BRADY

COMMODITY SOFTWARE SOLUTIONS

The software development process

A personal view

Dr Robert Brady

Founder and non-executive director, Brady plc

References

- “Debugging the Development Process”
  - Steve Maguire, Microsoft Press
  - Real disaster stories
  - Inspirational and rational solutions
  - Recommended template, unashamedly plagiarised here
- Big Blues: The Unmaking of IBM
  - Paul Carroll, Three Rivers Press
  - Journalist on how IBM couldn’t write software (and Microsoft worked out how to – sort of – at critical time of s/w change)
- “Showstopper!”
  - G Pascal Zachary, Macmillan
  - Fly on the wall for the first release of Windows NT
- Google Chrome development cartoon
  - http://www.google.com/googlebooks/chrome/index.html

Agenda

- The three most important things
  1. Bugs
  2. Bugs
  3. Bugs
- Why is software development hard?
- Managing the code
- Development team ground-rules
- Making the management decision to ship

BRADY

BRADY

COMMODITY SOFTWARE SOLUTIONS

Why is software development hard?

History

- 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?

BRADY

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
- 200Mb — Our current software product
- 750Mb — Human genome - including rubbish code
  (3×10^9 base-pairs)
- 4Gb — Windows Vista and associated products
- 218Gb — Storage on my laptop

BRADY

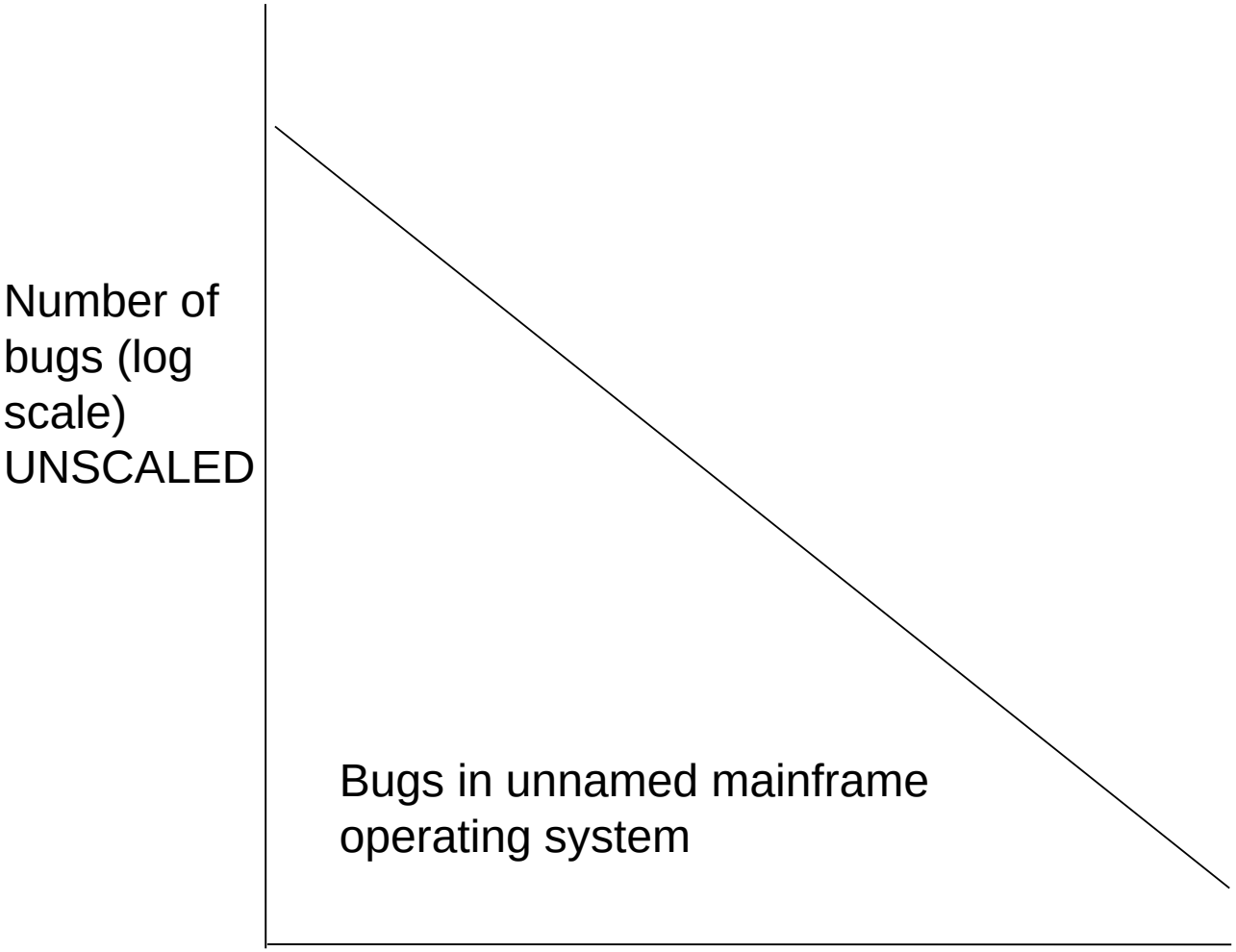
Software development process

How size affects the basic assumptions

	Punch-card program	2kb of code	Large program
Complete the design in advance	Almost essential	Difficult	Too complex - not possible
Complete the documentation in advance	Highly desirable	Difficult	Too complex - not possible
Prove it is bug-free	Very difficult mathematical challenge	Too complex - not possible	Too complex - not possible
“Right first time”	A worthy goal	Too complex - not possible	Too complex - not possible

BRADY

IBM: Seminal measurements 1984

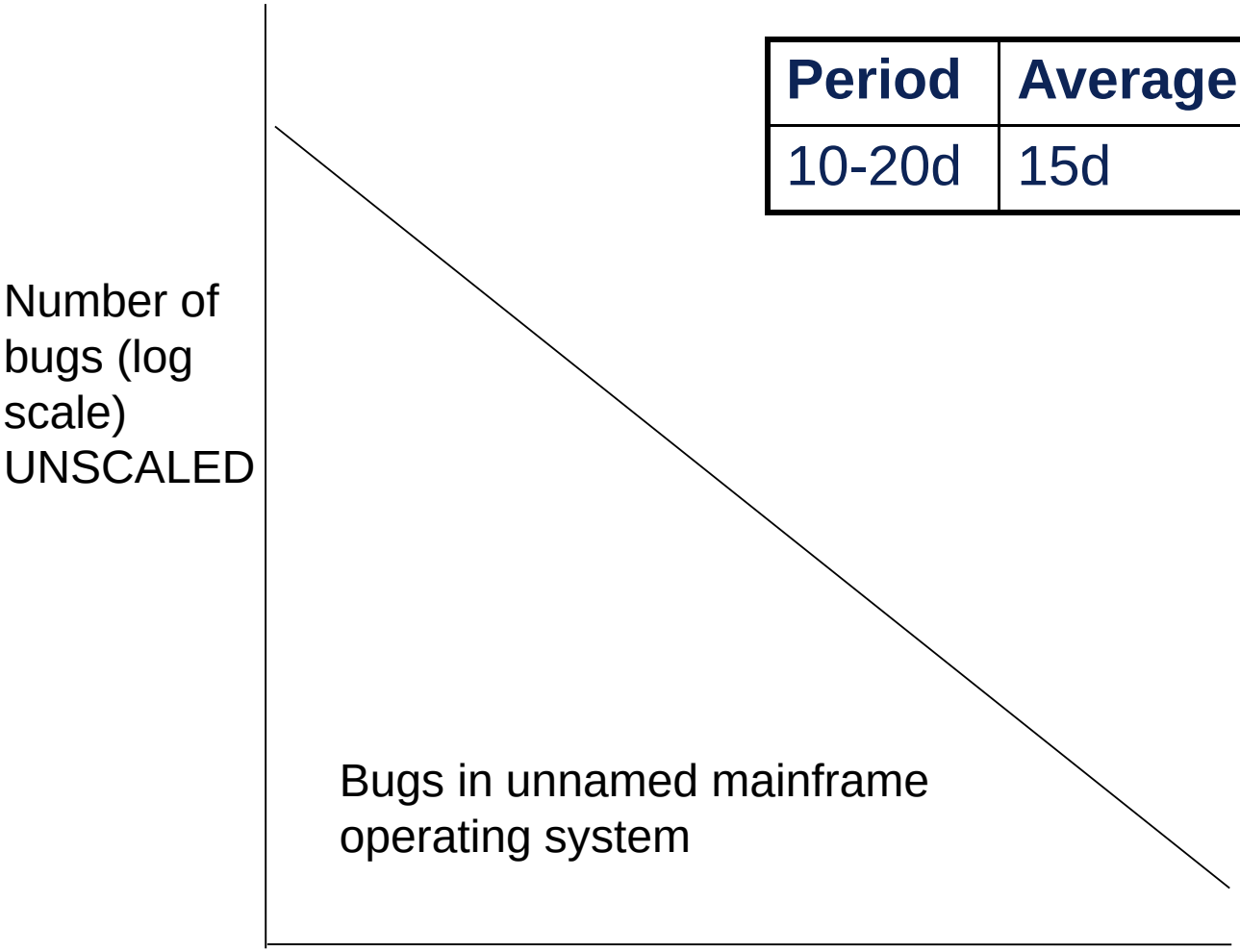


Adams E. N., *Optimising preventive maintenance of software products*, IBM Journal of Research & Development, Vol. 28, issue 1 pp 2–14 (1984)

BRADY

IBM: Seminal measurements 1984

Period	Average	bugs	MTTF
10-20d	15d	1	15d

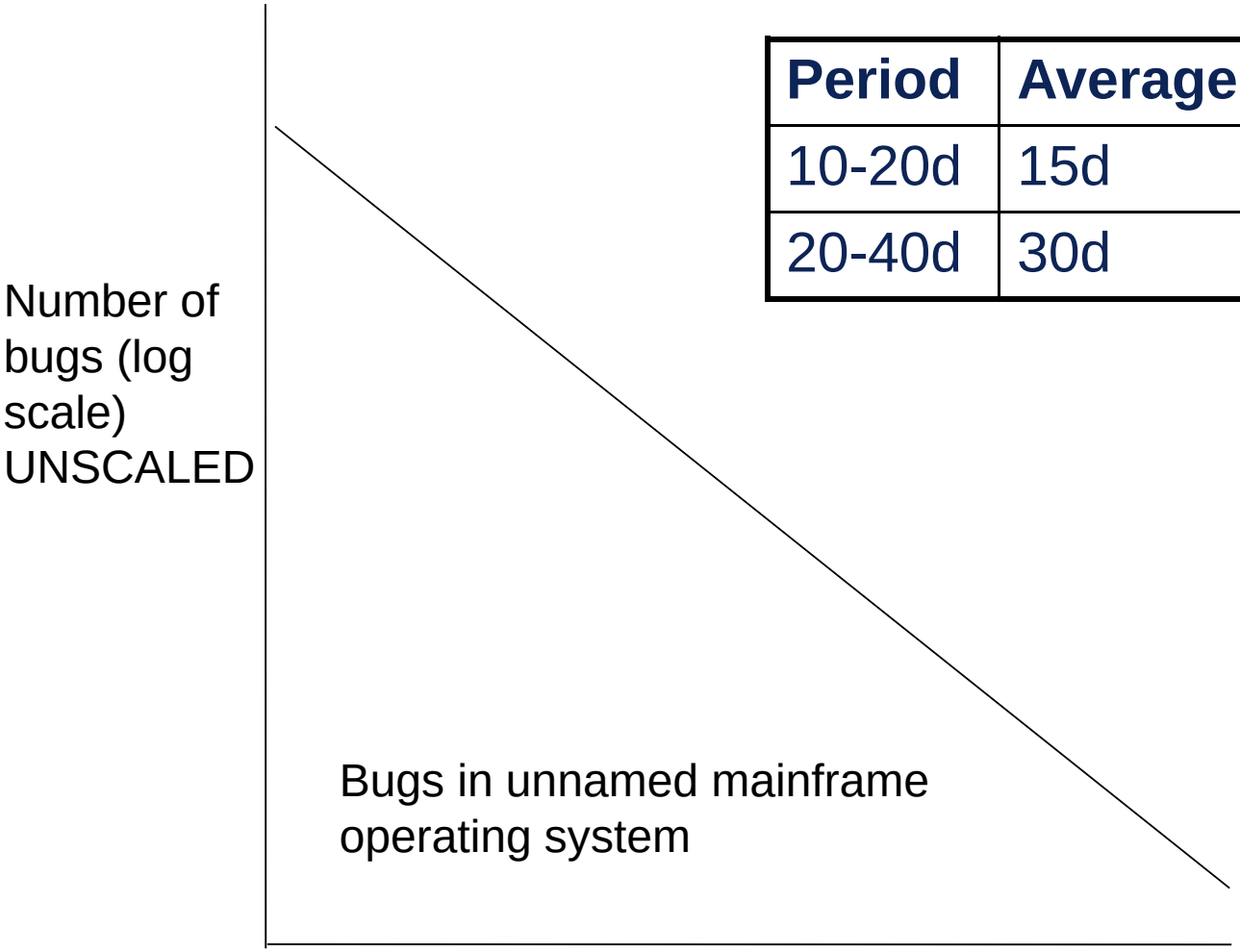


Adams E. N., Optimising preventive maintenance of software products, IBM Journal of Research & Development, Vol. 28, issue 1 pp 2–14 (1984)

BRADY

IBM: Seminal measurements 1984

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

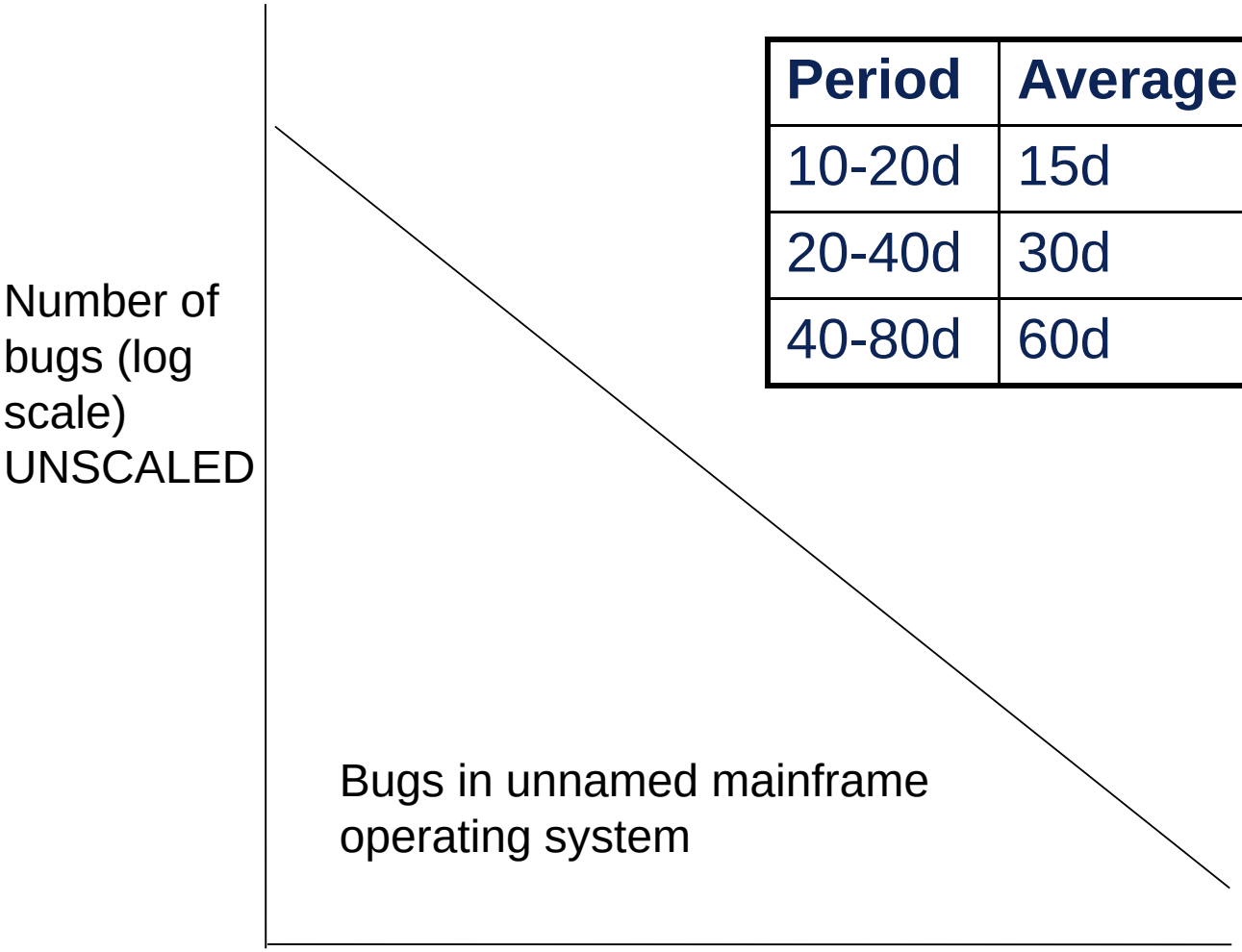


Adams E. N., Optimising preventive maintenance of software products, IBM Journal of Research & Development, Vol. 28, issue 1 pp 2–14 (1984)

BRADY

IBM: Seminal measurements 1984

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

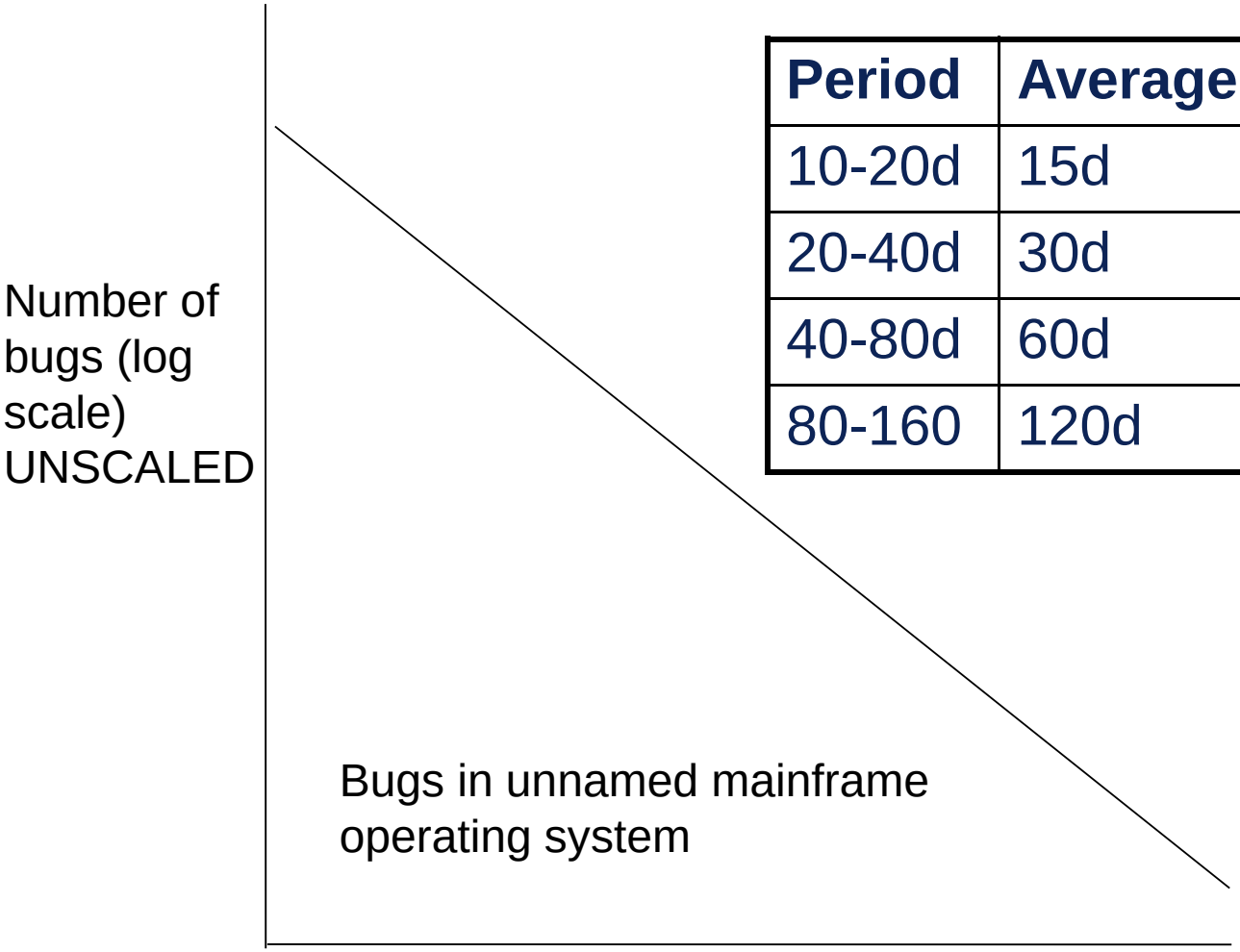


Adams E. N., Optimising preventive maintenance of software products, IBM Journal of Research & Development, Vol. 28, issue 1 pp 2–14 (1984)

BRADY

IBM: Seminal measurements 1984

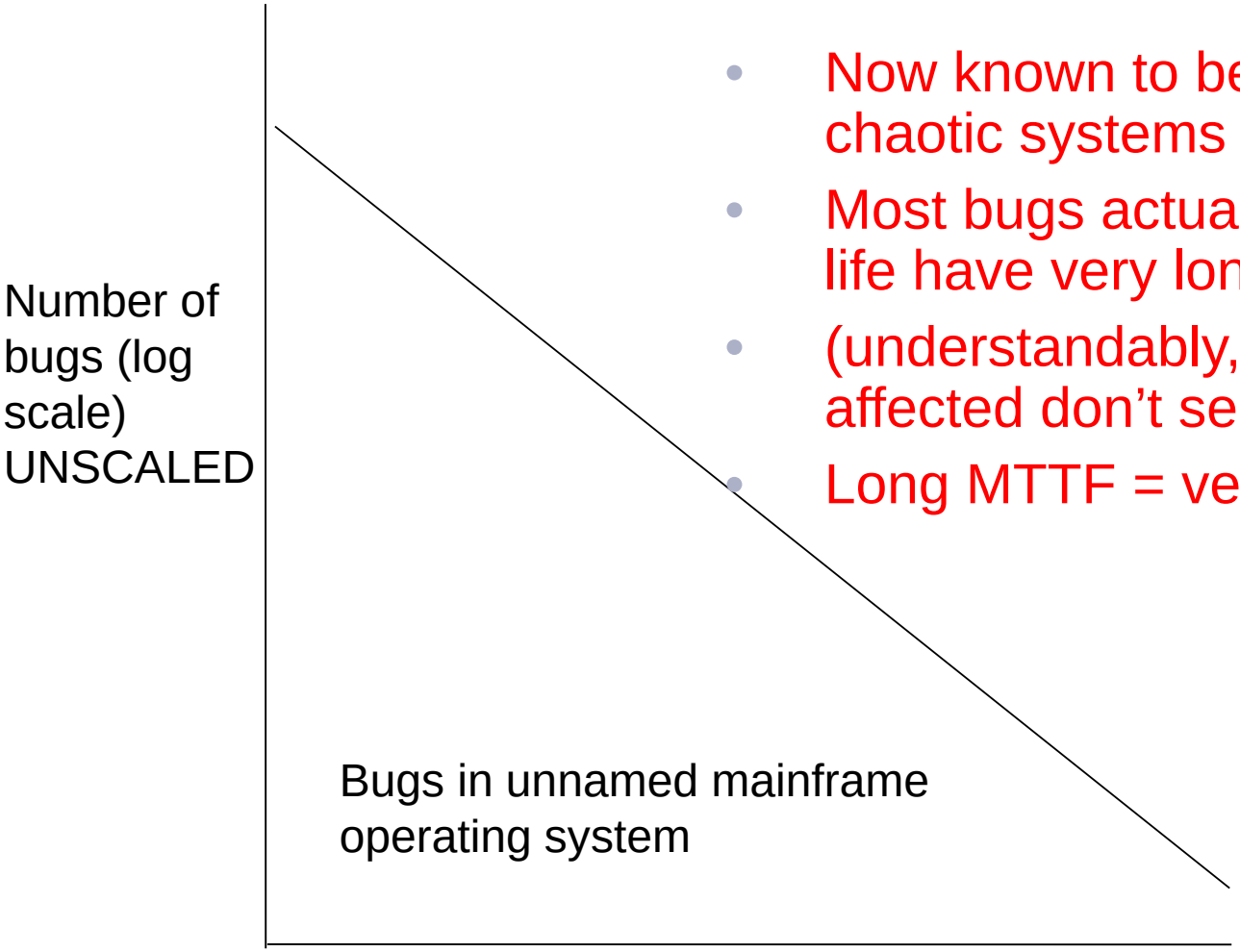
Period	Average	bugs	MTTF
10-20d	15d	1	15d
20-40d	30d	2	15d
40-80d	60d	4	15d
80-160	120d	8	15d



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

BRADY

IBM: Seminal measurements 1984



- Now known to be a key signature of chaotic systems
- Most bugs actually triggered in real life have very long MTTF
- (understandably, customers who are affected don't see it that way)
- Long MTTF = very hard to find

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

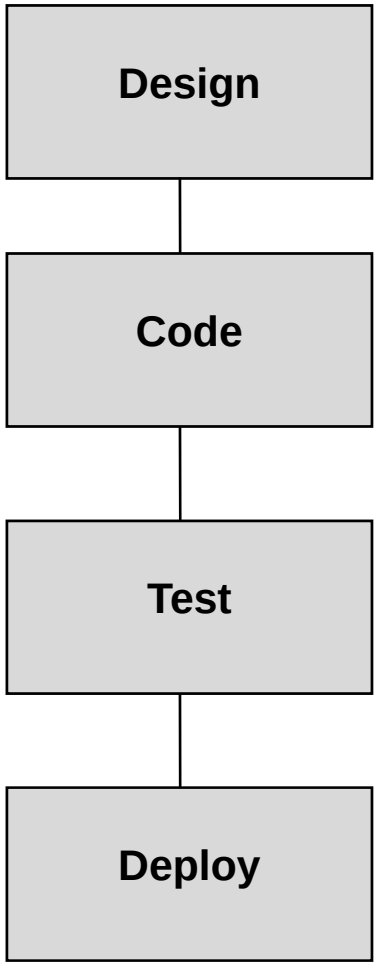
BRADY

BRADY

COMMODITY SOFTWARE SOLUTIONS

Managing the code

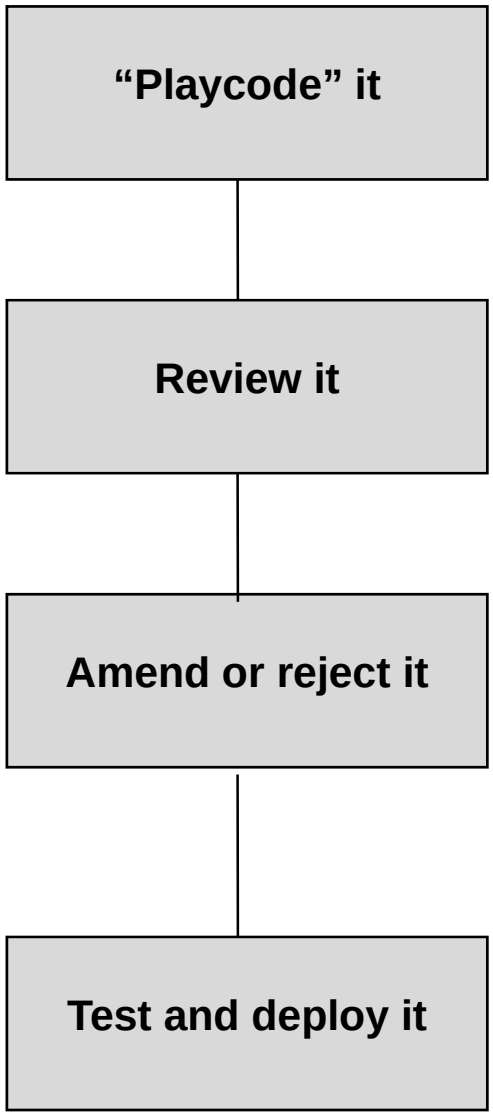
Waterfall Model



- 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”

BRADY

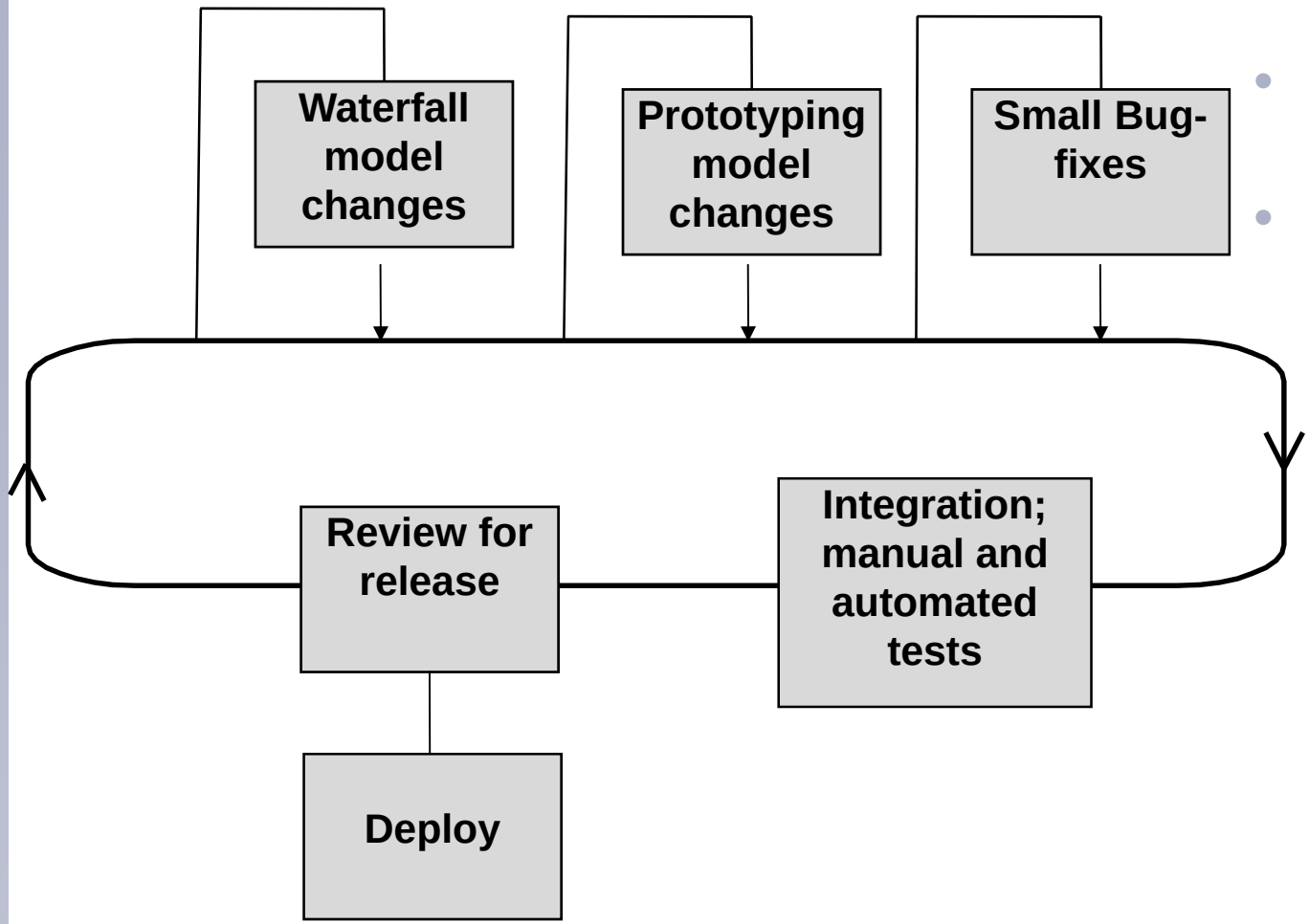
Prototyping model



- 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

BRADY

Evolutionary model



- Version control system
- Everyone does this in practice
- Manages complex interactions between developments

BRADY

BRADY

COMMODITY SOFTWARE SOLUTIONS

Development team ground-rules

An interactive quiz

Please don’t peek ahead!

Your starter for 10

- You are the manager of a small (2 person) software development/test team
  - David the Developer
  - Terry the Test Engineer
- They come to you with a **problem** and a proposed **solution**.
- Do you approve it?

BRADY

Your starter for 10

- **Problem**
  - We need to implement 10 features. We have reviewed the designs, we now need to code and test them.
  - Time is very tight. We will have to pull out all the stops to do it by the contracted deadline of next month
  - David is the best person to do development
  - Terry is the best person to do the testing
- **Proposed solution**
  - David and Terry work closely together to accelerate the development phase
  - David develops the features and makes quick releases to Terry during development
  - Terry provides testing feedback during this phase
  - After this development phase, the software will go into the normal release cycle for testing/bugfix
- **Do you approve?**

BRADY

If you approve the plan

- You will send a message to your developers that **bugs don't matter** – you can "throw them over the wall" and someone else will find them for you
- You will accelerate developers who produce sloppy code and slow down developers who produce good code
- The process will be inefficient, eg
  - David will rely on Terry, so won't run 'white box' tests
  - Silly bugs will stop Terry running his automated tests
  - Constant communication will slow the team down
- When you get to the original deadline
  - your project will have all the features, but too many bugs
  - You won't be able to advise the customer of the new ship date, because the automated tests don't work - and they always uncover something new when they do run
  - It will be too late to take corrective action

BRADY

If you reject the plan

“Developer has to test his code before release”

- The team will be forced to make the hard project decisions, eg
  - Go back to the design stage for feature number 3; can we implement it more simply?
  - Cut feature number 6 – it’s not strictly in the specification
  - Advise the customer there is a risk. Does he want a delay, or does he want feature number 7 in a later release?
  - Request more resources (a long shot…)
- Your team will work more efficiently
  - Terry will always work on code that is basically stable (so he can develop his regression tests etc.)
  - David will be rewarded for producing quality code, not for producing features that destabilise the product

BRADY

If you reject the plan

Your team will be better able to plan the project

- If a feature is in the product then it will “basically work”
- The team (and you) can now monitor progress
- You can get test results and customer feedback early on the features you have implemented
- Management can make the decision to ship with a more predictable freeze-time

BRADY

My Damascus moment

- Conventional way to scale development team
  - Tools, methodologies, targets
  - ‘Typing pool’ model
  - Not conducive to commitment, ownership, joy
- Feature teams – eg SCRUM
  - Small teams of 3 -6
  - Everyone fully involved & responsible for all decisions in functional area
  - Post-it notes to advertise work planned, in progress, ready, tested
  - WORKS WITH OUTSOURCING TO RUSSIA

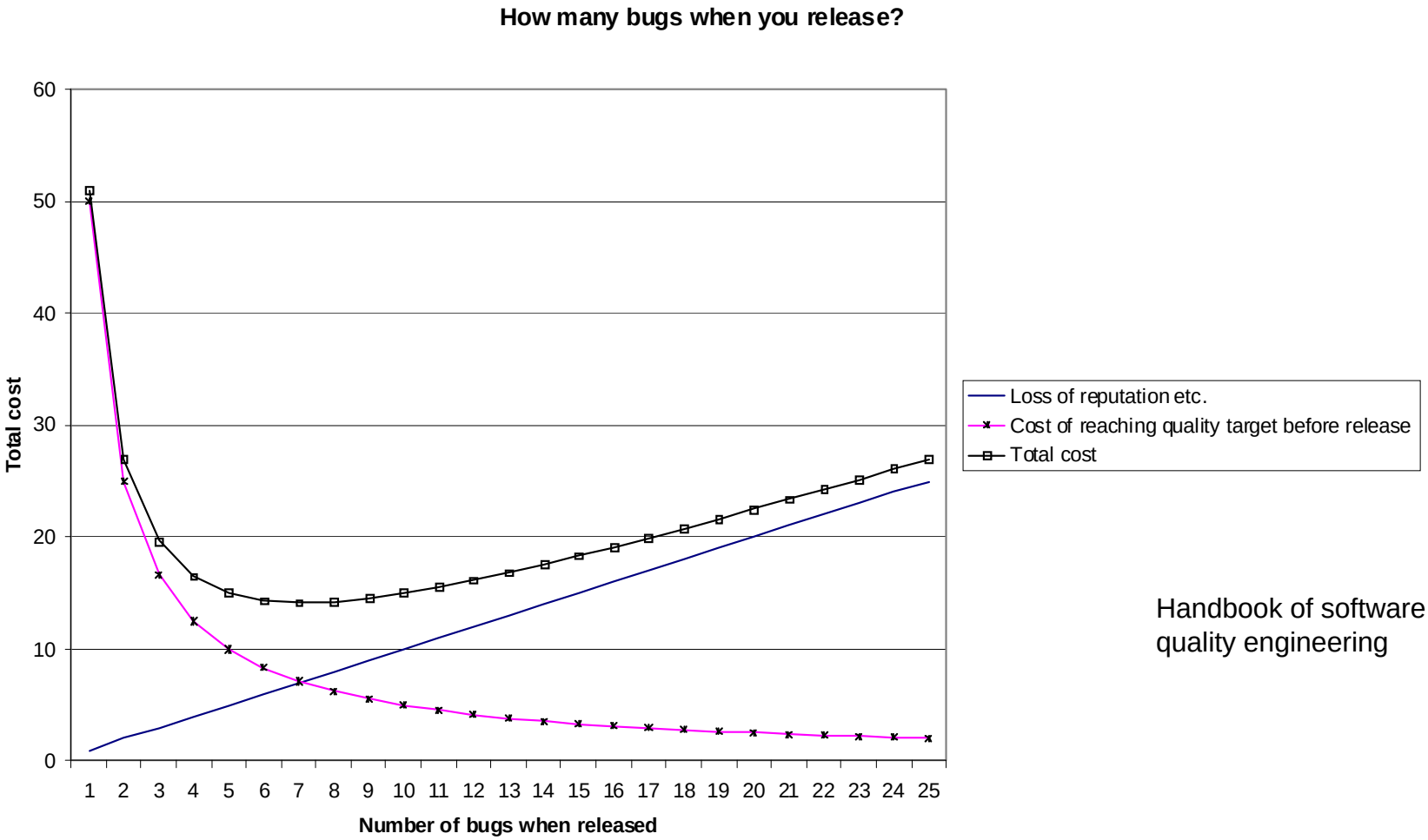
BRADY

BRADY

COMMODITY SOFTWARE SOLUTIONS

Making the management decision to ship

Recognising the trade-off



Handbook of software quality engineering

BRADY

Metrics: NT version 1

- **5.6M lines of code**
  - **1 bug for every 100 lines = 56K bugs to fix**
  - **1 bug for every 10 lines = 560K bugs to fix**
  - **1 bug per line (some academic industry estimates) = 5.6M bugs to fix**
- **Management's major activity: prioritising bugs**
  - **Showstopper (always fixed)**
  - **Priority 1 (fixed except in late stages of release)**
  - **Priority 2 (deferred)**

Date	Release	Serious or showstopper bugs
12 Oct 1992	Beta 1	2,000 known on release
8 Mar 1993	Beta 2	0 known on release 263 found in first 6 days
26 Jul 1993	Final	0 showstoppers known on release

BRADY

My personal view

- Avoid the “first release” syndrome altogether if poss
  - Make the first release very small
  - Make regular upgrades (SP or beta versions)
  - Typically monthly or quarterly releases
  - Each release contains only small changes
- Essential to limit risks OF EACH RELEASE
  - Invest in automated regression tests
  - The risk of each change is the primary focus
  - Manage higher risk changes by breaking them up
    - eg Have the ability to switch the risky part on/off
    - eg implement a big change in smaller bits
- Get real customers for each release
  - Forces focus on what the customer really requires
  - Gets real-world feedback that no lab can reproduce
- Problems? Add to automated regression tests

BRADY

Summary

- The three most important things
  1. Bugs
  2. Bugs
  3. Bugs
- Why is software development hard?
- Managing the code
- Development team ground-rules
- Making the management decision to ship

BRADY

Selling your company

- Typical acquisition size for us
  - 10 years, 15 developers
  - £2M-£12M
- My key question in due diligence (rough and ready indicator): How many lines of code?
  - 500,000 = functionality
  - 3M = duplication nightmare
  - Moral: Reduce code size in the first place
  - Design for code re-use
  - Refuse to code non-general features in the product

BRADY

BRADY

COMMODITY SOFTWARE SOLUTIONS

The software development process

Thank you

www.bradyplc.com for summer internships