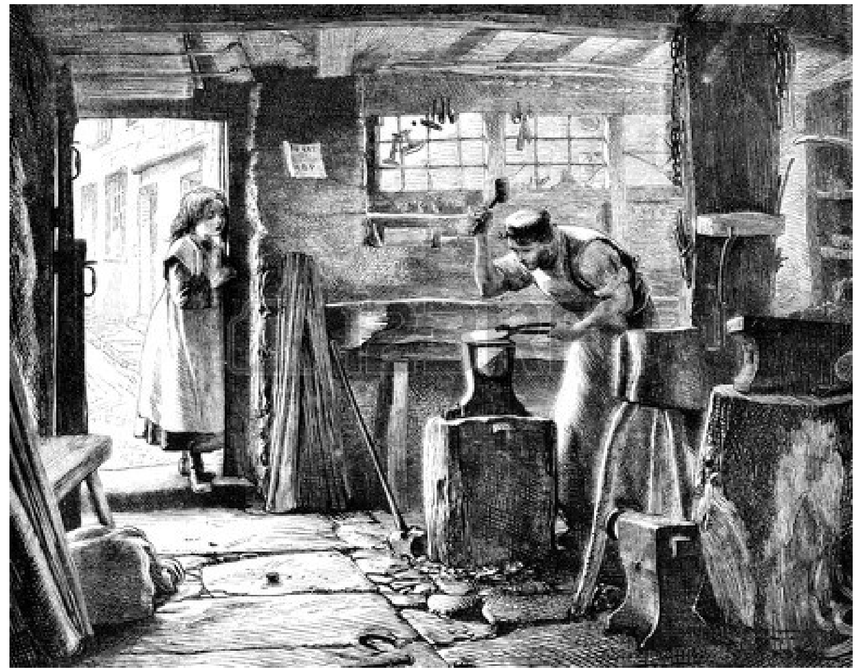
- Published originally in 1975
- Considered required reading for all budding software developers
- Sold 250,000 copies – astounding for a software project management book
- Twentieth Anniversary Edition, augmented, published in 1995
- Sadly, 90% of the pitfalls and problems reviewed in the book are still current.





Still the same old problems

- The MMM gives examples in archaic languages such as PL/1...
- …But the problems are still the same!
- Since 1975, software as a discipline has seen very little progress considering its importance
- Computer *science*? HA!
  - Scientists review each other's work. Developers work in secrecy.
  - Scientists look at what works. Developers keep reinventing square wheels
  - Scientists have a strict terminology. Developers have an imprecise, trendy jargon.
- Software is at best a cottage industry of craftsmen, not a scientific discipline.



OOP, the answer to the OS/360 lesson

- Developers are still making most of the same mistakes since 1975.
- But one mistake (information overload) now has a solution (sort of): OOP
- Object-Oriented Programming lets a library user focus on external interfaces
- Class internals are, by design, black boxes
- Data integrity can be guaranteed through accessors
- But OOP models clash with other programming models. Example: RDBMS
  - Relational DB tables can be modelled though Object Relational Mapping (ORM)
  - But ORM still forces you to be aware of the underlying SQL, so it's a limited solution.