The software development process
A personal view

Robert Brady

robert.brady@cl.cam.ac.uk

28 October 2014

Software development

Why any management at all?

‘Just recruit great developers’

- They are 10-50 times more productive than average developers
- who are 10-50 times more productive than poor developers
- Management will just get in the way

1980's – the Hacking revolution

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 called Microsoft
- Mainly because it couldn't write software effectively.

1980's – the Hacking revolution

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 called Microsoft
- Mainly because it couldn't write software effectively.

But IBM 'did it right'

It followed all the standard rules (later taught in computer science courses)

- 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"

1980's – the Hacking revolution

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 called Microsoft
- Mainly because it couldn't write software effectively.

But IBM 'did it right'

It followed all the standard rules (later taught in computer science courses)

- 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?

IBM method based on fundamental misconception

Complete design up front ...

0.1-1kb	Typical punch-card program
	The IBM development method was
	probably developed for this type of program

IBM method based on fundamental misconception

Complete design up front ...

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

IBM method based on fundamental misconception

Complete design up front ...

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 – 1982

IBM method based on fundamental misconception

Complete design up front ...

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 – 1982
200kb	Our first software product – 1986

IBM method based on fundamental misconception

Complete design up front ...

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 – 1982
200kb	Our first software product – 1986
18 Mb	Human Genome – active proteins

IBM method based on fundamental misconception

Complete design up front ...

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 – 1982
200kb	Our first software product – 1986
18 Mb	Human Genome – active proteins
100Mb	Typical software product today

IBM method based on fundamental misconception

Complete design up front ...

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 – 1982
200kb	Our first software product – 1986
18 Mb	Human Genome – active proteins
100Mb	Typical software product today
750Mb	Human genome (3×10^9 base-pairs)

IBM method based on fundamental misconception

Complete design up front ...

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 – 1982
200kb	Our first software product – 1986
18 Mb	Human Genome – active proteins
100Mb	Typical software product today
750Mb	Human genome (3×10^9 base-pairs)
4Gb	Windows Vista

IBM method based on fundamental misconception

Complete design up front ...

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 – 1982
200kb	Our first software product – 1986
18 Mb	Human Genome – active proteins
100Mb	Typical software product today
750Mb	Human genome (3×10^9 base-pairs)
4Gb	Windows Vista
200Gb	Storage on my laptop

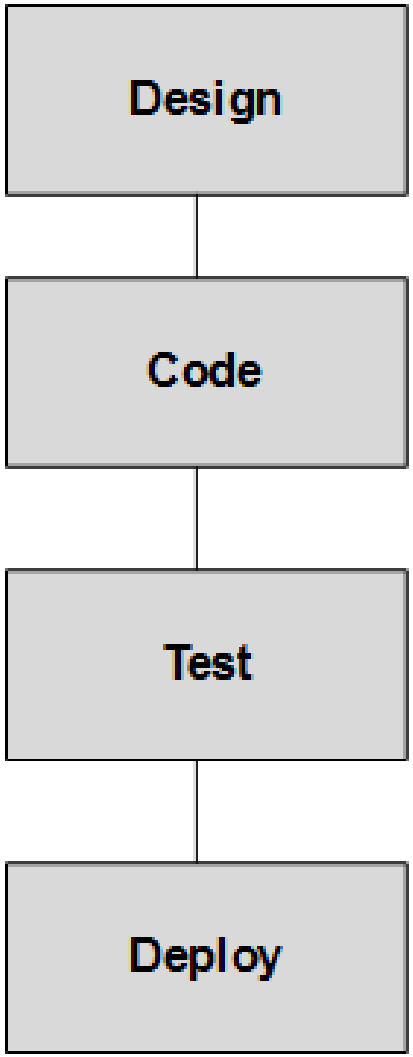
IBM method based on fundamental misconception

Complete design up front ...

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 – 1982
200kb	Our first software product – 1986
18 Mb	Human Genome – active proteins
100Mb	Typical software product today
750Mb	Human genome (3×10^9 base-pairs)
4Gb	Windows Vista
200Gb	Storage on my laptop
?Tb	Google code

Software development

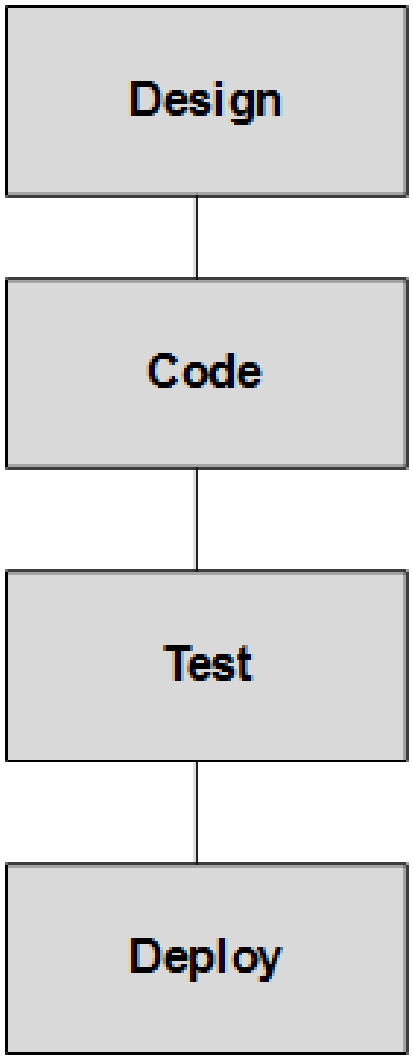
Waterfall model



Mainstay of development process

Good for small modules or sub-units, particularly if you can have simple and well-specified interface.

Waterfall model



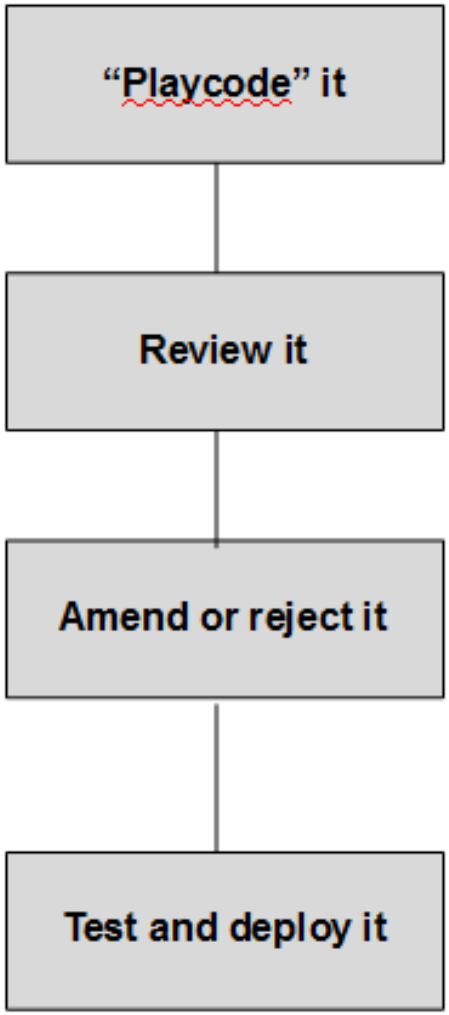
Mainstay of development process

Good for small modules or sub-units, particularly if you can have simple and well-specified interface.

Danger

- Different people for each stage
- Miscommunication

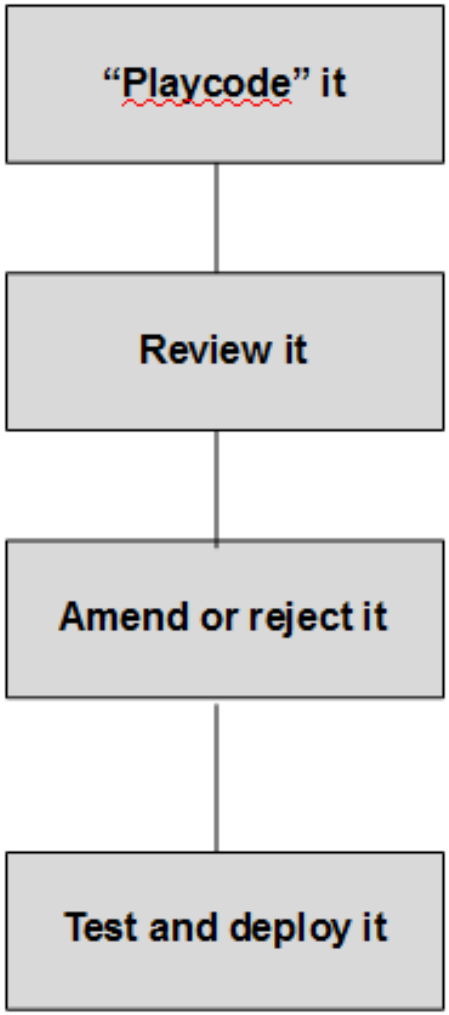
Prototyping



Good where there are significant project risks or unknowns

- external software
- new techniques or methods
- can’t decide between alternatives

Prototyping



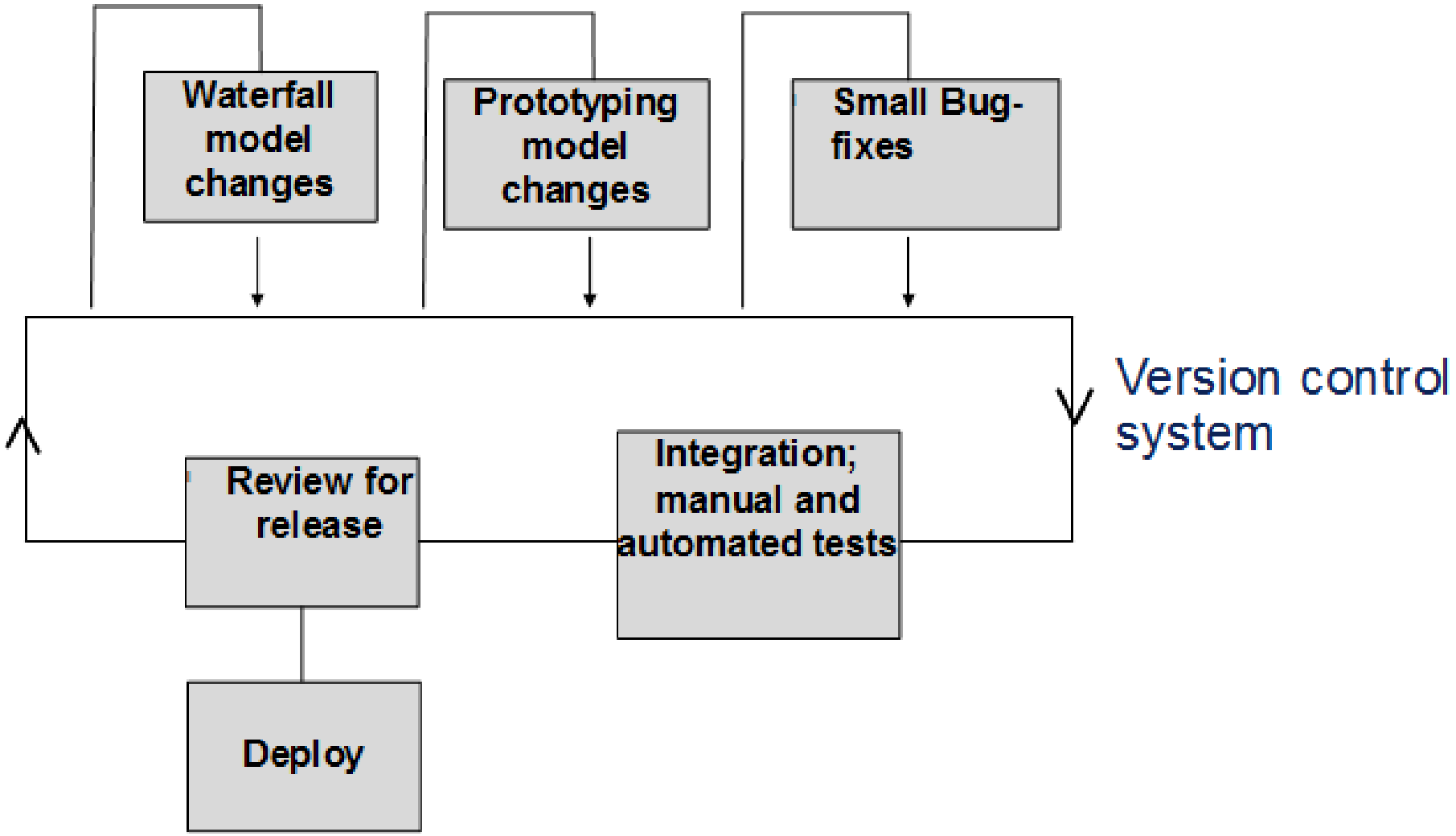
Good where there are significant project risks or unknowns

- external software
- new techniques or methods
- can’t decide between alternatives

Not very predictable

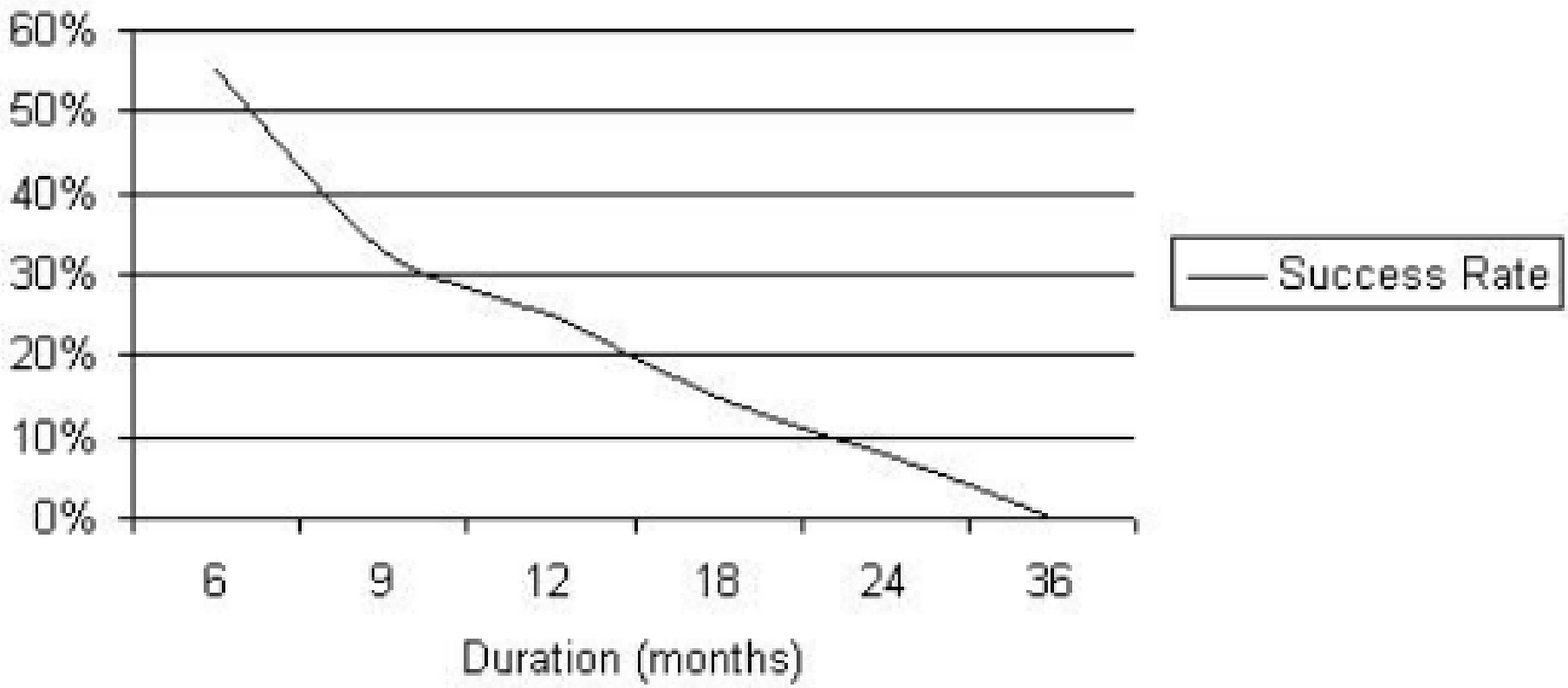
- a big problem in contracted developments

Evolution


Waterfall model changes
Prototyping model changes
Small Bug-fixes
Version control system
Review for release
Integration; manual and automated tests
Deploy

How well did it do? (1)

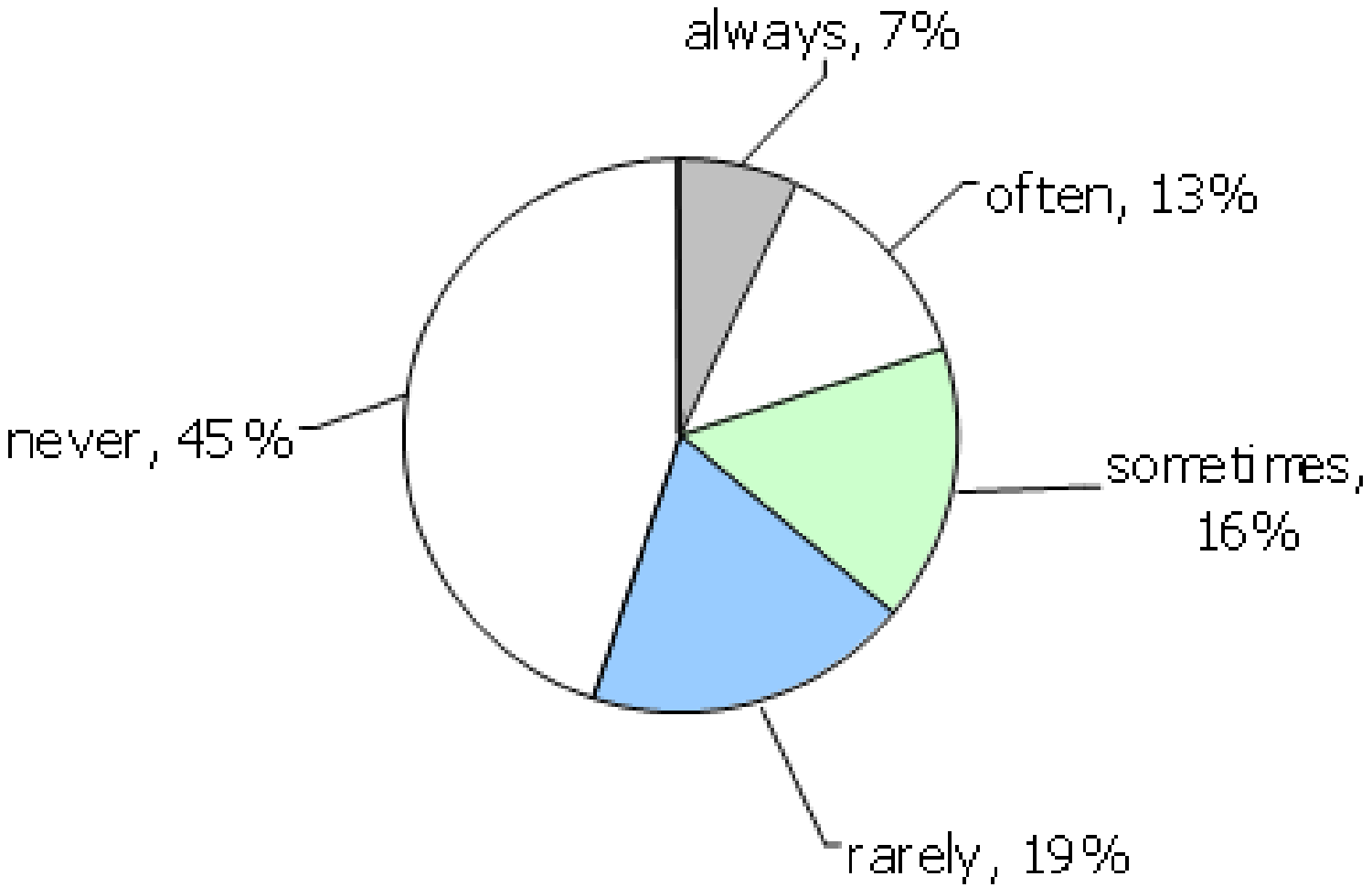
Success rate of projects (Johnson 1998)



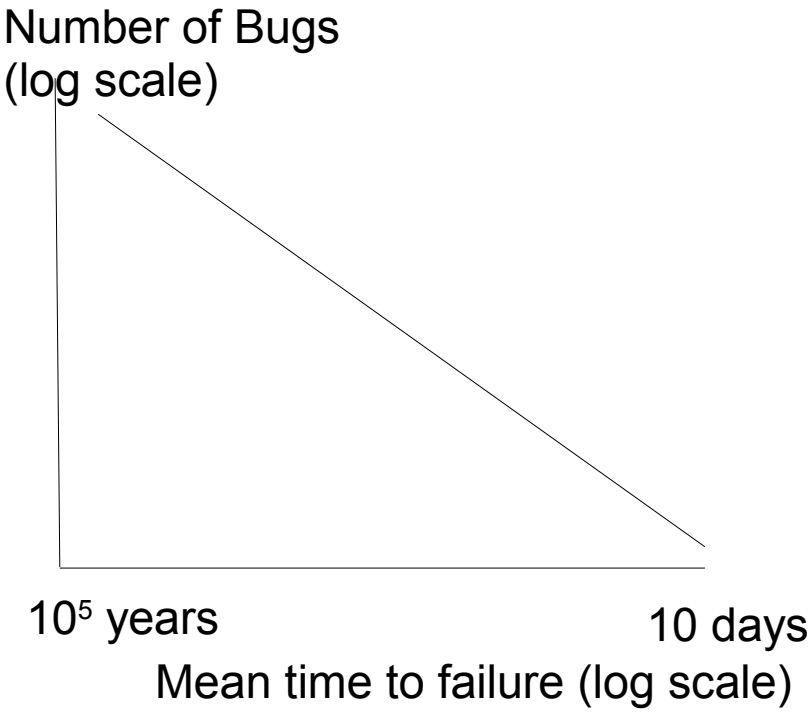
Break up large projects up into shorter ones (weeks not months)

How well did it do? (2)

Actual use of requested features (Johnson 2002)

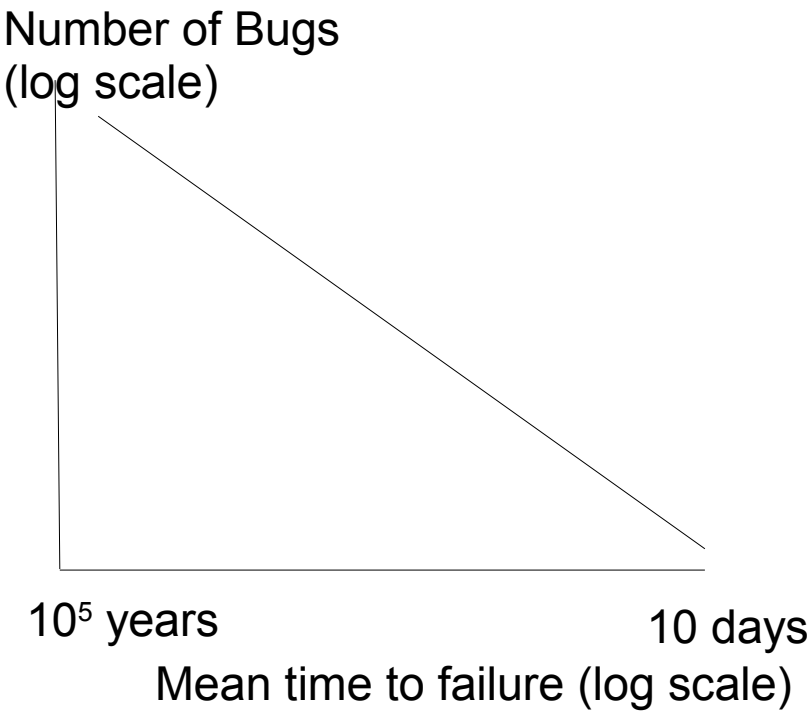


Most bugs are very hard to find



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

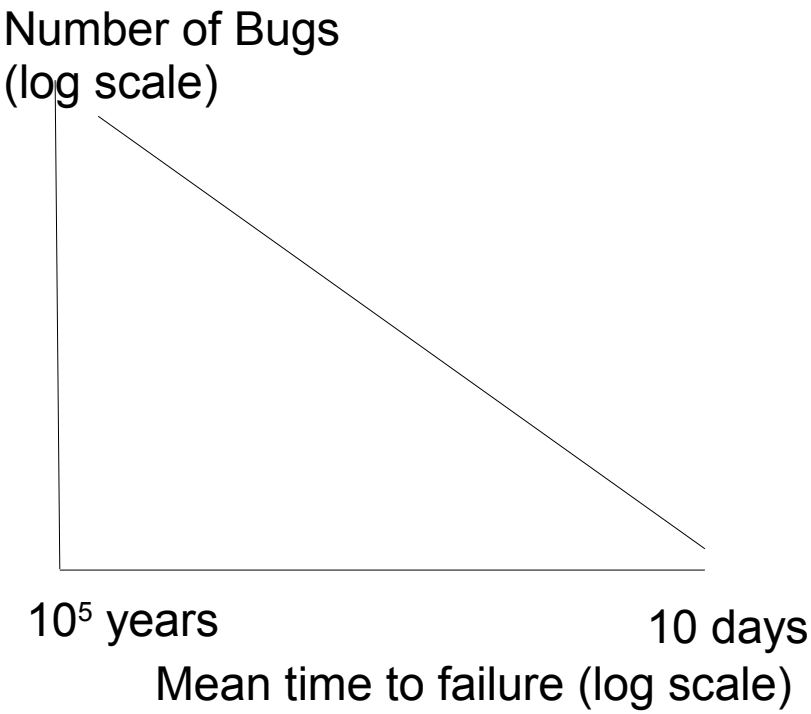
Most bugs are very hard to find



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

Period	average	bugs	mttf
10–20d	15d	1	15d

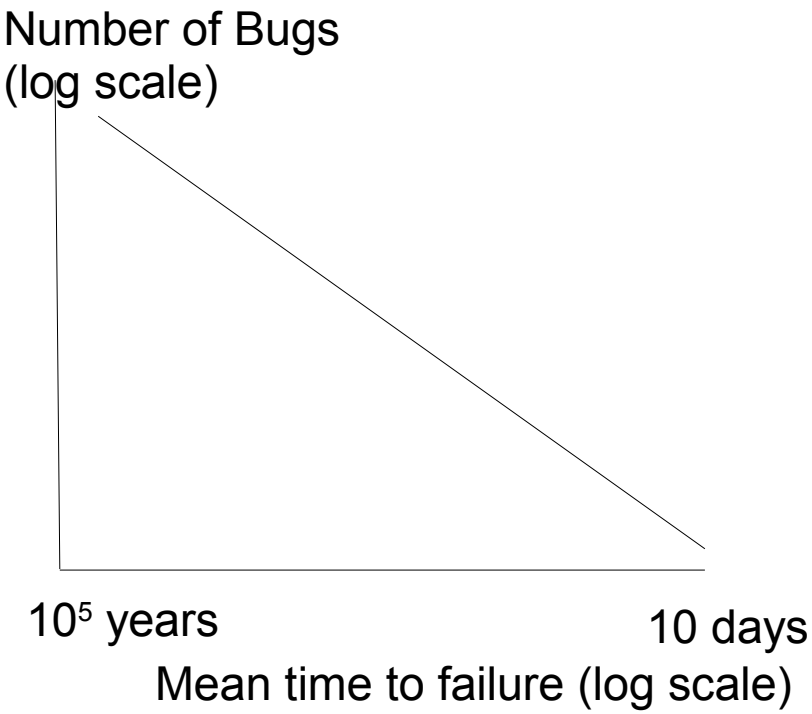
Most bugs are very hard to find



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

Period	average	bugs	mttf
10–20d	15d	1	15d
20–40d	30d	2	15d

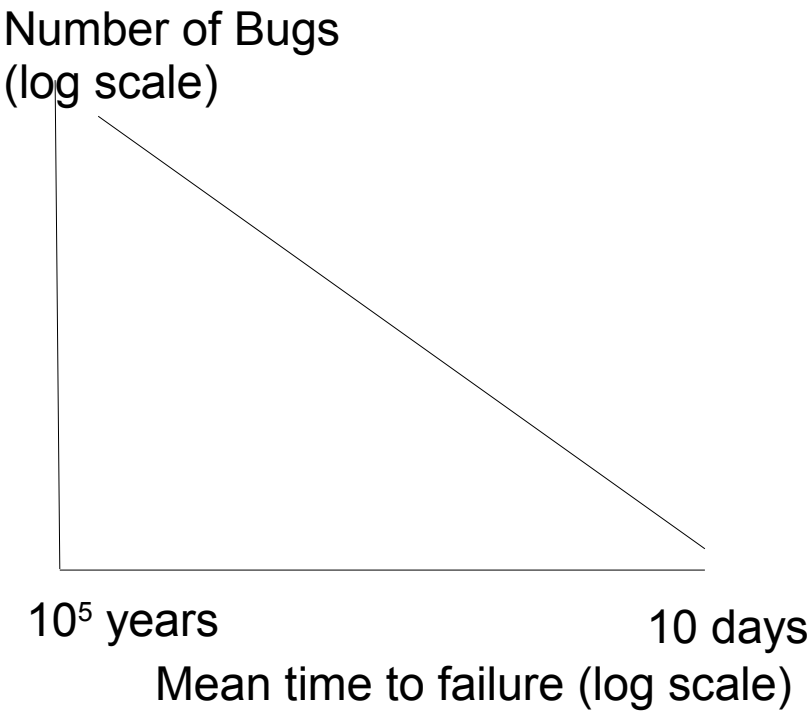
Most bugs are very hard to find



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

Period	average	bugs	mttf
10–20d	15d	1	15d
20–40d	30d	2	15d
40–80d	60d	4	15d

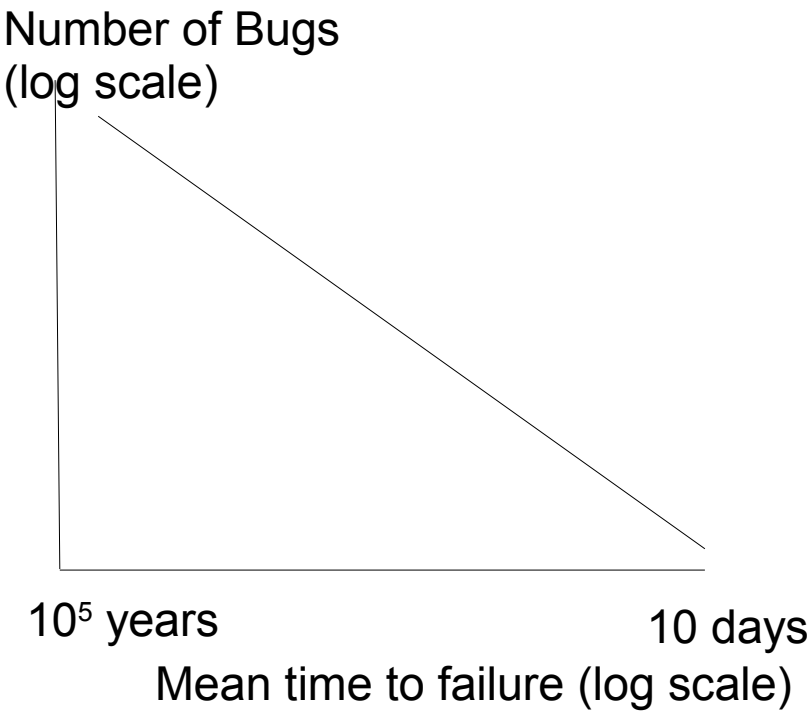
Most bugs are very hard to find



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

Period	average	bugs	mttf
10–20d	15d	1	15d
20–40d	30d	2	15d
40–80d	60d	4	15d
80–160d	120d	8	15d

Most bugs are very hard to find



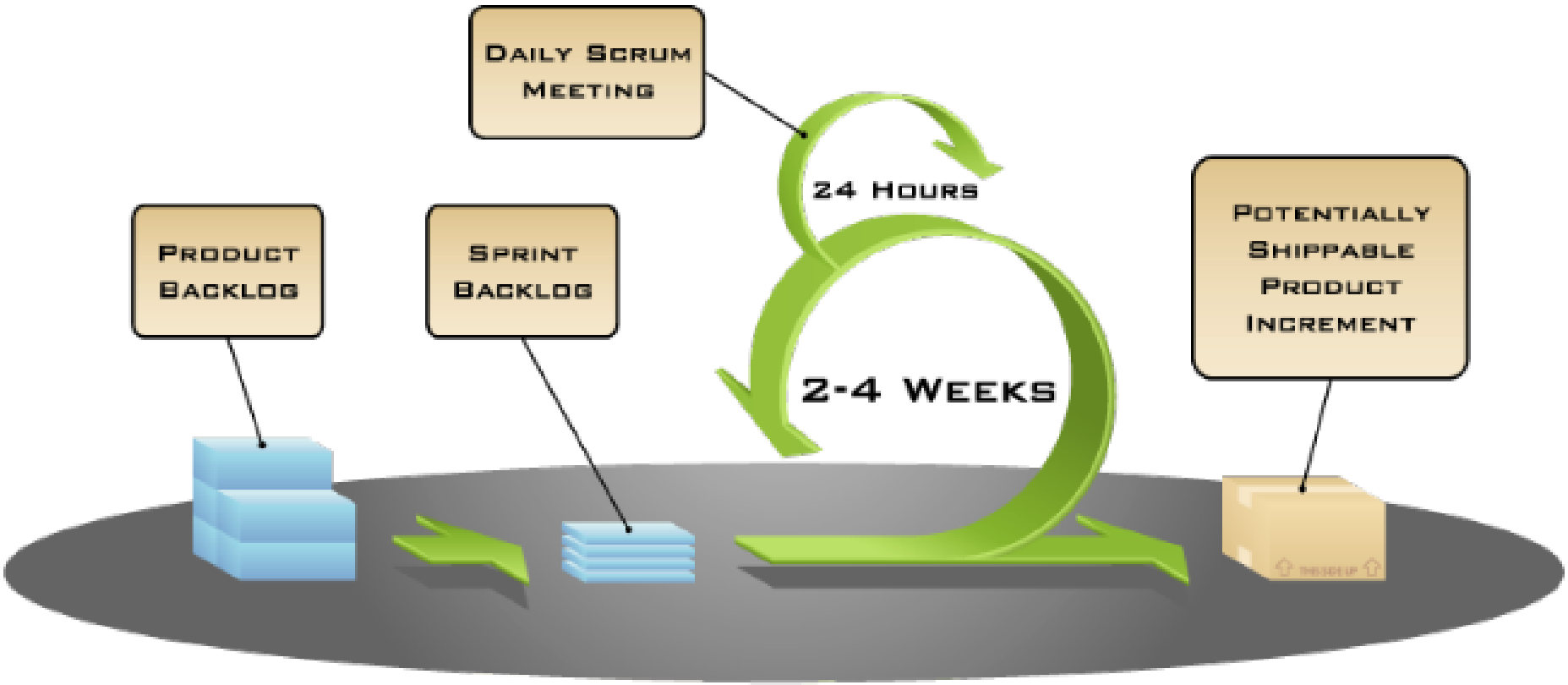
Adams E. N., Optimising preventive maintenance of software products, IBM Journal of Research & Development, Vol. 28, issue 1 pp 2–14 (1984)

Period	average	bugs	mttf
10–20d	15d	1	15d
20–40d	30d	2	15d
40–80d	60d	4	15d
80–160d	120d	8	15d

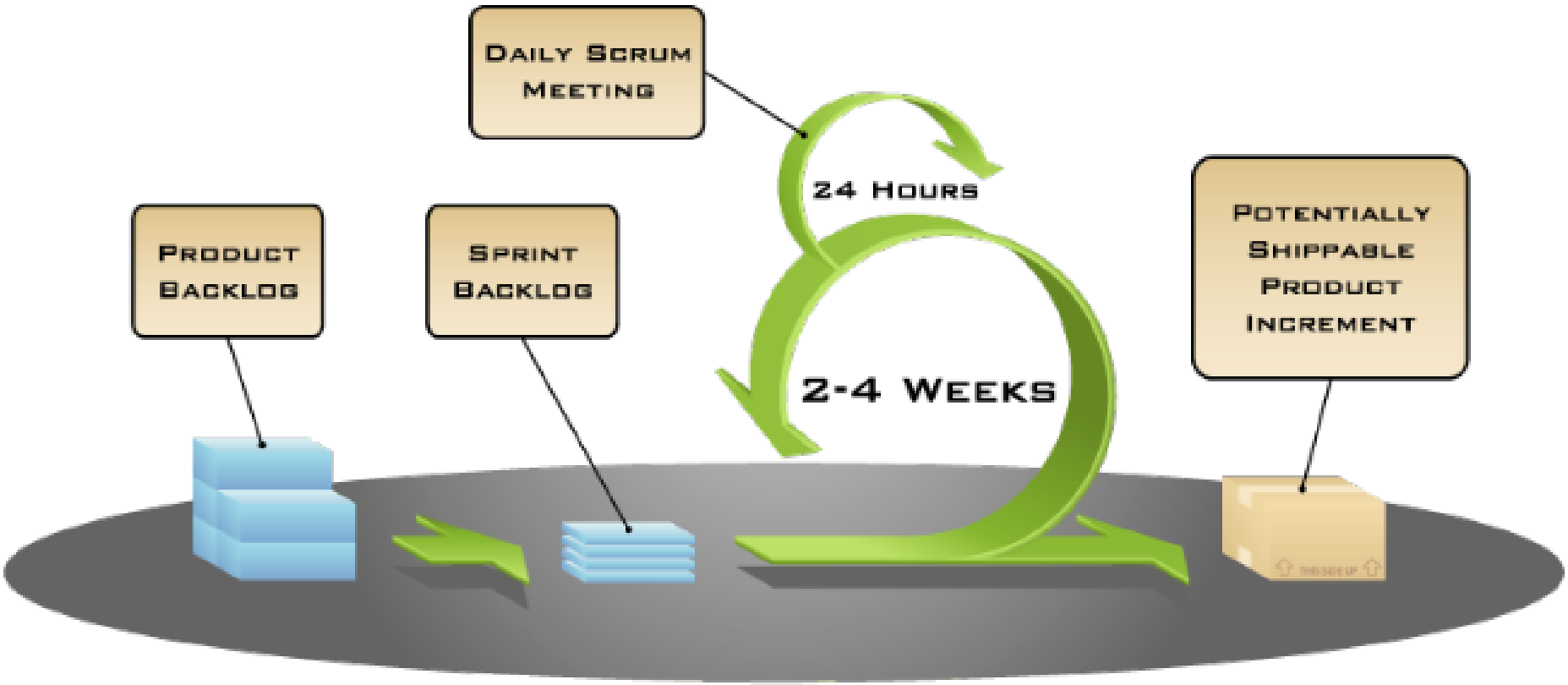
No hope without automated tests

Software development

Short development cycles (2-4 weeks)



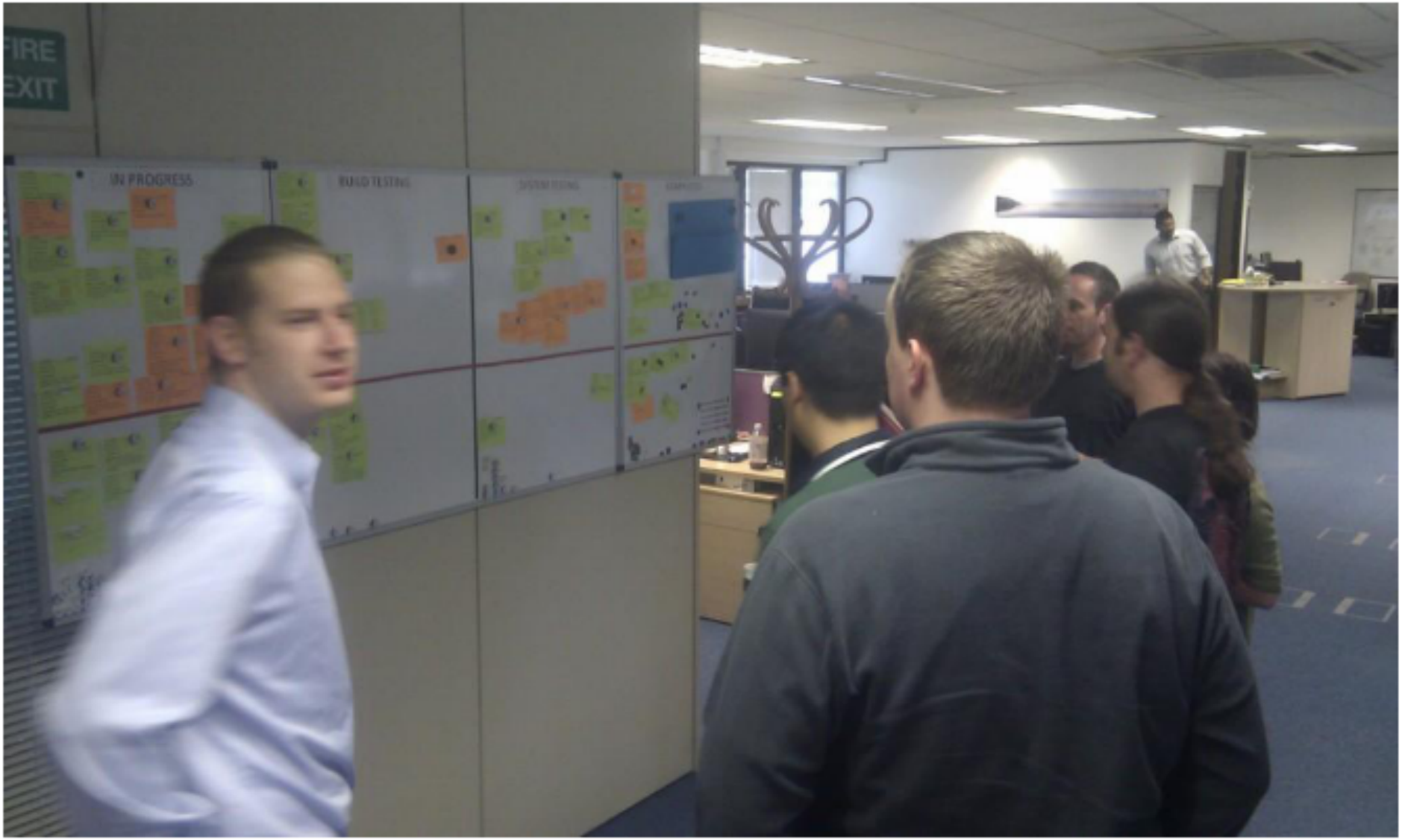
Short development cycles (2-4 weeks)



Mange using open source sytems (or market leader, Atlassian Jira)

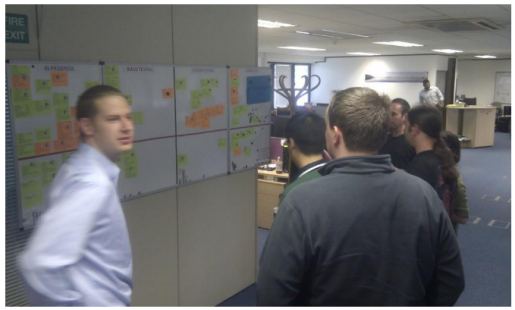
- Integrated with customer bug reporting, feature requests etc.

Daily scrum meeting



Daily scrum meeting

- Daily
- 15-minutes
- Stand-up



Co-ordination, not problem solving

- Whole world is invited
- Only ‘pigs’ may talk
  - team members
  - Scrum master
  - product owner
- Helps avoid other unnecessary meetings

Write tests before the code if possible

Write tests before the code if possible

Report a bug

- Write a test script
- Run it with each daily build (it fails!)
- Make it work and you are done
- Run the tests automatically with each daily build
  - Find out when someone else breaks your feature
  - May find regression problems unrelated to your feature
  - Refactor safely

Write tests before the code if possible

Report a bug

- Write a test script
- Run it with each daily build (it fails!)
- Make it work and you are done
- Run the tests automatically with each daily build
  - Find out when someone else breaks your feature
  - May find regression problems unrelated to your feature
  - Refactor safely

New features

- User stories from the customer
- Write the tests based on the user stories (check with user!)
- Make it work and you are done
- Run the tests automatically with each daily build

Software development

Focus on the team, not the individual

Scrum process builds team commitment

- Team members commit at the beginning of a sprint
- They help each other to achieve a shippable product
- Autonomy – management not allowed to interfere during sprint

Developers work with customers

- Key to success and motivation
- Help the team understand what the customer really needs

Getting teams to work together

Try to combine the best features of two products

- Bring two teams together
- Each sees the other as competition
- Personal experience – Two key mangers left, with half their teams

Getting teams to work together

Try to combine the best features of two products

- Bring two teams together
- Each sees the other as competition
- Personal experience – Two key mangers left, with half their teams

What seems to be working

- Start slowly (common log-on screen)
- Cross-platform technology (restful services)
- Promote re-usable code into a common repository with more rigorous controls
- Introduce features from product A into B
- Lubricate with plenty of alcohol

Software development