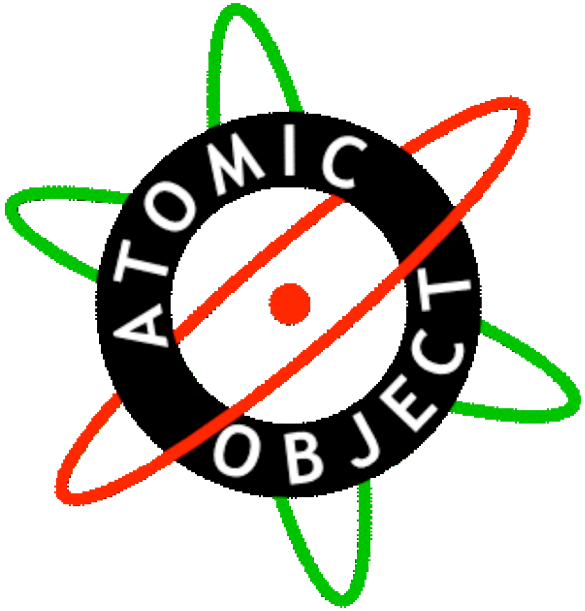
Software Engineering and the Software Crisis



Carl Erickson, PhD

Atomic Object LLC

State of our industry

- 2004 Standish Group study
  - 30% total failure, cancelled
  - 50% over budget
  - 90% late
- Chaos report, 1994
  - 31% cancelled
  - 53% more than 2x over budget

Status quo

"Our challenge is to get our software to the point that people expect it to work instead of expecting it to fail."

Jim Larus, leader of software quality project at Microsoft Research

MIT Technology Review, April 2003

The price we pay

NIST - industry losses

- $60 billion a year

Doesn't even count…

- government
- education
- home
- loss of life

JAS-39 Gripen

Therac-25

Denver Airport

Ariane 5

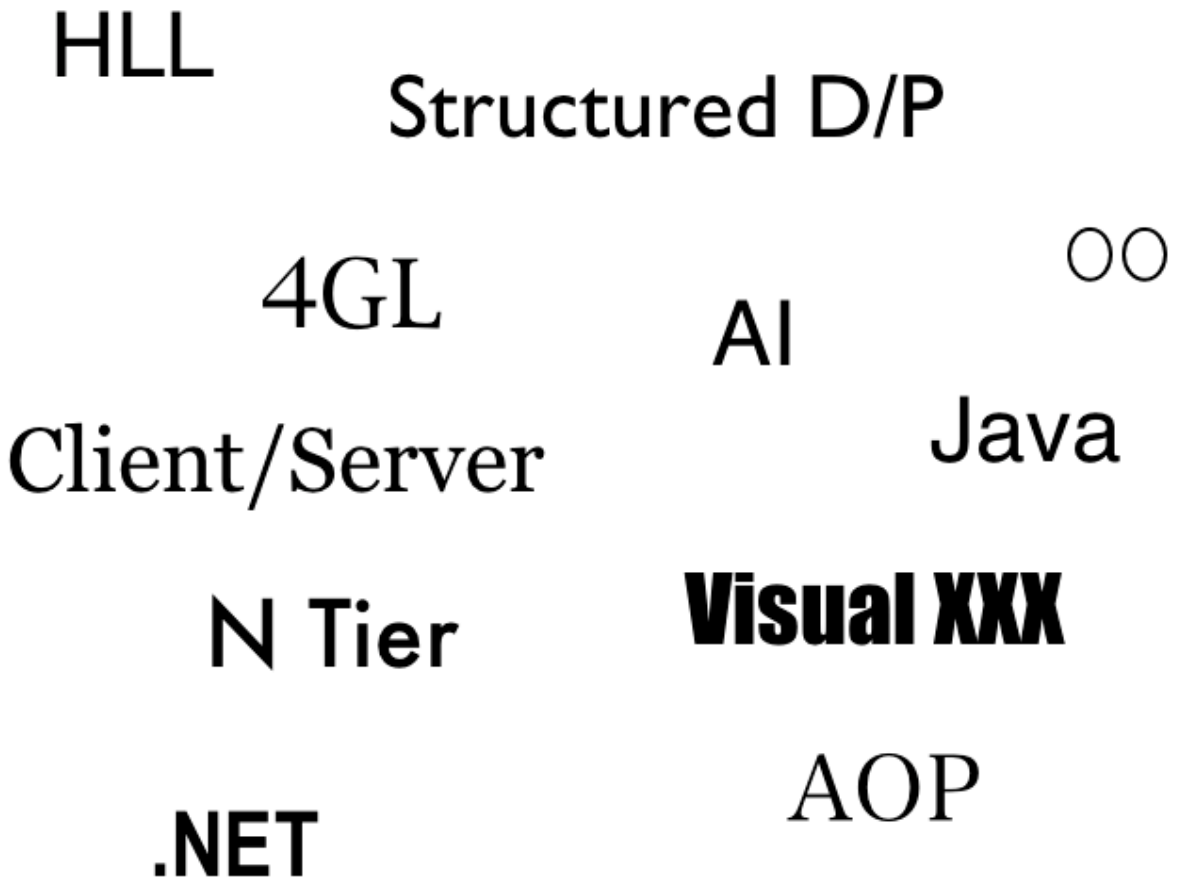
USS Yorktown

FBI Virtual Case File

Successful software projects

- Paid for itself per the business needs (ROI)
- Met important deadlines
- Produced software that
  - satisfies end users
  - is stable and reliable
  - performs acceptably
  - is extensible and maintainable
- Left developers and managers happy

To the rescue?



Scientific Management

"In the past the man has been first;
in the future the system must be first."

Frederick Taylor


Clones

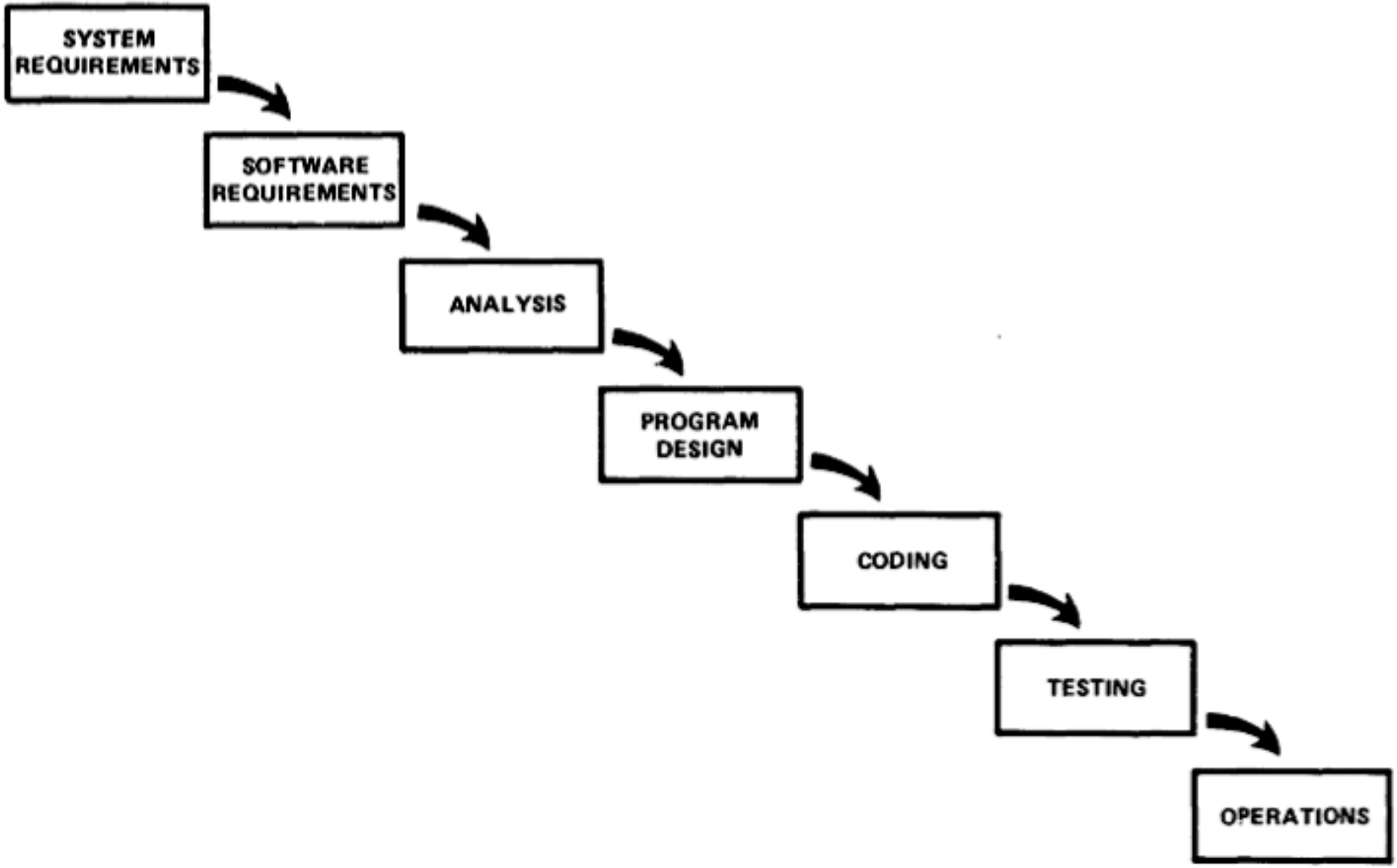


Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

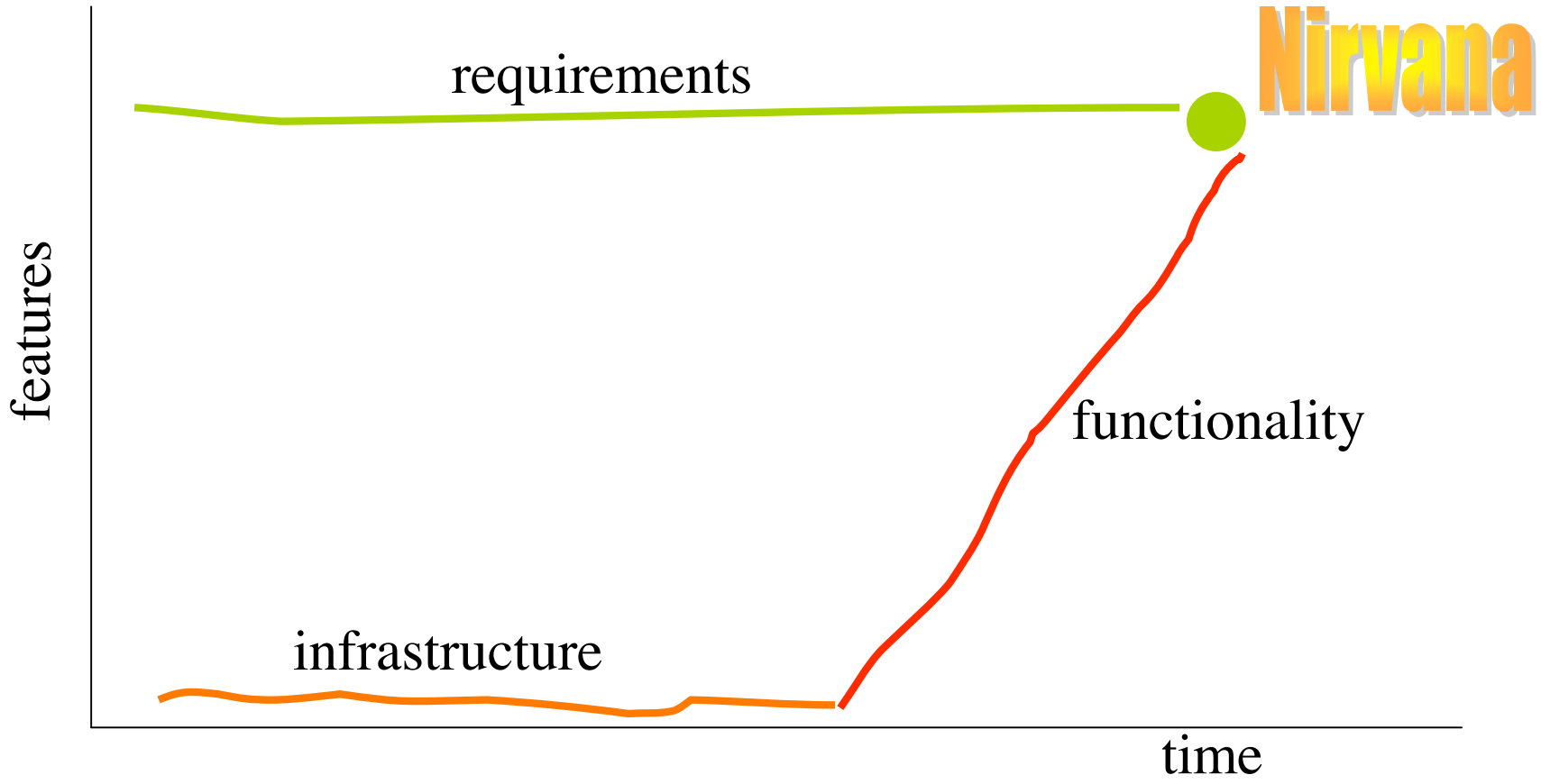
Page 2

"I believe in this concept, but the implementation described above is risky and invites failure."

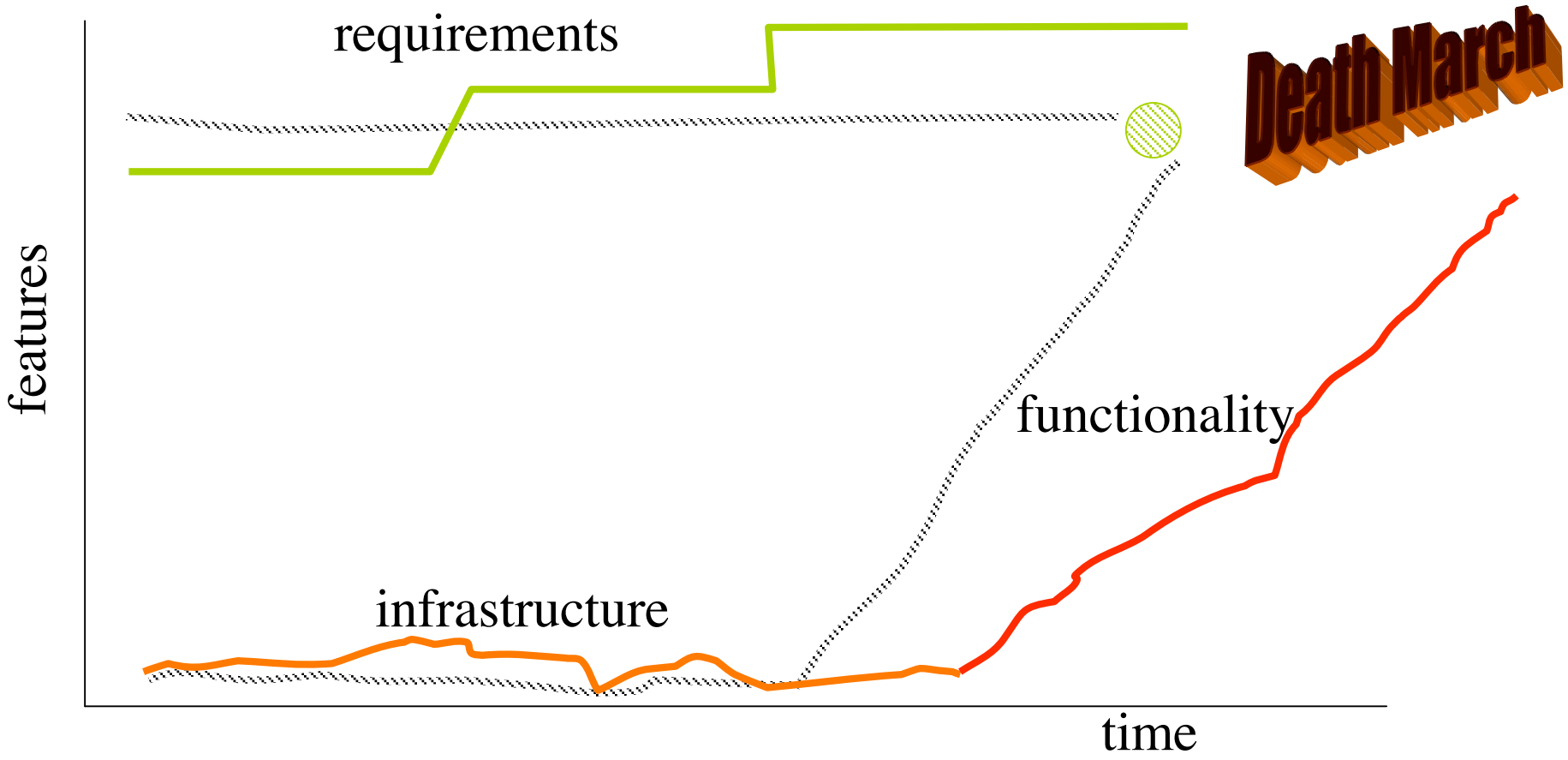
Winston Royce
"Managing the Development of Large Software Systems"
IEEE WESCON, August 1970

(Ron Jeffries inspired these graphs)

Waterfall according to plan



Waterfall reality



Heavyweight processes

Emphasis on non-source artifacts

- diagrams, models, documents

Formal communication

Driven top down

Means/end inversion

The myth of BUFD

Big Up Front Design
(requirements, analysis, modeling, and design)
is a necessary and sufficient condition for project success.

A small problem…

For any interesting system,
the requirements of a solution
are unknowable in advance
and out of context.

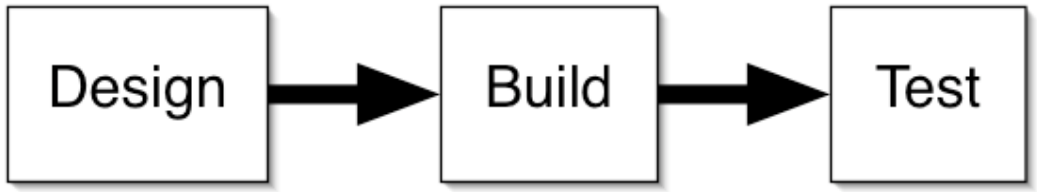
Why BUFD is flawed

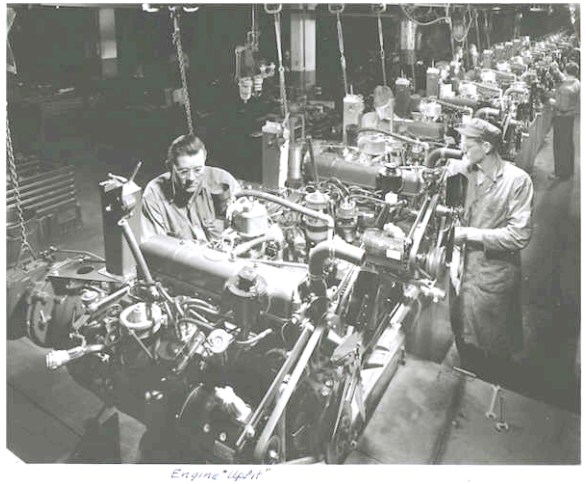
Monkeys and 1 year olds

Editing vs Writing

Requirements vs Solutions

Software != Widgets





Impedance mismatch

Software development

- chaotic,
- unpredictable,
- craft-like,
- cyclical,
- contextual

Heavyweight processes

- rational,
- require prior knowledge,
- scientific,
- linear,
- isolated

Booch's Observation

Complex working systems

- There is some hierarchy to the system.
- The primitive components of a system depend on your point of view.
- Components are more tightly coupled internally than they are externally.
- There are common patterns of simple components which give rise to complex behavior.
- Complex systems which work evolved from simple systems which worked.

Roots of Software Engineering

- Huge projects (1000s of person years)
- Engineering projects (hardware, software, systems)
- Idle programmers
  - waiting for hardware, what can they do?
  - finishing software as soon as possible

IEEE Definition

Software engineering is the application of a systematic, disciplined, quantifiable approach to development, operation, and maintenance of software; that is, the application of engineering to software.

Excellent Results

- NASA Space Shuttle control software
  - 420,000 lines of code
  - 1 error each in last three versions
  - 17 errors total in 11 versions
- Commercial projects of same size
  - 5000 errors
- But…
  - The highest $/loc of any project

Good Enough Software

- What happens when…
  - you don't have NASA's budget
  - bugs don't kill people
  - you must offer lots of features
  - users will tolerate bugs
  - it helps being first to market

The Human Element

- Systematic, disciplined, quantifiable
  - Focus on mechanical engineering analogy
  - Hides the importance of individuals
- Fred Brooks, Mythical Man Month (1975)
  - 200 programmers total, best and most experienced 25 as managers
  - Fire the 175 and let the 25 program

QSM Study: Team Size

- QSM
  - Consultancy specializing in measuring, estimating, and controlling software dev
  - Database of 4000+ projects
- 2005 study on schedule vs team size
  - 564 information systems projects since 2002
- Divided into small (< 5) and large (> 20)
  - By team size

QSM Study, cont.

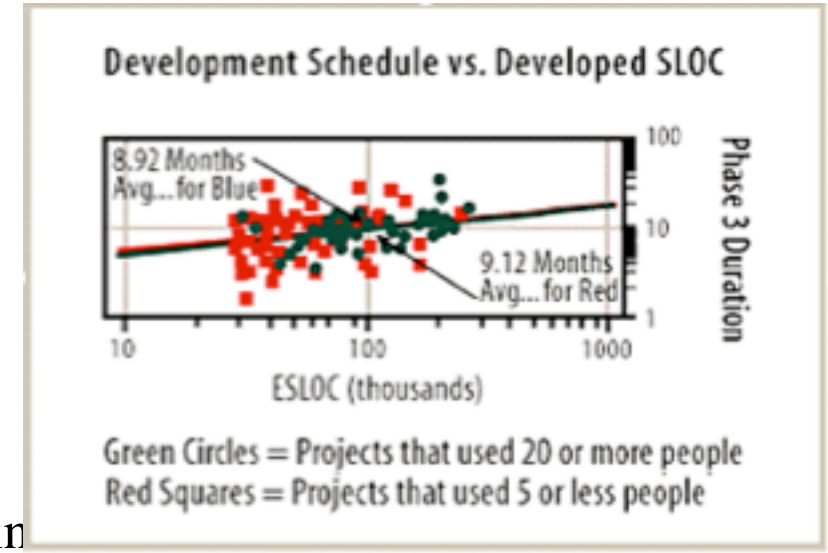
For projects of 100,000 SLOCs

- Peak staffing of project
  - Average large team: 32 people
  - Average small team: 4 people
- Total effort (person months) for project
  - 178 p.m for large teams ($2.1 M)
  - 24.5 p.m. for small teams ($0.3 M)

QSM Study, cont.

- Calendar time to complete project
  - 9.12 months for large team
  - 8.92 months for small team

The one week shaved
off delivery cost $1.8M



- Explanations?
  - Communication and coordination in
  - Greater rate of defects (5x)

Requirements

- Requirements come from humans
  - incomplete, ambiguous, imprecise
- If software development really is SDQ
  - then automate it!
- There is lots of automation
  - compilation, assembly, linkage
  - build, test

Not Good Enough

What good enough software promotes:

- Bloat and bugs
- Eye candy and features over usability
- Missing deadlines, delivering less
- Ignoring bugs that affect few people
- Acceptance of bugs as inevitable

What is engineering?

If it takes 2 people 4 days to dig a hole,

how long will it take 4 people?

- What's the rate limiting step?
  - pumping water out of the hole
- What are the resource limits?
  - backhoe, not people
- What are the scheduling constraints?
  - only certain workers available now
- Is the problem parallelizable?
  - no room in the hole for 4 people
- Will it help to finish earlier?
  - other steps might be delayed anyway
- Can everybody do all jobs?
  - backhoe operators require special training

Scientific Management

- The specialization of labor increased productivity in the mechanical world dramatically
- So why not in software?
  - Specialization means handoffs
  - Handoffs means documentation, errors, misunderstandings, time
  - New facts rock lots of boats

Software is Different

- Software as capital
  - Production costs are almost zero - everything is design
  - The design of the car, not the car itself
- Shared mental model
  - What you need to create software
- Teamwork
  - collaborative, social, intellectual, communicative work
  - minimum project size is 2 people