The software development proccess
A personal view

Robert Brady

robert.brady@cl.cam.ac.uk

May 2017

Program size

Approx Size	Description	Approx date
1kb	Punch-card program	1965
2kb-10kb	Computer Science project	
16kb	Operating system of Sinclair Spectrum	1982
18 Mb	Human Genome – protein coding	
20Mb	Our trading system	1996
500Mb	Windows (50 Mlines)	2015
20Gb	Google (2 Glines)	2015
300Gb	Storage on my laptop	

Complex, vast number of ‘corner cases’

Recruiting great developers

‘Just recruit great developers’

- They are 10-50 times more productive than ordinary developers
- who are 10-50 times more productive than poor developers
- Self-starters: artists who don’t want or need managing
- Most of you

For a growing company, great developers...

- Do not exist in sufficient numbers
- Progress to more important things (sales, CEO)

So this is for the rest of us...

What management can do

Tell developers what you want them to do, measure their performance

- Ordinary software companies do this
- Turns great developers into ordinary ones
- Lose a factor of 10-50 productivity

Supply infrastructure for developers to succeed, and trust them

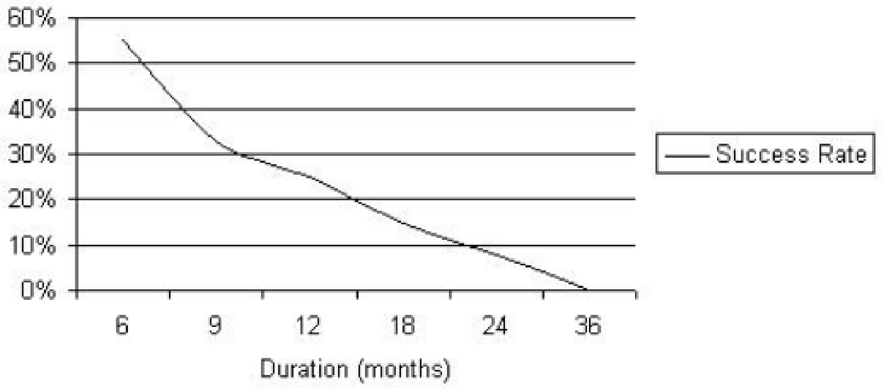
- Communicate the problem – eg we really want to land customer A
- Cut out the middle-man – eg onsite customer experience
- Autonomy – eg work on what you want 20% of your time
- Recognition – eg ‘brown bag’ talks
- Creates great developers

Not limited to developers

- Literature on "Theory X-Theory Y" management culture

Statistics (1)

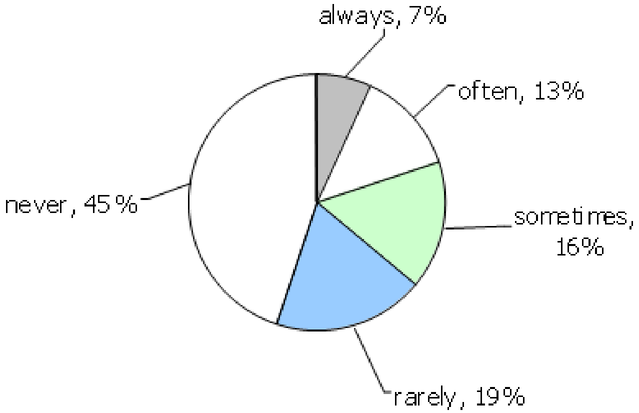
Success rate of projects (Johnson 1998)



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

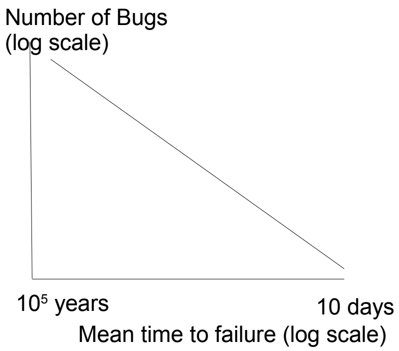
Statistics (2)

Actual use of requested features (Johnson 2002)



Ship frequently. It is the only way to stop wasting effort.

Statistics (3)



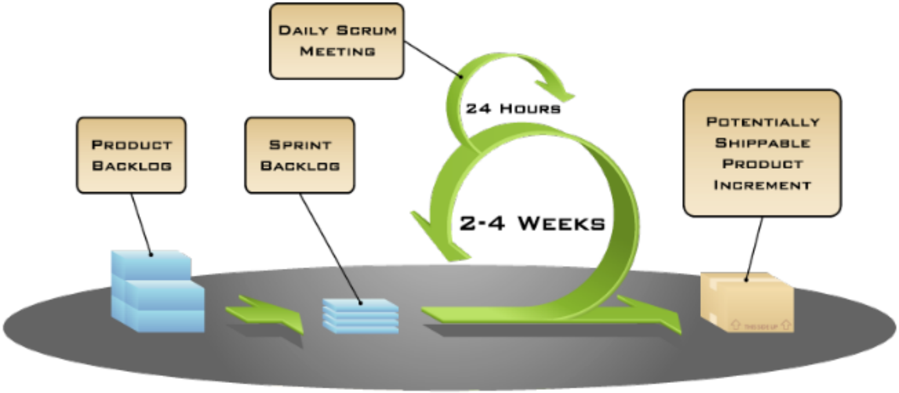
Adams E. N., Optimising preventive maintenance of software products, IBM Journal of Research & Development, Vol. 28, issue 1 pp 2–14 (1984)

Hard-to-find bugs really matter

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

Run automated tests every day

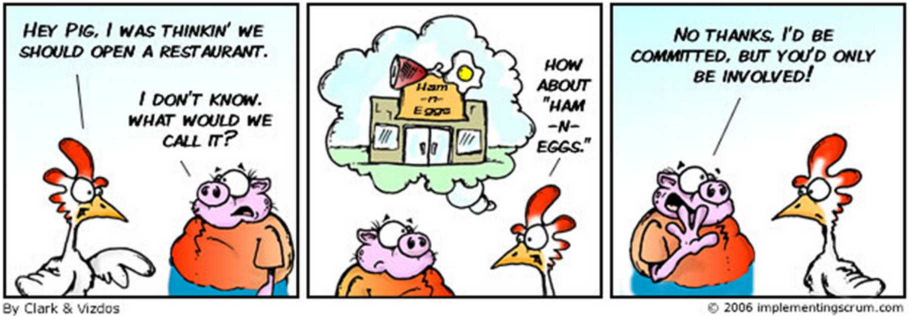
Short development cycles or ‘sprints’



Most people use Jira by Atlassian (equivalent of Salesforce)

- Everything you need
- Integrated customer site
- Statistics for team motivation and feedback

Agile Commitment model


HEY PIG, I WAS THINKIN' WE SHOULD OPEN A RESTAURANT.
I DON'T KNOW. WHAT WOULD WE CALL IT?
Ham -n- Eggs
HOW ABOUT "HAM -N- EGGS."
NO THANKS, I'D BE COMMITTED, BUT YOU'D ONLY BE INVOLVED!
By Clark & Vizdos
© 2006 implementingscrum.com

How a sprint works

Team members (‘pigs’) commit at the beginning of a sprint

- Scheduling meeting with ‘chickens’ (sales, management ...)

Chickens refrain from interfering during sprint

- Unplanned changes reduce efficiency and motivation
- Customers don’t deploy fixes in less than 2 weeks anyway

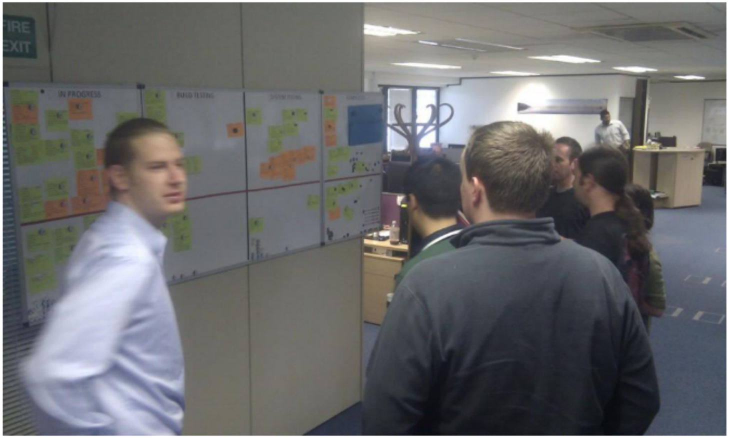
Scrum master fixes problems

- My computer doesn’t work
- The customer didn’t answer my question

Team succeeds, not individuals

- eg, John hasn’t finished testing, Fred will help
- Fred gains cudos with peers
- Part 1B project is run this way

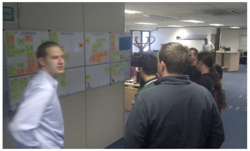
Daily scrum meeting



Old picture – now cluster around a Jira screen

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

There is an ‘I’ in team

Developers at customer site

- Transforms understanding and motivation
- Fixes things at the airport on the way back

Developers have 20% own time

- Gives a ‘Brown bag’ talk on what he did
- Enables large organizations to innovate

Test-oriented development – example

```
Class MyClass

int Square(int a) {
    return a**2
    (return a*a in v.2)
}

Test() {
    assert Square(0) = 0
    assert Square(-5) = 25
    assert Square(5) = 25
}
```

Some environments run the tests every time you check in

Test manager's job – "prove to me that your change works"

Automated tests

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