The software development process

A personal view

Robert Brady

Summary

Why bother?
Early (procedural) approach
Seminal research on complexity (IBM 1984)
The first revolution – IBM vs Microsoft
Sophisticated tools (objects etc.)
Agile and SCRUM (last 5 years)
Test-oriented development?

Why bother?

- "Recruit great developers"
  - They are 10-50 times more productive than average developers
  - who are 10-50 times more productive than poor developers
- "The process is secondary"

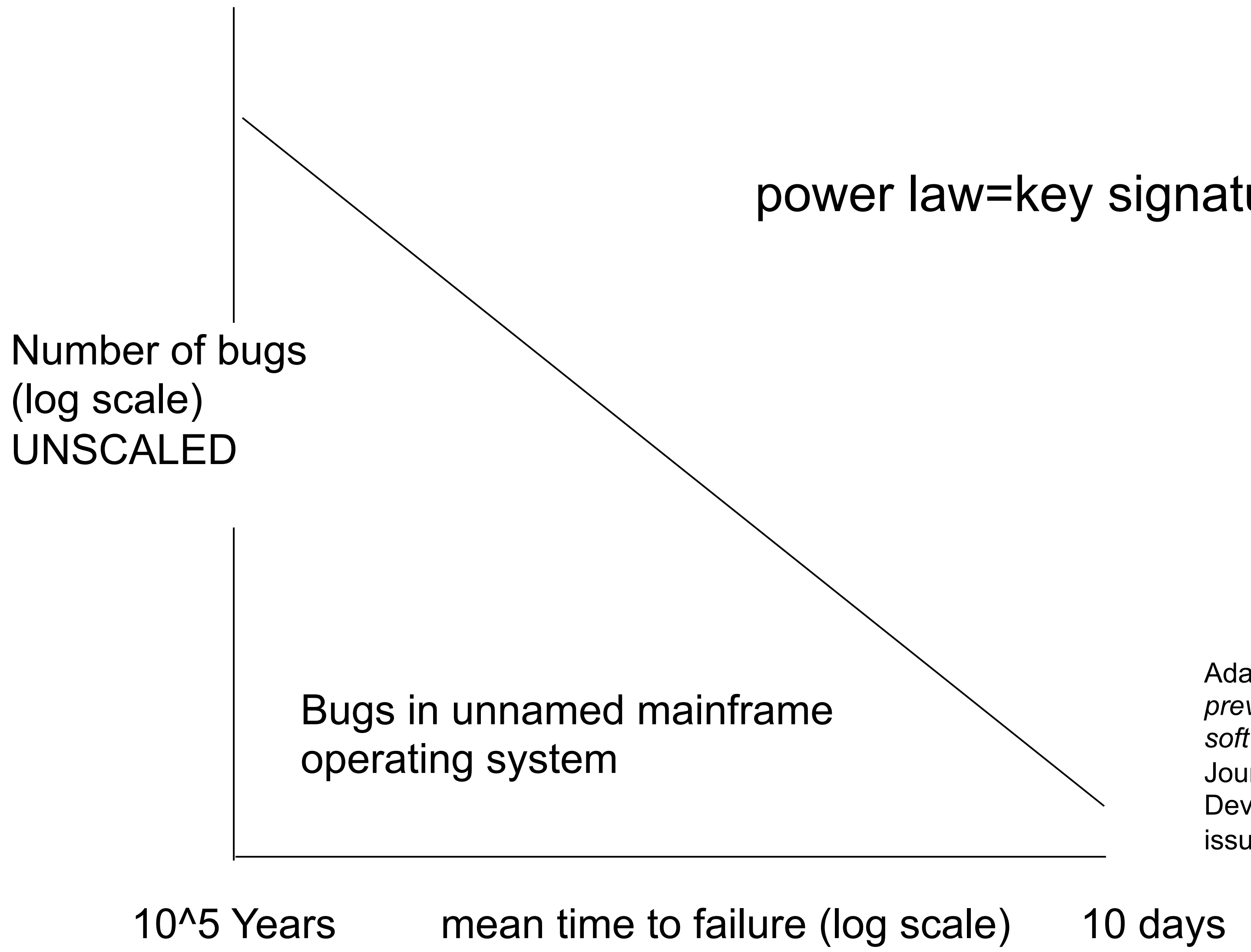
First revolution - IBM vs Microsoft

- According to “Big Blues: the Unmaking of IBM”:-
  - In the late 1980’s, IBM lost $70 billion of stock value
  - and gave an entire market away to a small company
  - Mainly because it couldn’t write software effectively.
- But IBM “did it right”. It followed all the standard rules taught in computer science courses at the time:
  - Get the design right before you write the code
  - Write complete documentation
  - Get it right first time
  - Use formal methods, design walk-throughs etc. to satisfy yourself that the code is bug-free, before release
  - Regard other methods (eg Microsoft’s) as “hacking”
- So what went wrong?

Size is important

0.1-1kb	Typical punch-card program The IBM development method was probably developed for this type of program
2kb-10kb	Typical software module/class Typical computer science project(?)
16kb	Operating system of Sinclair Spectrum
200kb	Our first software product – 1986
18 Mb	Human Genome – active code (30k genes * protein size 800) Number varies from year to year
20Mb	Our current software product (~20b per line)
750Mb	Human genome - including rubbish code (3 x 109 base-pairs)
4Gb	Windows Vista and associated products
218Gb	Storage on my laptop

IBM: seminal measurements 1984



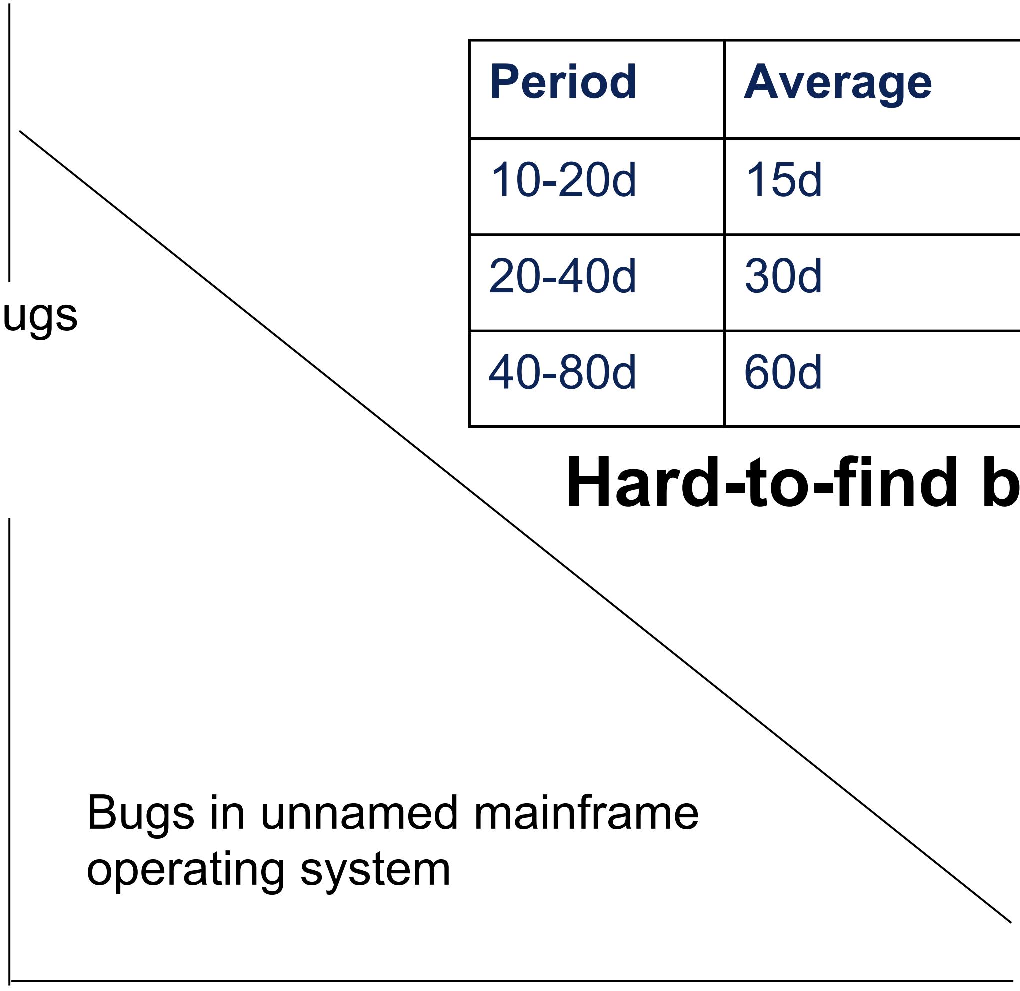
power law=key signature of complexity

Adams E. N., *Optimising preventive maintenance of software products*, IBM Journal of Research & Development, Vol. 28, issue 1 pp 2–14 (1984)

IBM: seminal measurements 1984

Period	Average	bugs	MTTF
10-20d	15d	1	15d
20-40d	30d	2	15d
40-80d	60d	4	15d

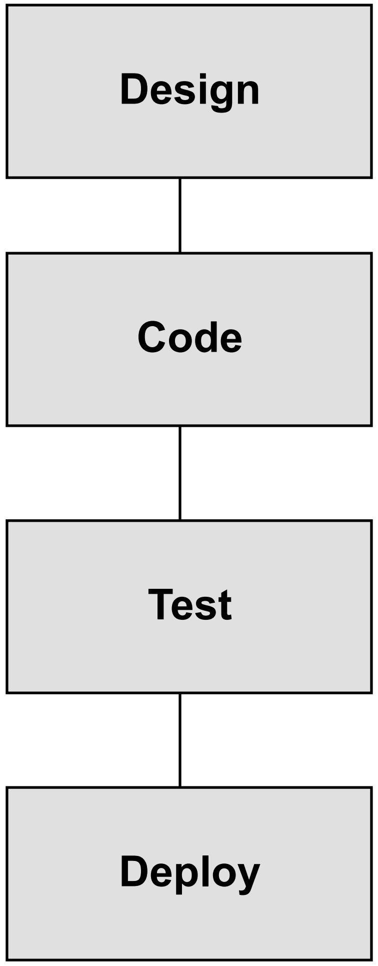
Hard-to-find bugs dominate



Adams E. N., *Optimising preventive maintenance of software products*, IBM Journal of Research & Development, Vol. 28, issue 1 pp 2–14 (1984)

Organising the code

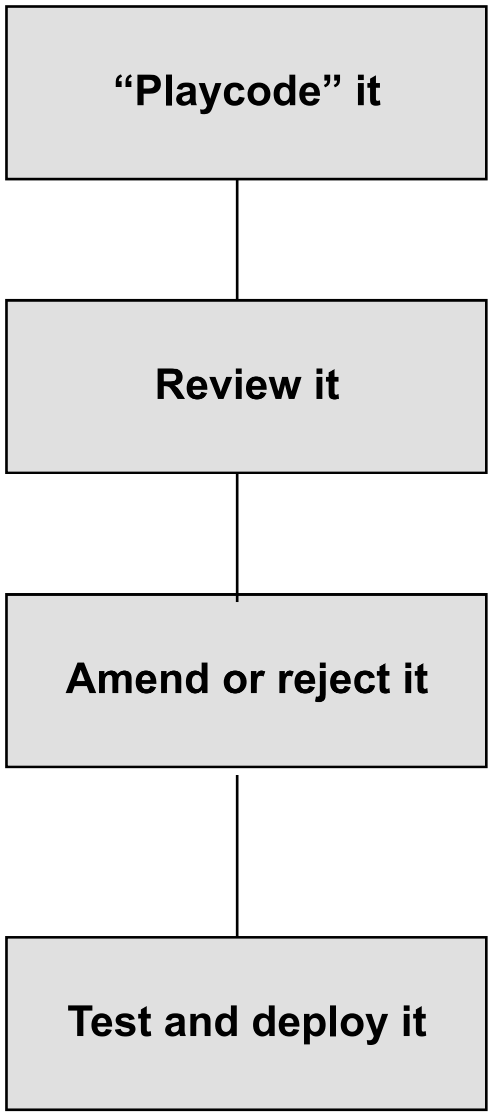
Waterfall model



Mainstay of development process
Good for small modules or sub-units, particularly if you can have simple and well-specified interface.
Be careful
- Different people for each stage = lost information = failure
- Microsoft at one stage: "We don't have programmers, we have developers"

Ideal process
- Sit with a user
- Agree small issues/problems
- Fix some yourself (nobody else)

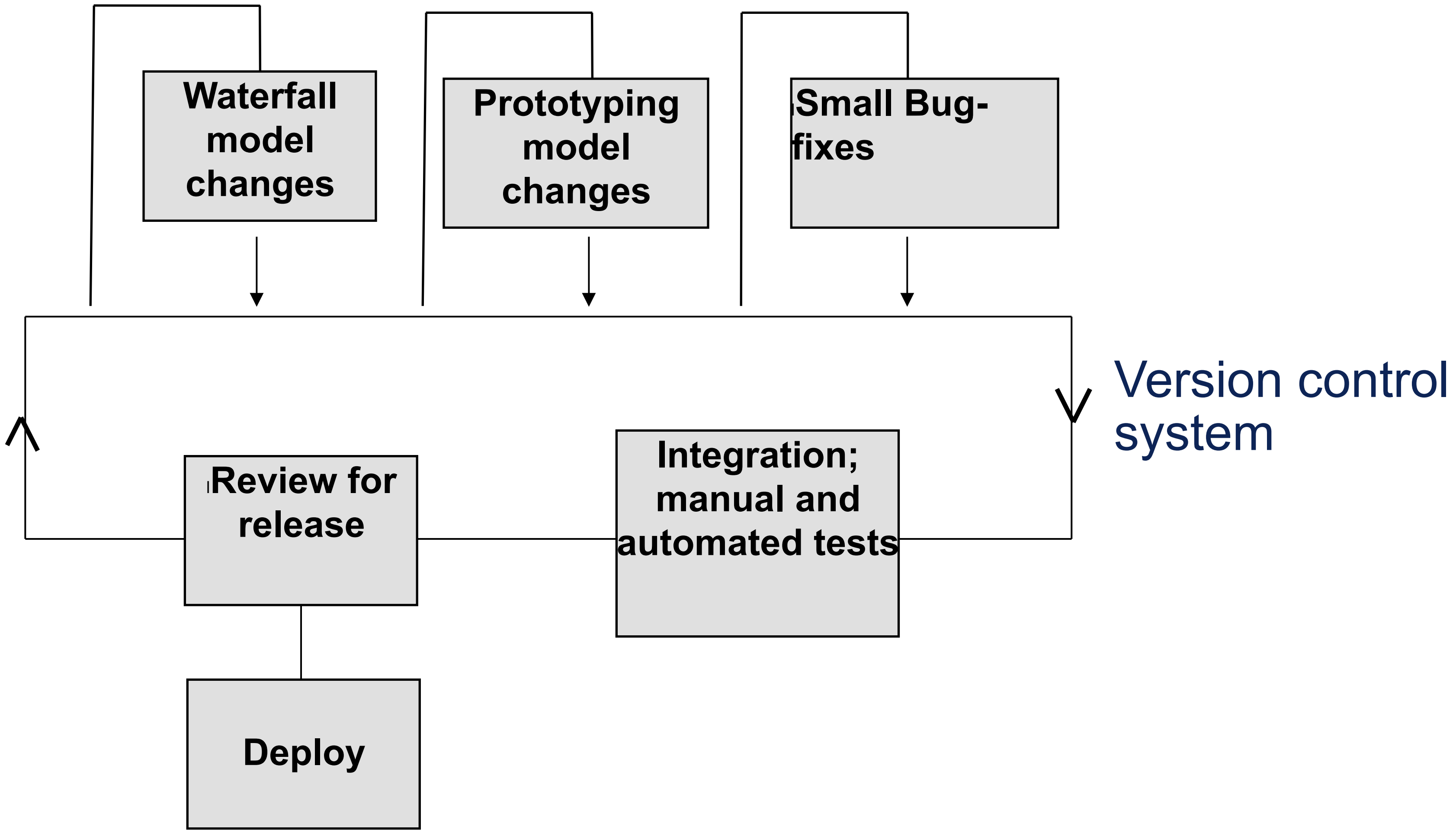
Prototyping



Good where there are significant project risks or unknowns
e.g. external software, new techniques or methods, or can't decide between alternatives

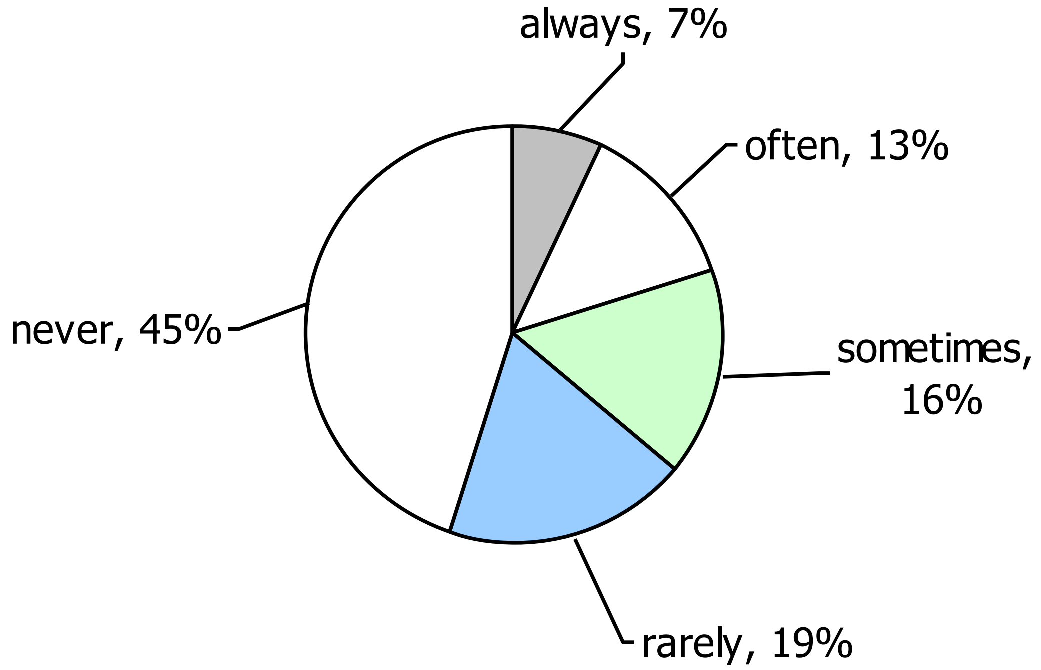
Not very predictable
a big problem in contracted developments

Evolutionary model



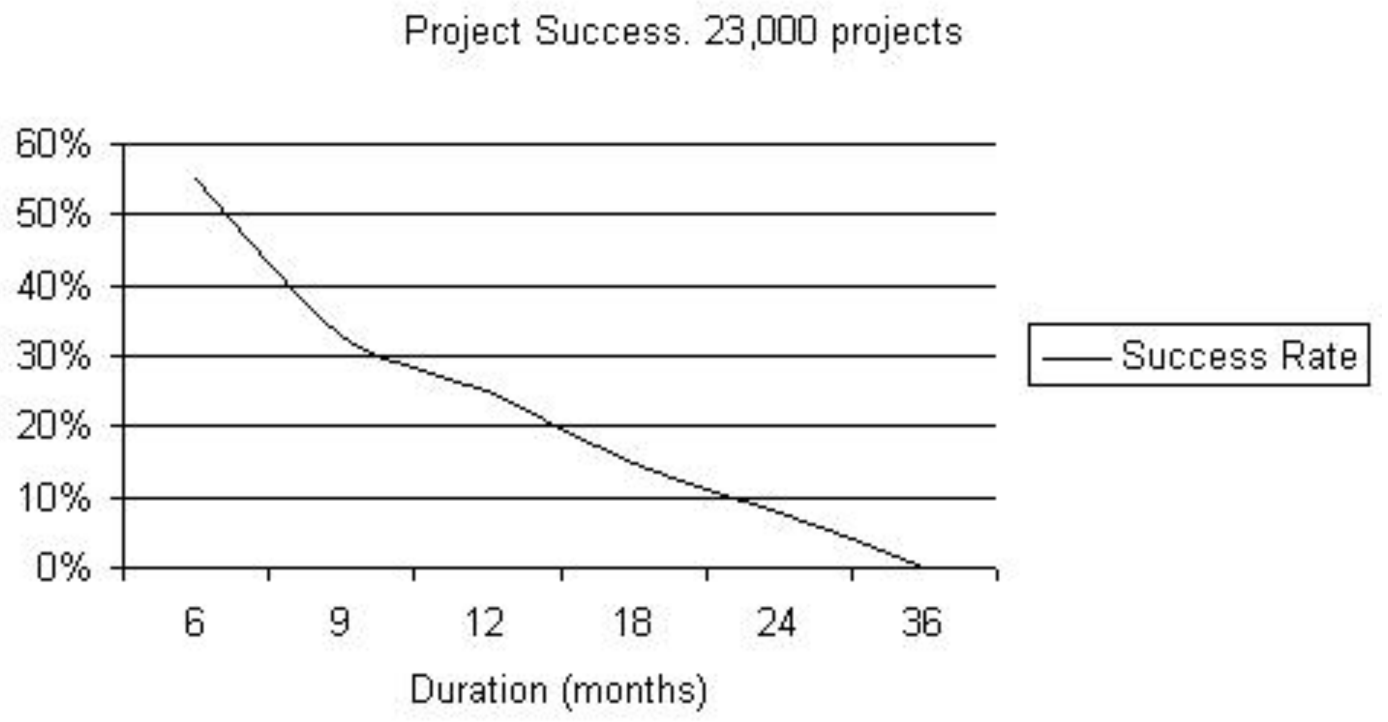
Organising the project

Problem with waterfall projects: 1. Unused features

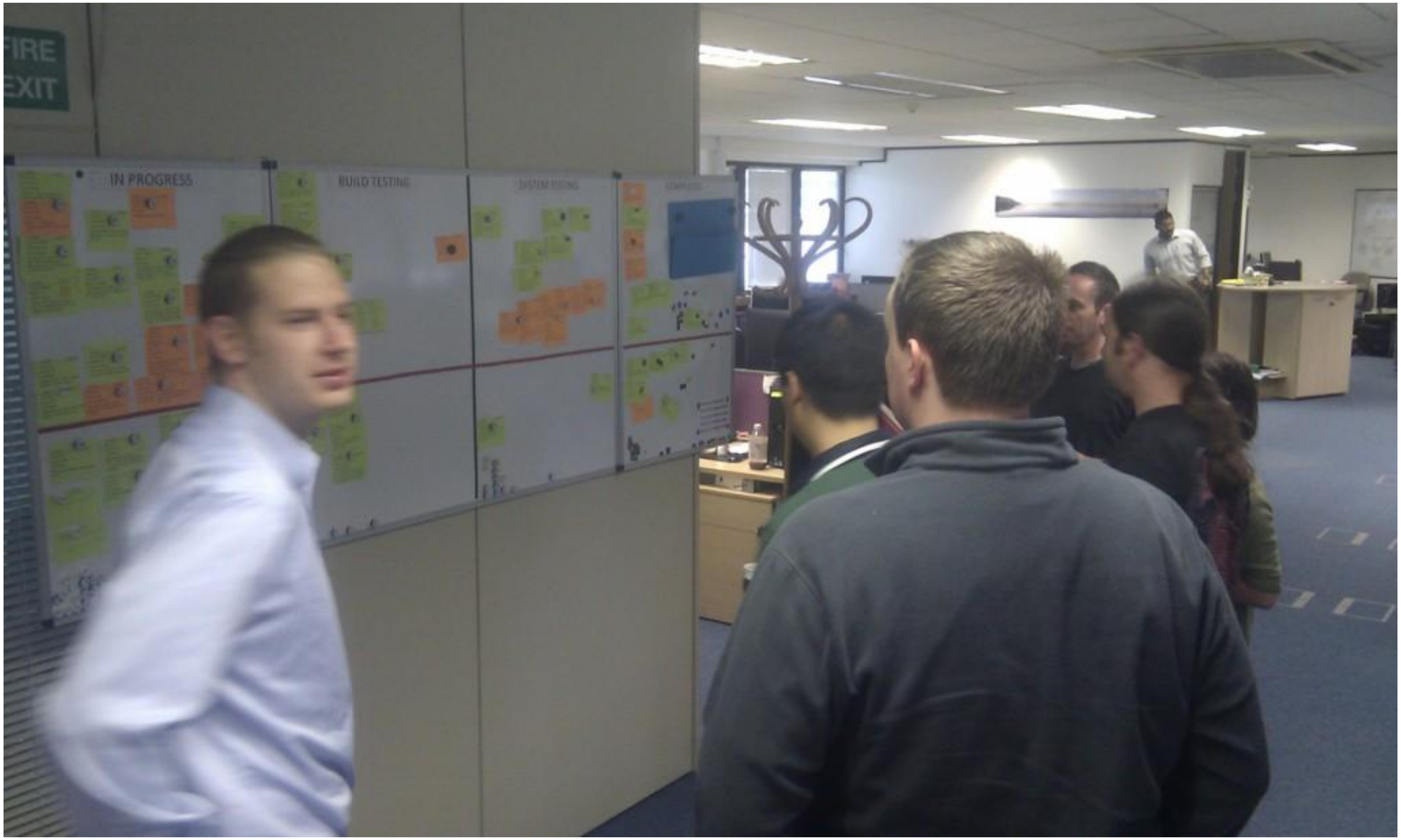


Actual use of requested features [Johnson02]

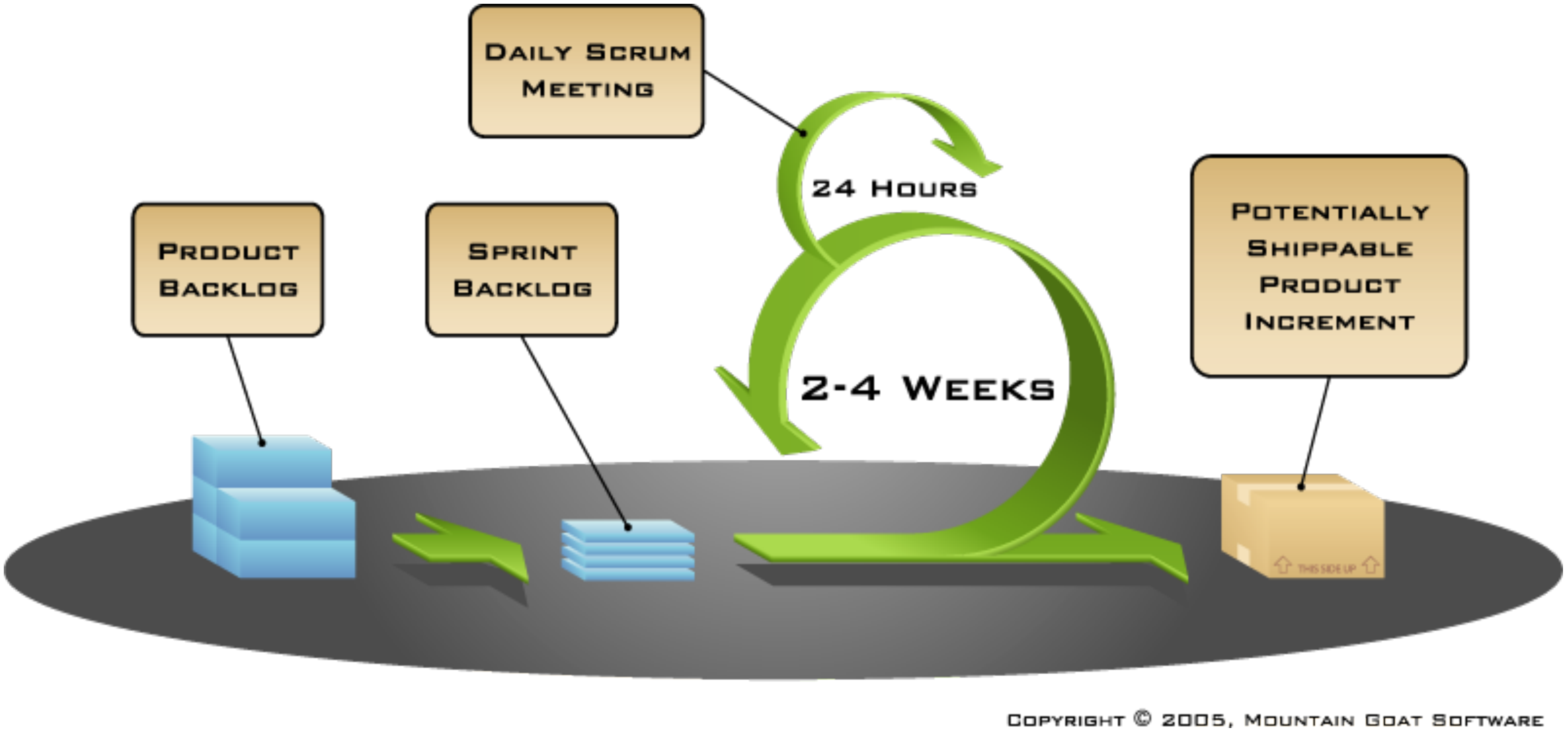
Problem with waterfall projects: 2. Project length/success profile



Project length vs. success [Johnson98]


FIRE
EXIT
IN PROGRESS
BUILD TESTING
SYSTEM TESTING

Scrum methodology


Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment
This side up
Copyright © 2005, Mountain Goat Software

Scrum framework

Roles

- Product owner
- ScrumMaster
- Team

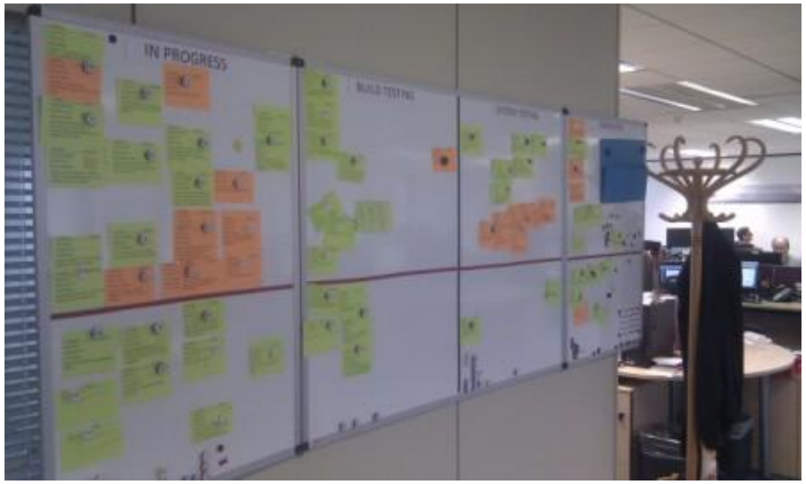
Ceremonies

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artifacts

- Product backlog
- Sprint backlog
- Burndown charts

The daily scrum



- Parameters
    - Daily
    - 15-minutes
    - Stand-up
- Not for problem solving
    - Whole world is invited
    - Only team members, ScrumMaster, product owner, can talk
- Helps avoid other unnecessary meetings

Test-oriented development

Experimental (for us)

Requirements – design – develop – test
miscommunication throughout the chain
testing at the end so it suffers most

Requirements – design – test – develop
test engineer is part of the development team
Tests run automatically with each daily build

Summary

Why bother?
Early (procedural) approach
Seminal research on complexity (IBM 1984)
The first revolution – IBM vs Microsoft
Sophisticated tools (objects etc.)
Agile and SCRUM (last 5 years)
Test-oriented development?