Sixth International TOCPA Conference
18-19 May, 2013, Moscow, Russia

TOC and Software development

Clarke Ching

Agile Lead

Royal London Group

UK

Clarke Ching

Clarke Ching is an expert in the application of Eli Goldratt's Theory of Constraints to Agile software development. He is author of Rocks into Gold and Rolling Rocks Downhill (BETA). He gained Scrum certification from Ken Schwaber in Scotland in 2004 and now works as an Lean and Agile internal expert Royal London Group. Clarke focuses on Cash-Flow-Driven-Development & the use of positive psychology to help development teams flourish.

Clarke.ching@gmail.com
www.clarkeching.com

Agile builds on CCPM thinking, but …

	Critical Chain	Agile
Attacks these problems:	Critical Path	Critical Path & Waterfall Engineering (big test phase at end)
Project Promise	Deliver agreed scope on Time. Faster.	Maximize Cash-flow & Deliver on time or earlier.
Buffers	By adding time buffer	By dropping low value features.
Visualization of Execution	Gantt Chart with Buffer	Release (or product) Map
Best when:	Scope & Solution clearly known. (e.g. bridge)	Scope & Solution Emerge
Use the other option when	Use Agile when building stuff which is easy to grow incrementally.	Use Critical Chain when building things which are hard to grow incrementally
Key behaviour change	Don't treat estimates as commitments	Test in parallel, not at the end.

Agile & TOC in general

Flow system: Bottleneck & Batch Size:

Agile transforms a big batch into smaller batch or flow system.
It delivers many smaller things inside each batch.
-> This creates a flow system
-> The flow system has a bottleneck
-> The internal bottleneck is (almost always) the capacity to test.
-> We try to automate the testing, but there are other ways to exploit (such as avoid work which requires a lot of testing).

Distribution Solution

Agile is similar to the distribution model because, rather than building to a big-batch forecast, we build a small-batch forecast, ship it, see if the customer likes it, then adapt.

When do we start executing? When do we release?

Critical Chain plans rely on a good (but not excessively detailed) understanding of the solution. Execution doesn't start until the solution is known.

Agile plans break a big solution down into 3 or 4 releases, plan the first bit of release 1 in detail, then start executing.

Agile projects try to "go live" many times during a project. Critical Chain projects can do this too.

You can use Critical Chain inside agile projects too

See my article:
http://www.stickyminds.com/s.asp?F=S13948_COL_2

Three types of software Development work

1. Maintenance - Smallish projects, adding to, or fixing, existing code.
   - Manage Flow using 3 lists:
     1. List 1- Not started (prioritized weekly)
     2. List 2- Doing (ideally limited to 1 or 2 requests per dev)
     3. Done!
   - Constraint? Normally code complexity & testability.
2. Big Projects
   1. Gantt Chart
   2. Analysis, Design, Development, (very long) Test&Rework phase …
3. Maintenance work batched up and done as a project.

Quiz: Count the f's

Finished files are the result of years of scientific study combined with the experience of years ….

Quiz: Count the f's

Finished files are the result of years of scientific study combined with the experience of years ….

Quiz: Count the f's

Finished files are the result of years of scientific study combined with the experience of years ….

Quiz: Count the f's

Finished files are the result of years of scientific study combined with the experience of years ….

Answer = 5

(I meant lower case)

Why **WALK** when you can **RUN** for **FREE**?

Agile 101

Clarke Ching

Agile 101 – Introduction

The Problem
The Desire
&
The Opportunity

The problem: Russian Roulette

- Average IT project overrun: 27%
- But 1 In 6 IT projects overrun >200%

From "Why Your IT Project May Be Riskier Than You Think"
by Bent Flyvbjerg and Alexander Budzier , Harvard Business Review, Sept. 2011

The problem: Russian Roulette

- Average IT project overrun: 27%
- But 1 In 6 IT projects overrun >200%

The desire: Predictability: Finish on Time.

From "Why Your IT Project May Be Riskier Than You Think"
by Bent Flyvbjerg and Alexander Budzier , Harvard Business Review, Sept. 2011

The opportunity: Make More Money. What happens if you finish every project 25% sooner?

Before: AAAABBBBCCCC

After: AAABBBCCC

You gain 33% more staff, for free.
You can do 33% more work, for free.

But do you have more work waiting?

What if you "sold" or "used" half that 33% increase?

Fixed Costs	£2,000,000
Revenue per project	£800,000
Before: 3 projects	£2,400,000
Profit:	£400,000

But now you can do 4 projects!

But you can only sell a small one, bringing in revenue of £400,000.

Your revenue goes up to £2,800,000

Your profit £800,000

Your profit has doubled!
You (and your boss!) are heroes.

Agile 101 – Part 1

Don't be late.

Software Development is *naturally* Iterative and Incremental.

NHS chaos exposed by new e-mails

A COMPUTER project costing £6.2 billion that is central to Tony Blair's National Health Service reforms is in "grave" danger of being "derailed", leaked Whitehall e-mails reveal.

The warning has been issued by Richard Granger, the £250,000-a-year civil servant in charge of what has been billed as the world's biggest civil information technology project.

The scheme is central to the government's plans to give patients wider choice by allowing GPs to book hospital appointments online with consultants throughout the country.

The problems have already caused a year-long delay in the booking system and now threaten to add millions to the cost of the project.

To date the system has made only about 20,000 appointments for patients. It was supposed to have made 250,000 by December 2004.

When it is fully operational the system is meant to be capable of making up to 9.5m first hospital appointments a year.

In the e-mail exchanges in September, Granger blames a senior civil servant in the Department of Health for the fiasco, criticising her **repeated last-minute changes** and failure to heed his advice.

Granger censures Margaret Edwards, the department's director for access and patient choice, for **adding numerous new specifications** to the booking programme, known as Choose and Book. Granger writes: "Choose and Book's £20m IT build contract is now in grave danger of derailing (not just destabilising) a £6.2 billion programme." He concludes: "Unfortunately, your **consistently late requests** will not enable us to rescue the missed opportunities and targets."

Sir Nigel Crisp, the NHS chief executive, was forced to admit to the Commons health select committee two weeks ago that the booking system was at least a year behind schedule. However, he failed to mention that the delay was having a serious impact on the entire project.

The National Audit Office has identified changes to specifications after the award of IT contracts as a key reason for regular delays and overspends on government projects.

Granger's comments were triggered by an e-mail on September 9 from Edwards marked "Restricted — Policy" which begins: "We have a problem!" The e-mail reveals that patients and their GPs still cannot book treatment at any of the country's 32 foundation trust hospitals by computer because they are not on its "choice menu".

The 10 private sector treatment centres, set up by the government to reduce waiting lists, are also absent from the official list on the computer.

Edwards warns that the treatment centres and foundation trusts will not be on the "choice menu" until next summer.

The delay places hospitals at a financial disadvantage. Under the government's payment-by-results regime, they are supposed to compete with other NHS hospitals for patients.

Edwards admits: "We haven't yet told ministers that there is a problem."

Granger was incensed by the implied criticism of the booking system and fired off a trenchant 11-point reply. Although Edwards's original e-mail was encrypted and her password protected, Granger decrypted it, sent it out with his reply and widened the distribution.

Granger complains that **the project has been allowed to change beyond recognition from the original specification.** "The original request from your predecessor and yourself was for an Electronic Booking System. The change of this to Choose and Book occurred in (the second quarter of) 2003. This was the first of what are **now recurrent major changes in your requirements."**

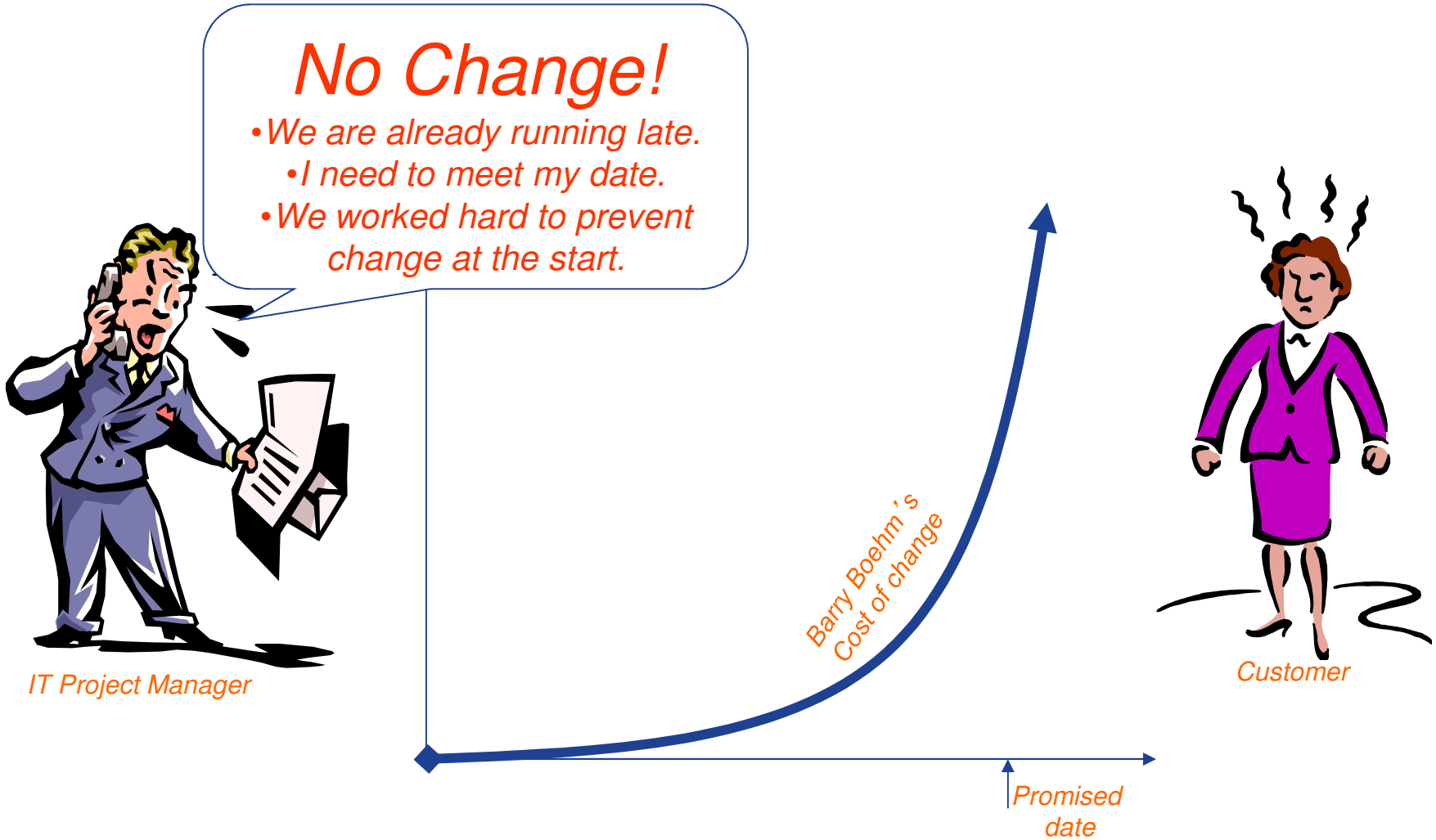
The booking system has been dogged with difficulties since its inception. GPs have refused to use it and early pilot schemes identified fundamental software design flaws.

Last week Granger, who insists that the booking system now works, broke civil service protocol and publicly blamed policy officials in the Department of Health for failing to get GPs to use the system. In an interview with Computing Magazine, he said: "Low usage is not something I can do anything about."

Both the health department and Granger's spokesman refused to comment on the leaked e-mails.

Jonathon Carr-Brown, The Sunday Times, November 13, 2005
http://www.timesonline.co.uk/article/0,,2087-1869851,00.html

The Change Conflict



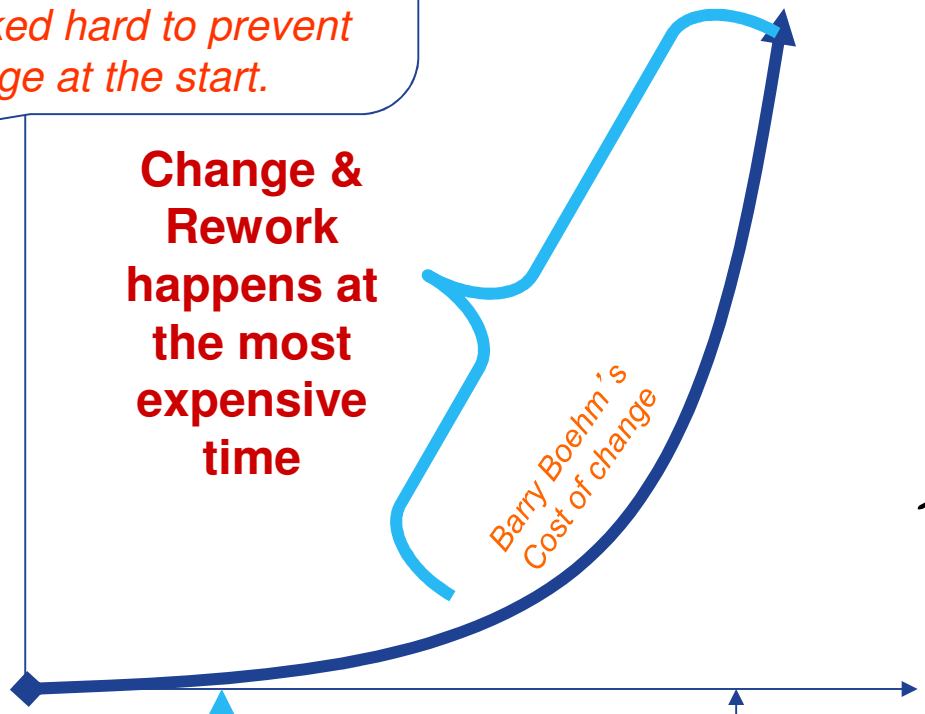
The Change Conflict

No Change!

- *We are already running late.*
- *I need to meet my date.*
- *We worked hard to prevent change at the start.*

IT Project Manager

Change & Rework happens at the most expensive time



Spec signed off here

Promised date

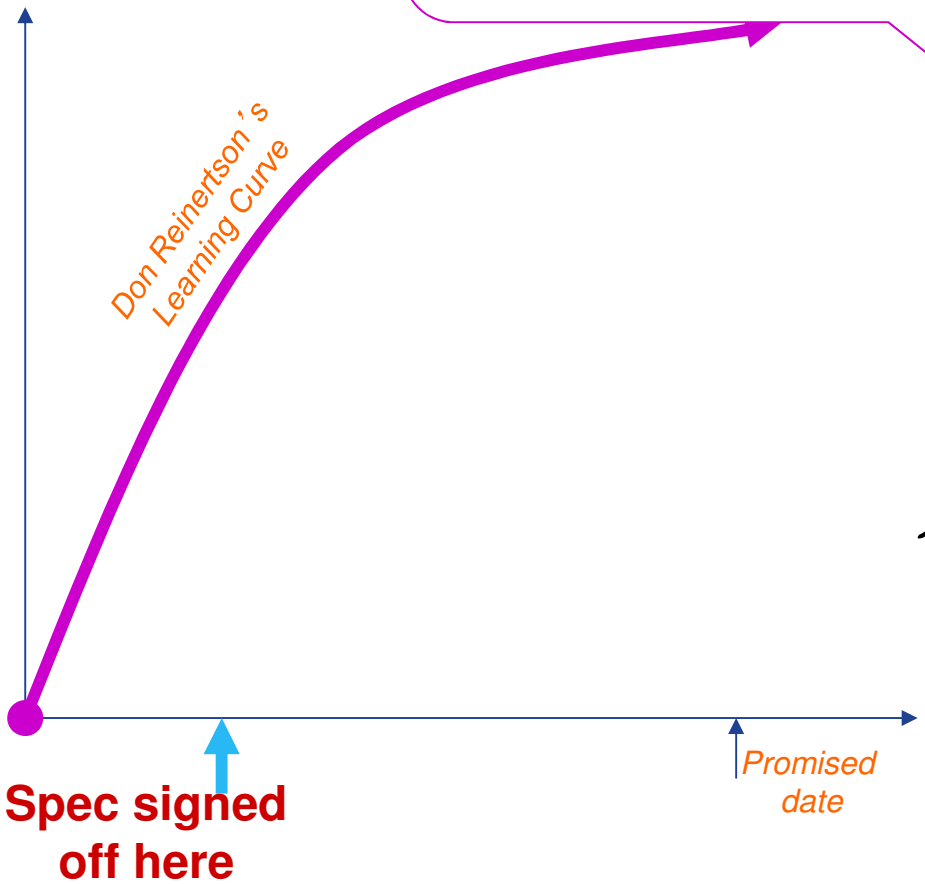
Customer

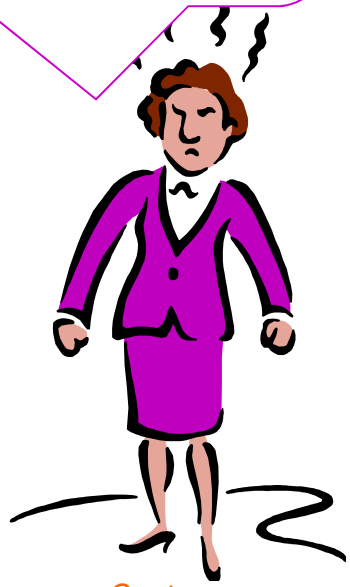
The Change Conflict

Change!

- *Our spec was an educated guess*
- *We learn by doing the project*
- *We need the best product*

IT Project Manager





Customer

The Change Conflict

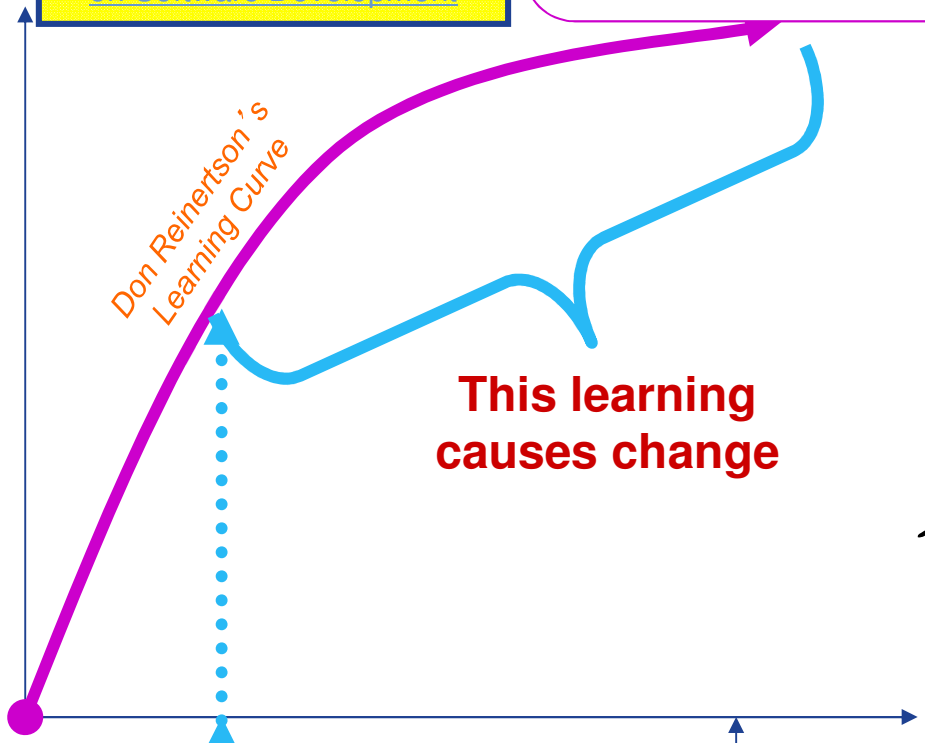
Out of hundreds of projects, **there is no case in which requirements remained stable** throughout the design - Reinertson (1998) on Product Development

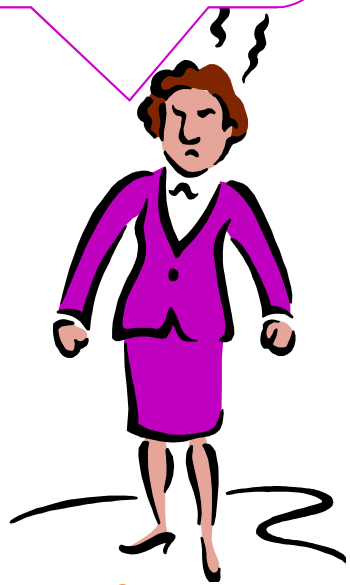
A typical software project experiences a 25% change in requirements - Boehm and Papaccio (1988) on Software Development

Change!

- *Our spec was a guess*
- *We learn by doing the project*
- *We need the best product*

IT Project Manager



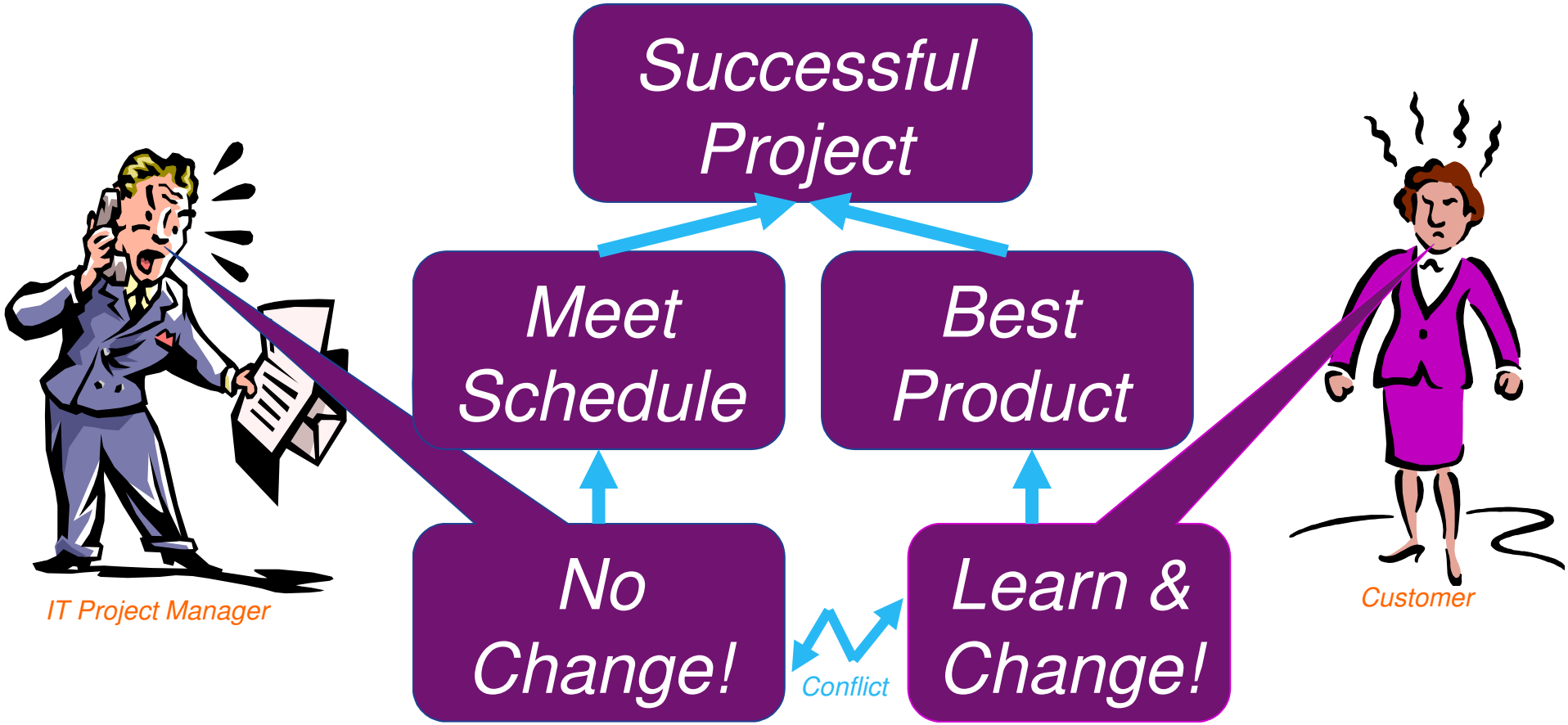


Customer

Medium to Large projects (1000+ function points) experience 25 – 35% requirements change - Jones (1997) on Software Development

Conclusion: We can't successfully *prevent* change

The Change Conflict



The Change Conflict

Who is to blame?

- The customer? ✗
- The project manger? ✗
- An simple misunderstanding in the 70's ✓

IT Project Manager

Successful Project

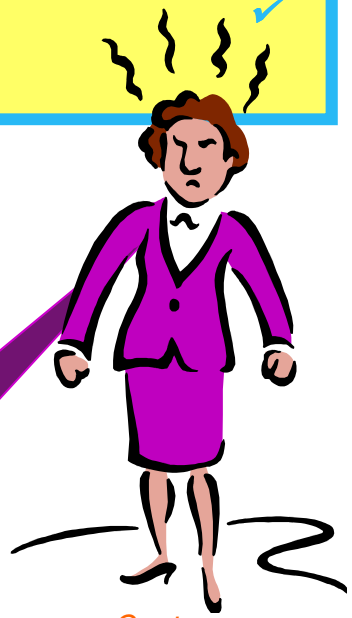
Meet Schedule

Best Product

No Change!

Conflict

Learn & Change!



Customer

A simple misunderstanding in the 1970's

In the 1970's

- The software guru's borrowed quality thinking from Japanese ***manufacturing***

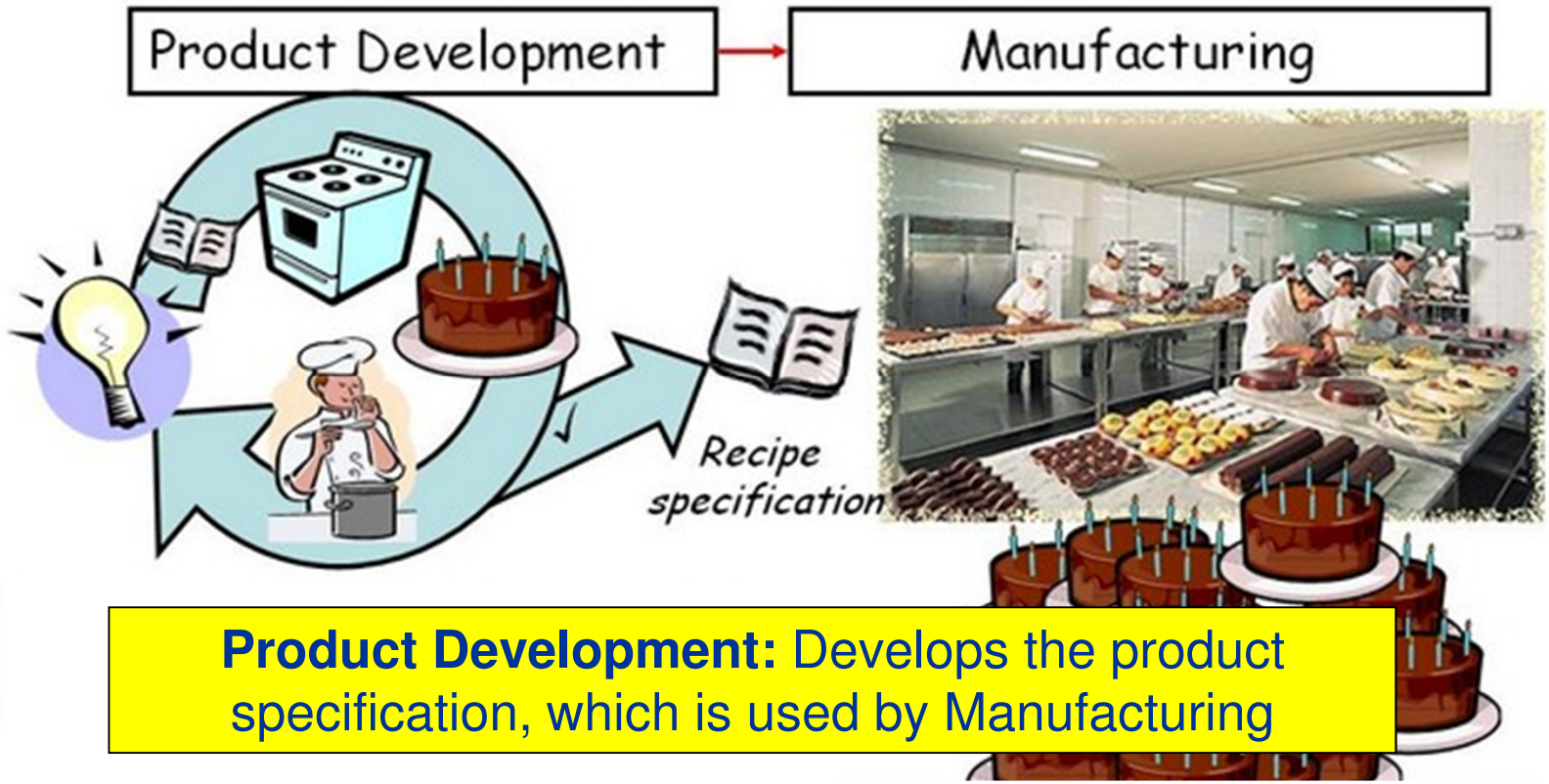
But,

- Software development <u>is not</u> manufacturing ✗
- Software development <u>is</u> product development ✓

Software development is a type of *product development*

Concept -> Iterative Learning -> Spec

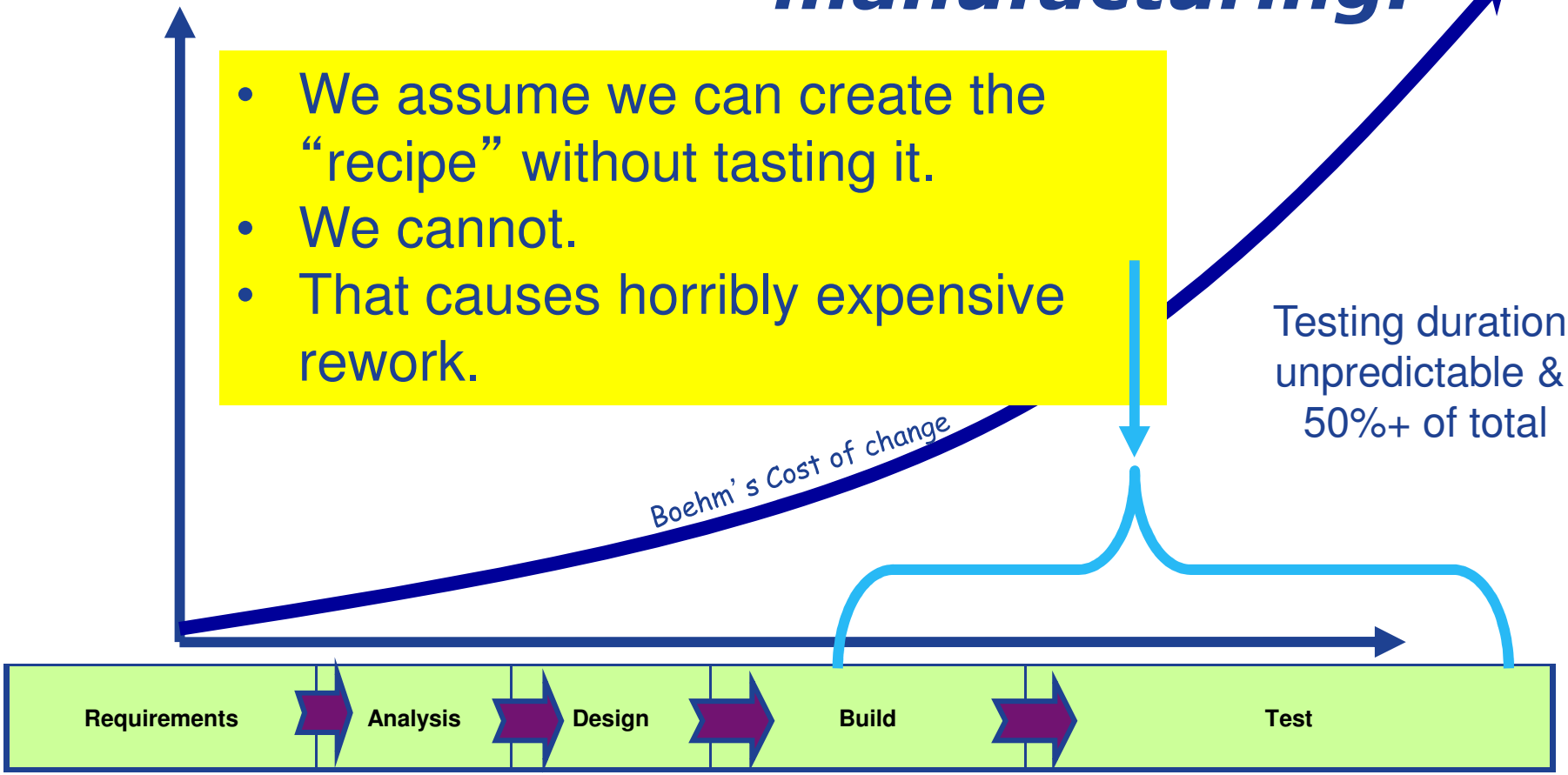
Spec -> Build to spec -> Product



Product Development: Develops the product specification, which is used by Manufacturing

But, we treat software development like it is *manufacturing.*

- We assume we can create the "recipe" without tasting it.
- We cannot.
- That causes horribly expensive rework.



Testing duration unpredictable & 50%+ of total

We often de-scope, sacrifice quality or run-late

In other words – it's like scraping burnt toast

We spend the first half of the project putting defects in

We spend the second – most expensive - half of the project reworking them out.

So, how do we "Build quality in"?

Step 1: Work in small batches.

i.e. do big projects as a series of potentially shippable smaller projects.

Agile Methodologies – such as XP, DSDM and Scrum, Kanban …

… are specifically designed to *efficiently* **cope with change.**

They do this by building software **iteratively and incrementally**.

- *The first Space shuttle control software was developed in 1970's using incremental methods*
- *US Department of Defence now demands that software is constructed using incremental and evolutionary techniques [see MILSTD-498]*
- *Barry Boehm recommended Incremental Development in the 1970s.*

Six principles of Agile Development

1: Customer lists known requirements (to a high level), then prioritises them.

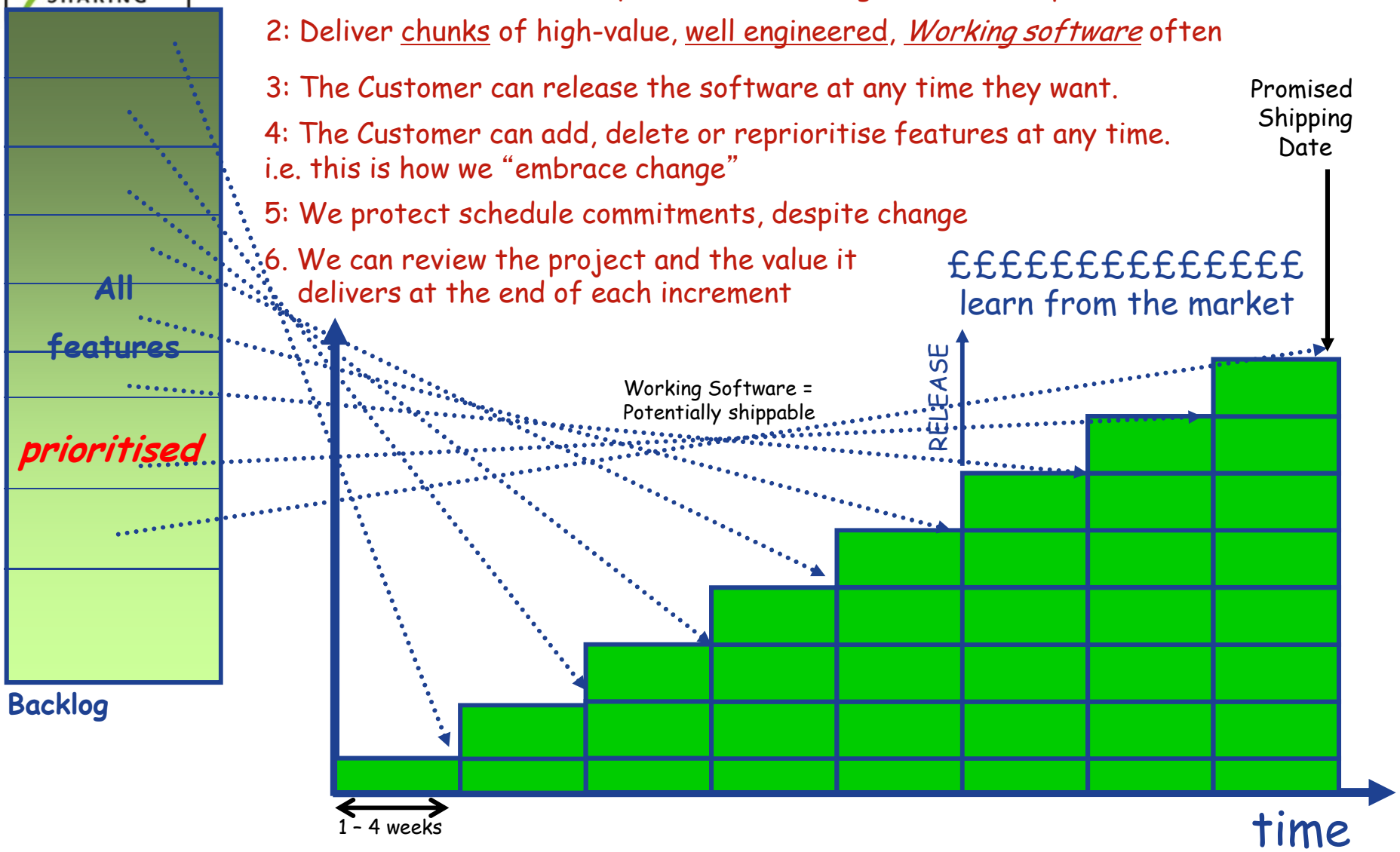
2: Deliver chunks of high-value, well engineered, *Working software* often

3: The Customer can release the software at any time they want.

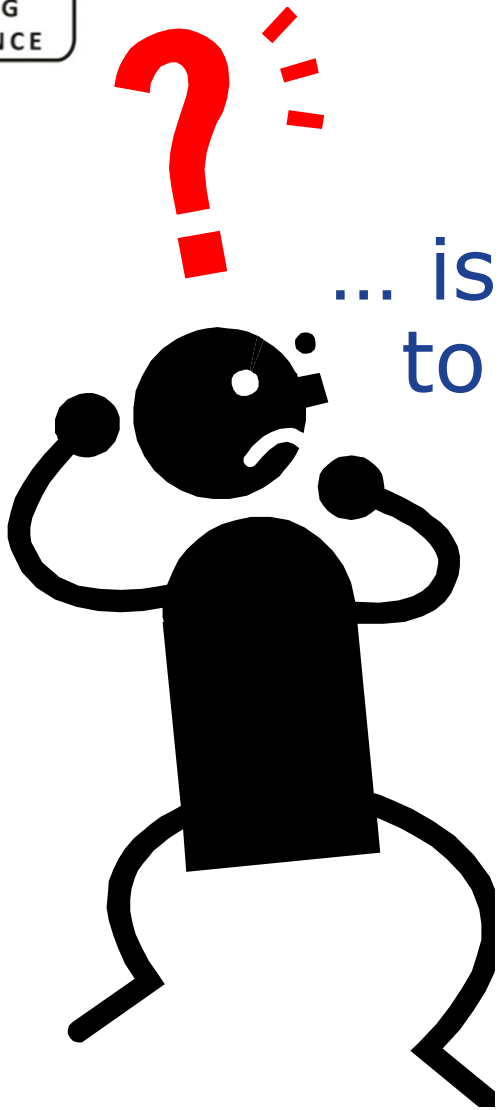
4: The Customer can add, delete or reprioritise features at any time.
i.e. this is how we "embrace change"

5: We protect schedule commitments, despite change

6. We can review the project and the value it delivers at the end of each increment

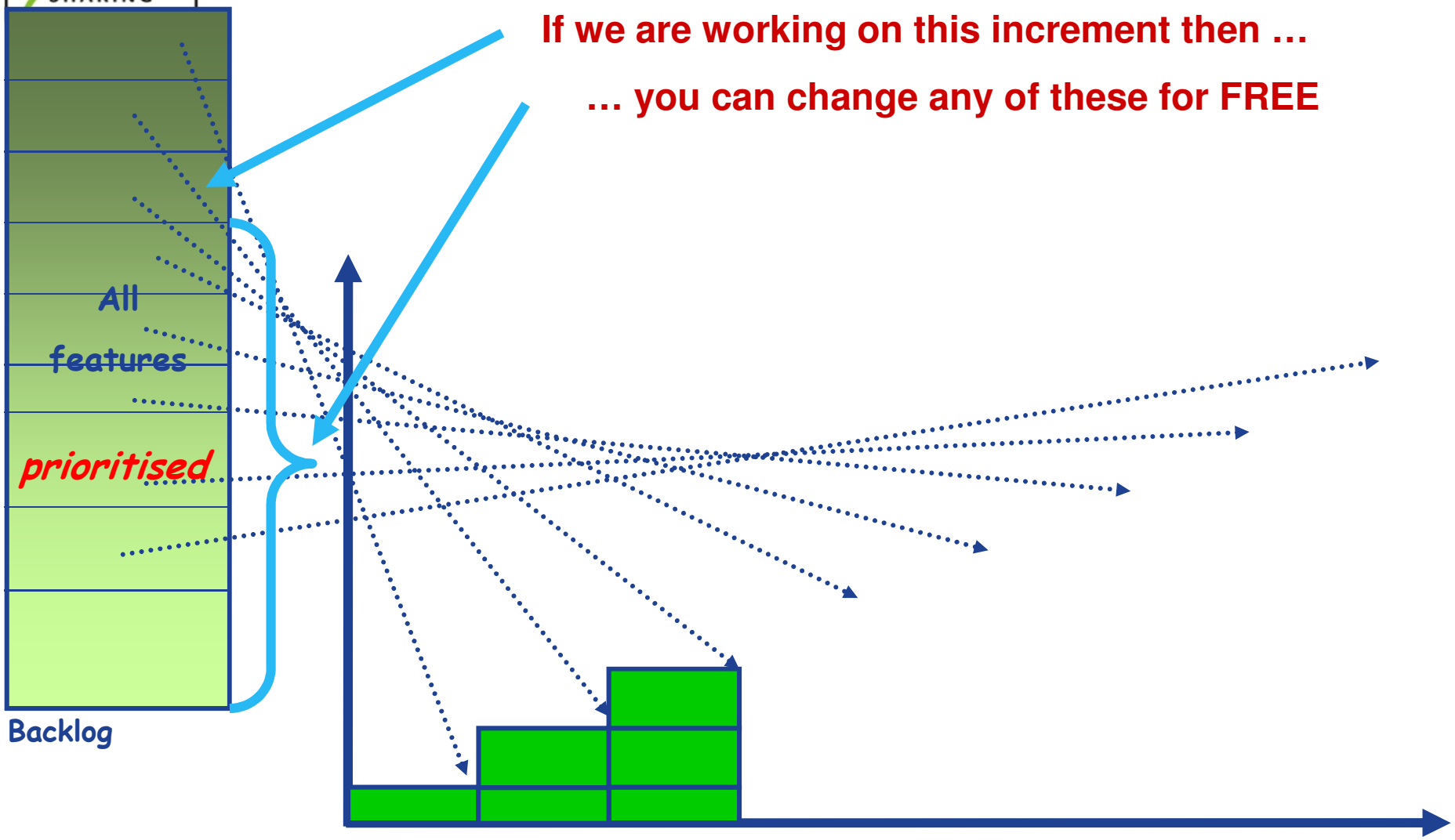


But?



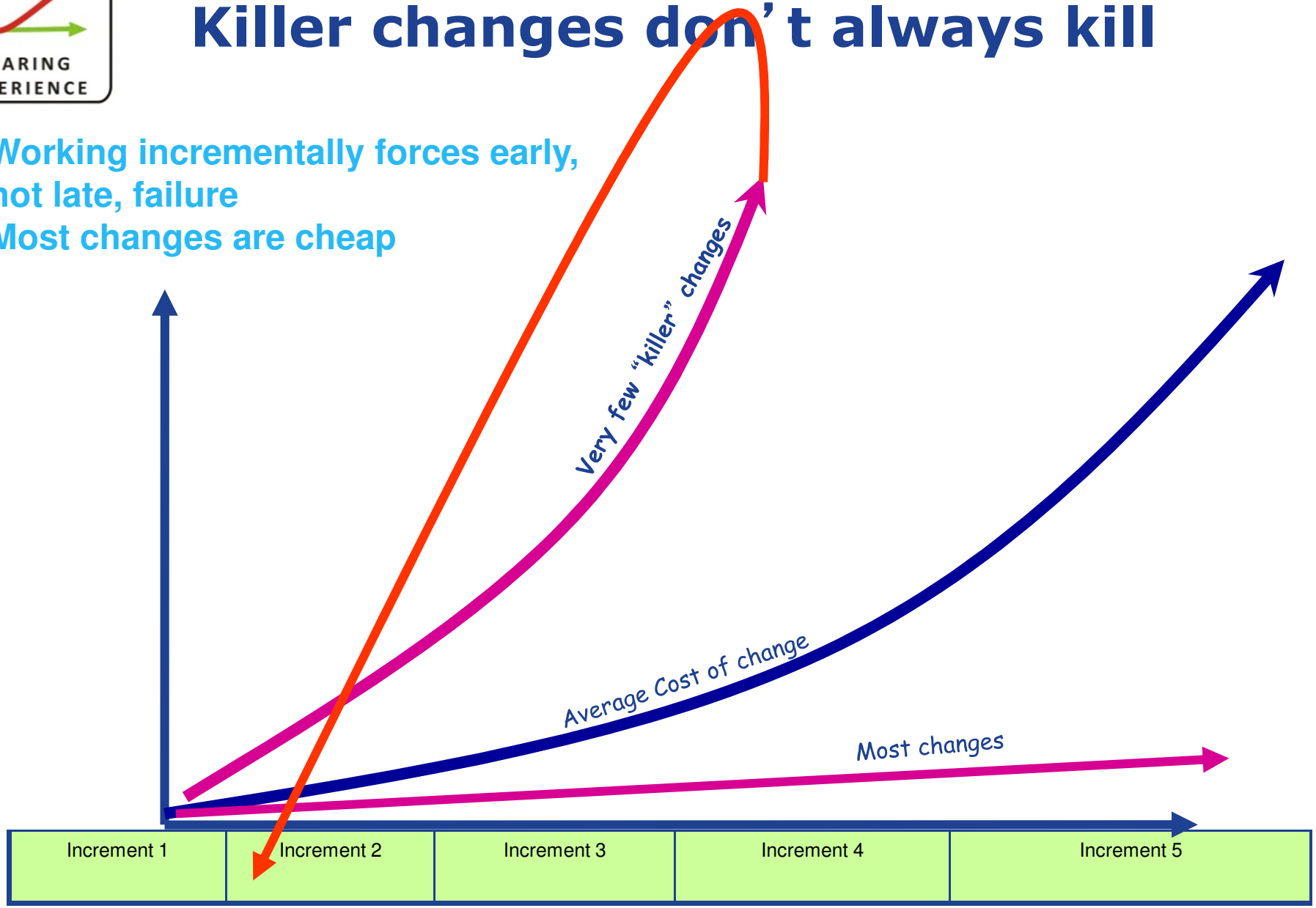
… isn't that an *expensive* way to work?

Benefit: Many changes are free

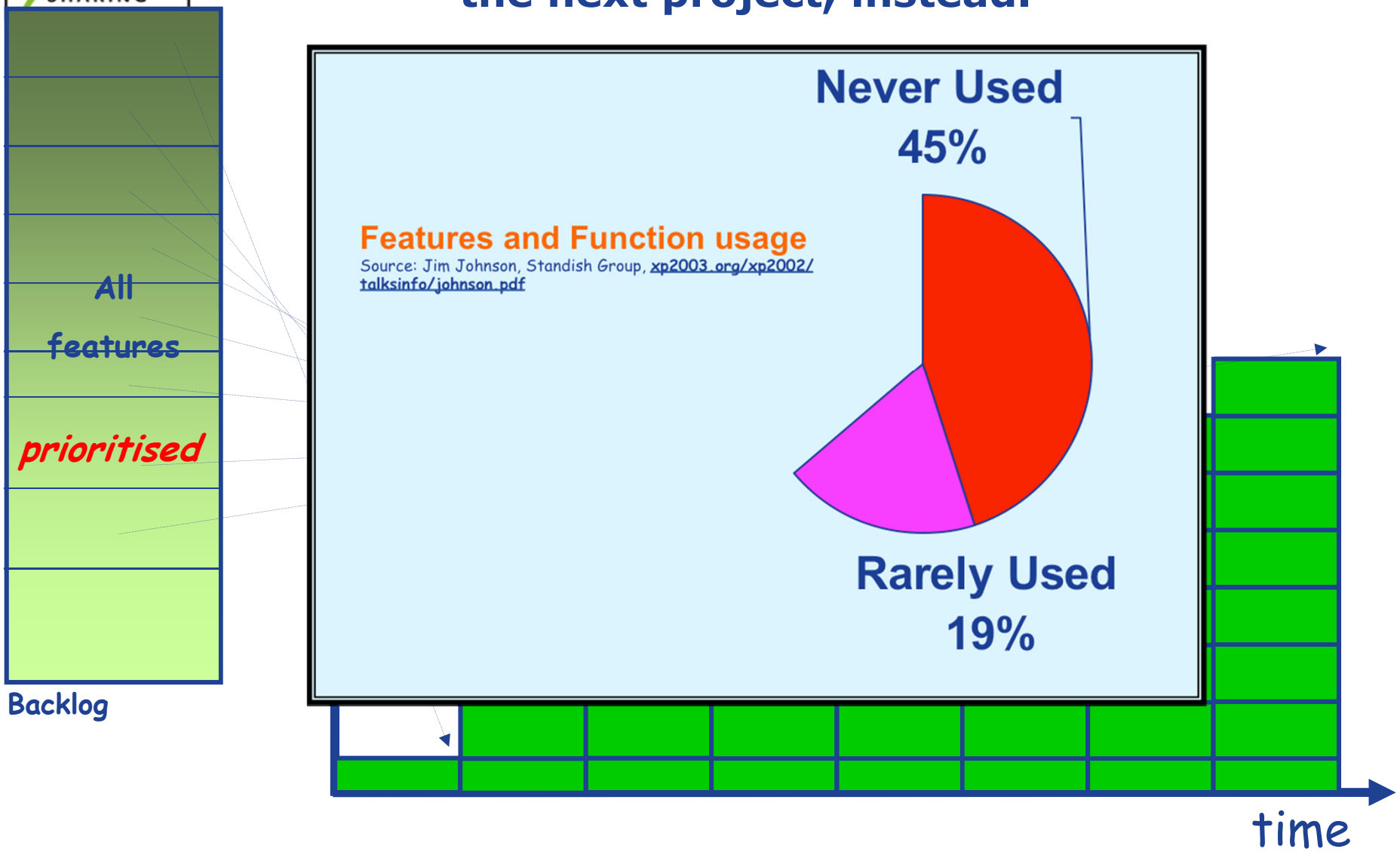


Killer changes don't always kill

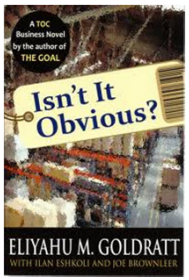
1. Working incrementally forces early, not late, failure
2. Most changes are cheap



Don't build low priority features. Start the next project, instead.



This might not be so obvious …



Software Requirements* are Forecasts.

- The customer asks to change (i.e. expensively rework) 25% – 35% of the features.
- They want to changes others but know the PM will reject the request.
- The customer never uses 45% of the features built.
- The custom rarely uses 20% of the features built.
- Many features, deemed "must haves" at start, are part built then eventually de-scoped.

under -production

over-production.

scrap

Agile is *like* the TOC Replenishment solution, combined with the Critical Chain's buffering (but with a choice between dropping low value feaures, or finishing early).

- "Requirements" = Things which are required.
- Required = Obligatory, Compulsory, Mandatory, Vital, Essential.

Agile 101 – Part 2

How do you want your steak cooked?

Software Development is a Team Sport

Quiz: Count the f's

Finished files are the result of years of scientific study combined with the experience of years ….

Answer = 5

(I meant lower case)

Correct answer?

Test & Behavioral Driven Development (TDD and BDD)

1. Conversation

How do you like your steak cooked?

Rare, Medium or Well done?

Ask before cooking.

2. Automate under-the-hood (optional but desirable)

3. Execution / Test

Test manually and/or automatically.

BDD test template

Given –
When --
Then --

TOC Operations solution

- We don't do mini-waterfall projects.
- We do create a flow system:
  - We slice and dice big features into smaller features – requests or stories.
  - We slice and dice these requests/stories further into test scenarios (which we mostly define before building)
  - We are a "test processing factory".
- Testing almost always becomes our internal bottleneck.(So we automate what testing we can).
- Customer engagement and ownership becomes our external constraint. (So we have hard discussions)
- I try to have engineers as the internal bottleneck.

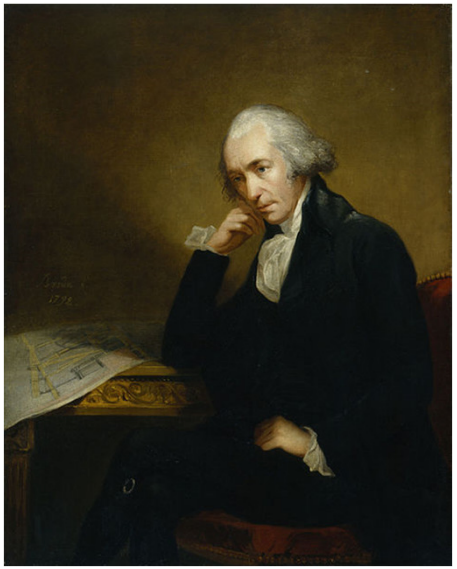
Agile 401

Cash-Flow Driven Development

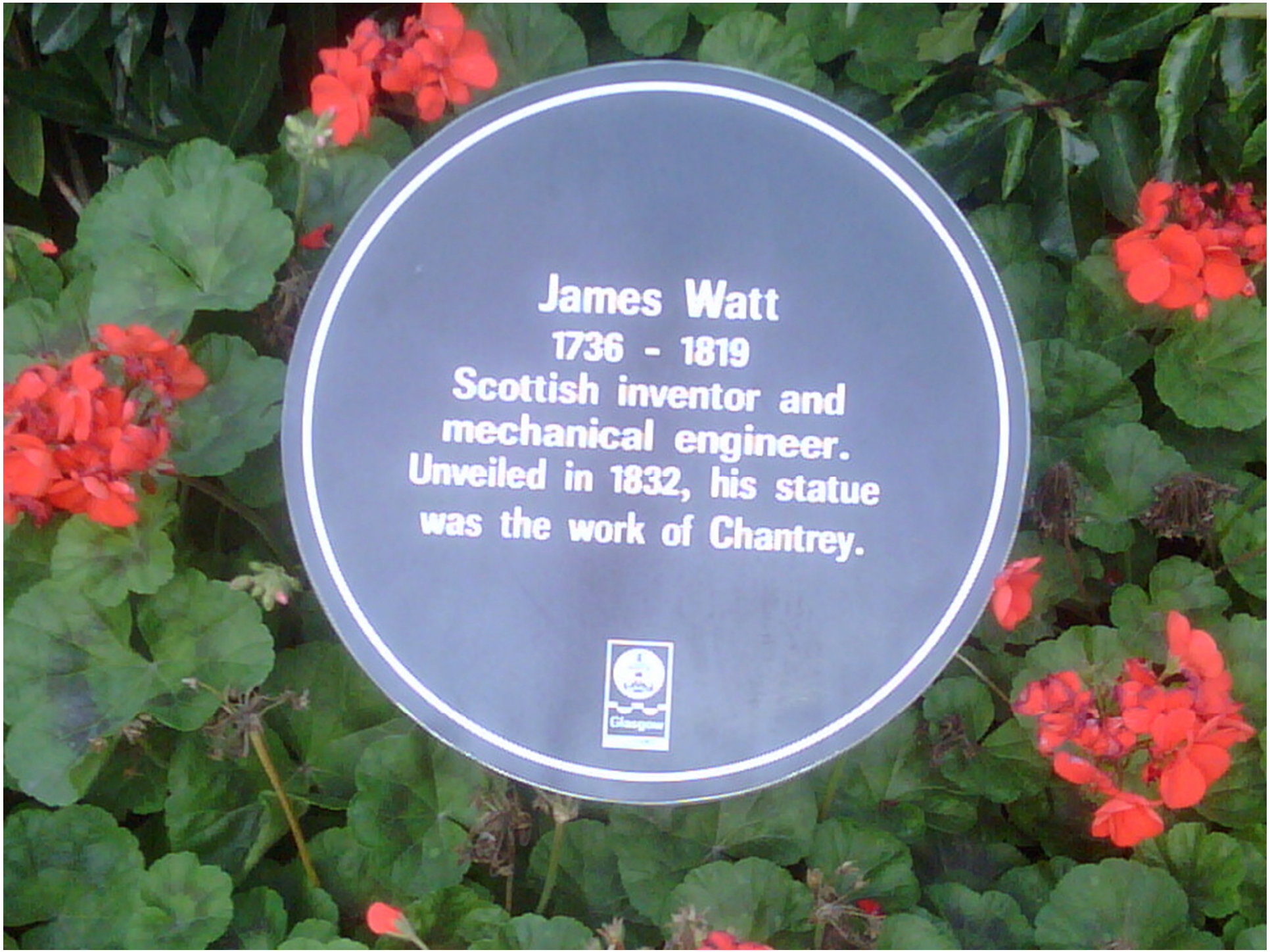
Why do businesses build software ?

To Make Money

Introducing: James Watt



Source: wikipedia.org


James Watt
1736 - 1819
Scottish inventor and
mechanical engineer.
Unveiled in 1832, his statue
was the work of Chantrey.
Glasgow

His greatest

http://en.wikipedia.org/wiki/File:Grazebrook_Beam_Engine.jpg

This is a Boulton & Watt beam blowing engine re-erected on the Dartmouth Circus roundabout, on the A38(M) in Birmingham, UK.
It was built in 1817 and used in Netherton at the ironworks of M W Grazebrook.

Michael
My father made him an offer
he couldn't refuse.
Kay
What was that?

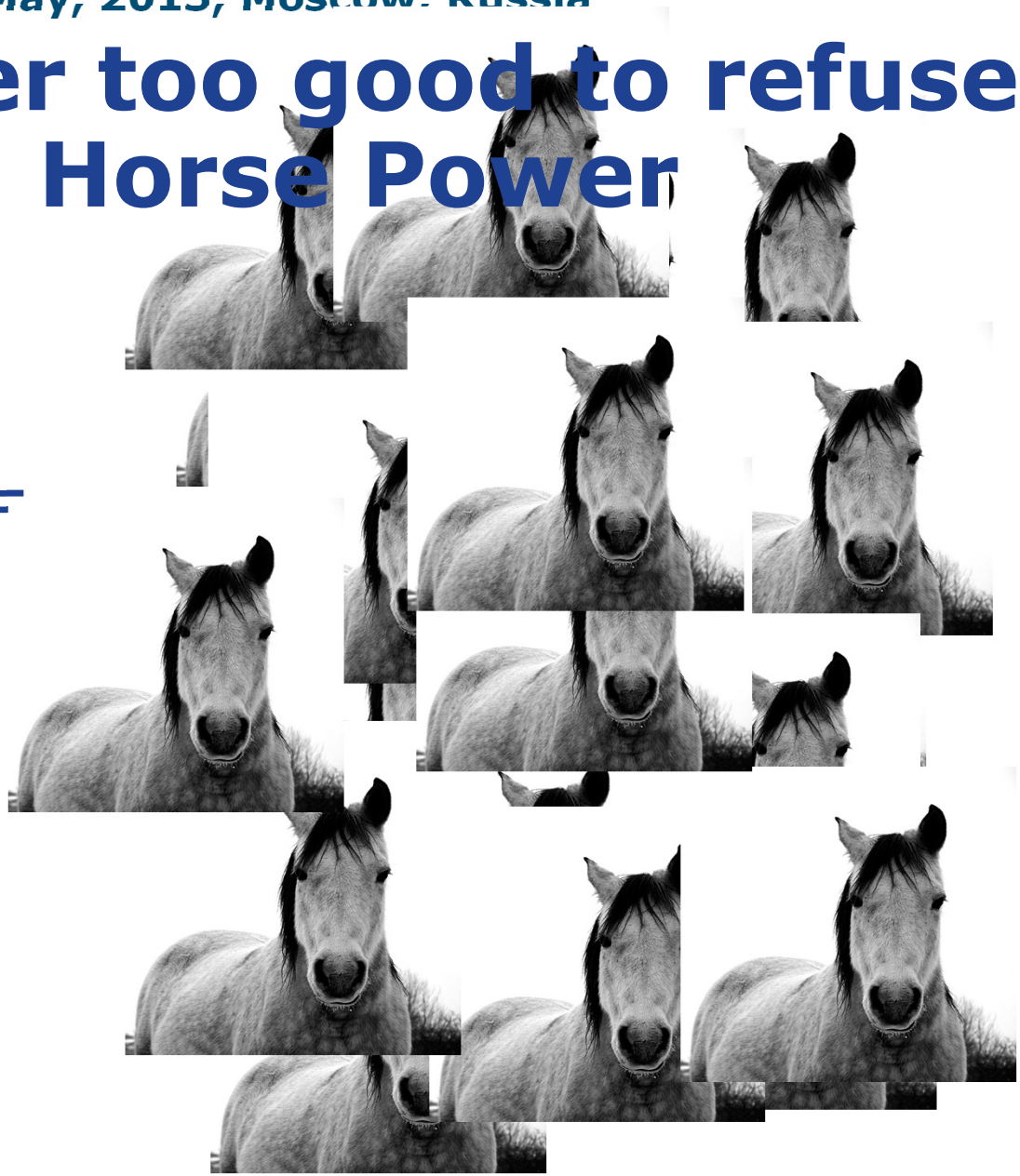
http://en.wikiquote.org/wiki/The_Godfather

www.tocpractice.com

Sixth International TOCPA Conference
18-19 May, 2013, Moscow, Russia

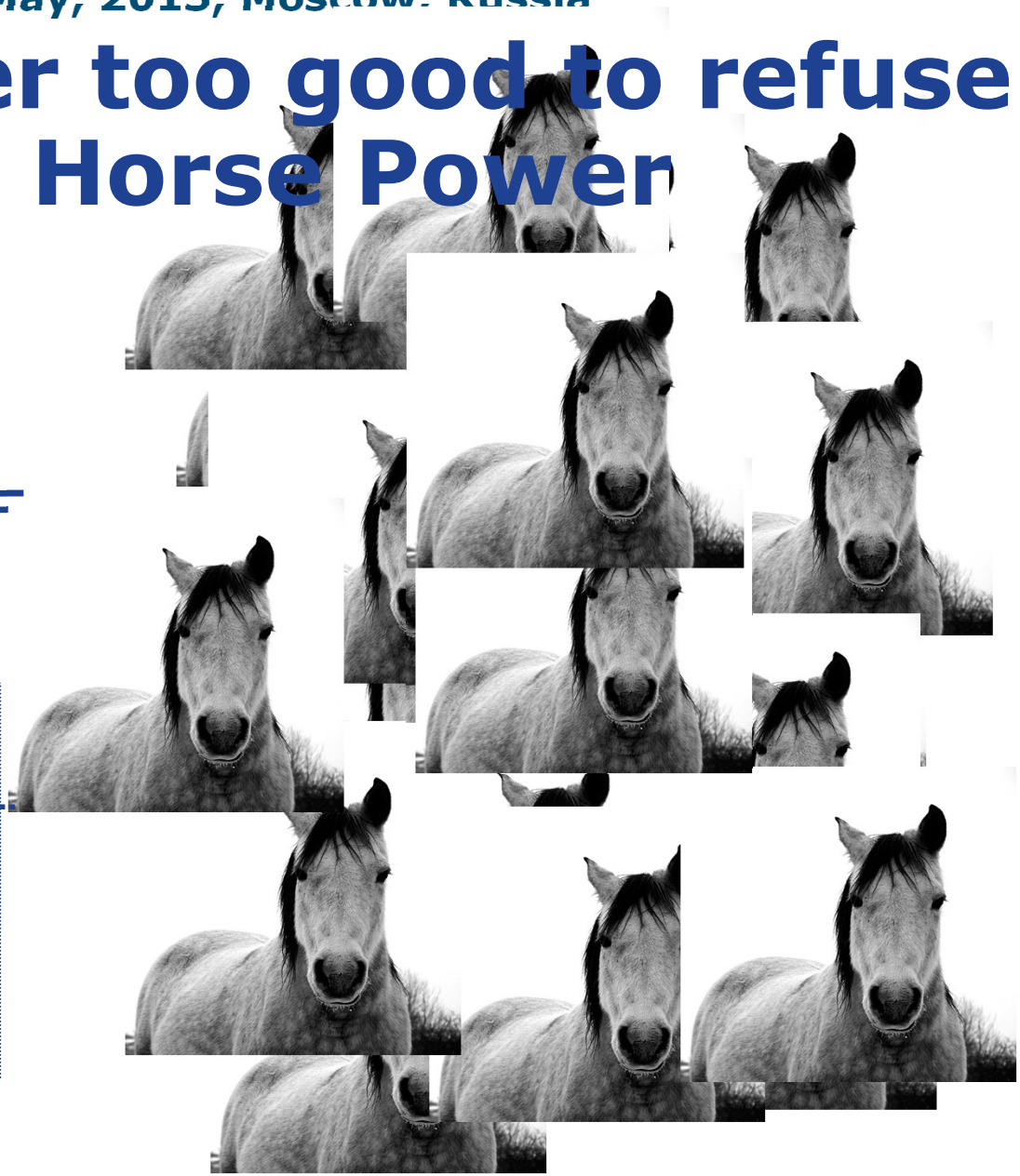
An offer too good to refuse: Horse Power

=



An offer too good to refuse: Horse Power

=

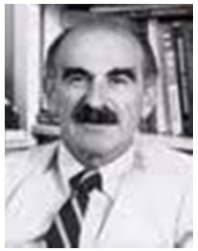


= 30 HP

Which currently costs you £900 p.a..

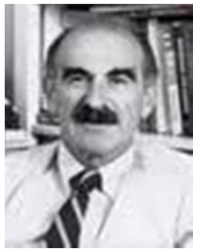
I'll charge you £300 p.a..,
saving you £600 p.a..

If you sign up for 20 years.



"People don't want to buy a quarter-inch drill. They want a quarter-inch hole!"

- Harvard Business School marketing professor Theodore Levitt.



"People don't want to buy a quarter-inch drill. They want a quarter-inch hole!"

- Harvard Business School marketing professor Theodore Levitt.

Agile is our Drill.

More profitable business is our quarter inch hole.

A Happier workplace is the 100inch 3D HD TV we hang from the hole.

1. Snowball effect.

Remember these 3 projects?

Before: AAAABBBBCCCC

After: AAABBBCCC

We sold the extra capacity for half-price & doubled profits.

Remember these 3 projects?

Before: AAAABBBBCCCC

After: AAABBBCCC

If each project makes £100,000 a month *once delivered* then how much more does project A make?

Project B? Project C?

The projects not only finish sooner, they start sooner too.

Just out of interest:

Before: AAAABBBBCCCCD

After: AAABBBCCCDDDE

Notice how D now finishes on month 12, but before it didn't start until month 13?

This is the snowball effect.

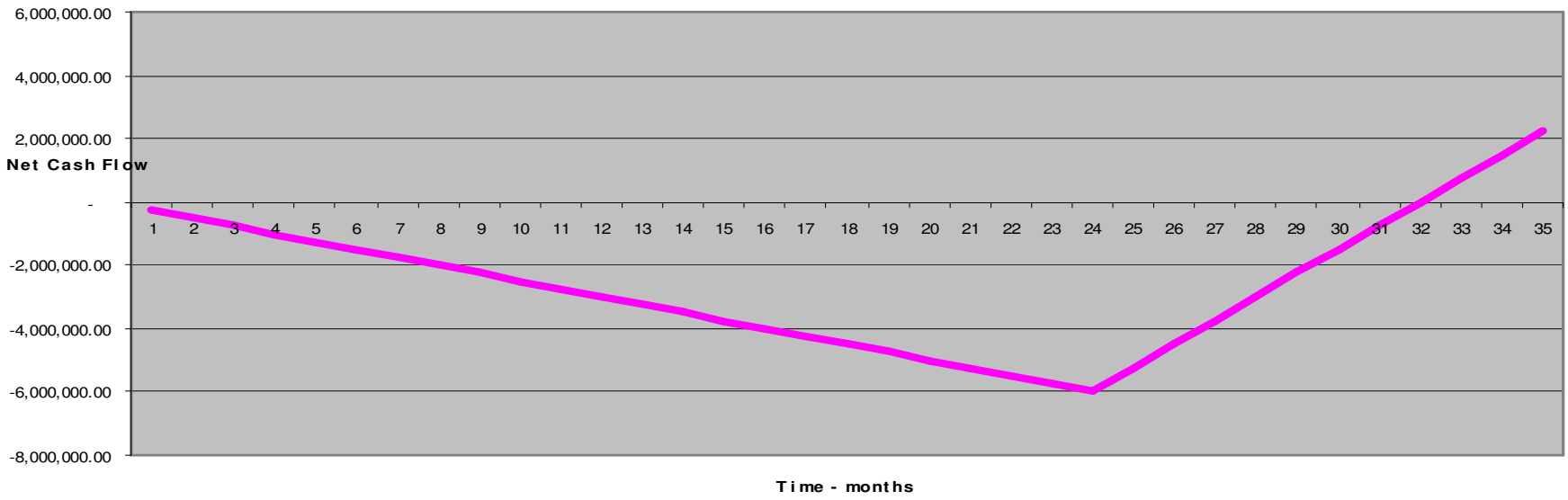
2. Release Early, Release Often, make Cash Flow …

Before: ZZZZZZZZZZZZZZZ

After: AAABBBCCCDDDEEE

What if we released 1 big project as 2 (or more) smaller projects?

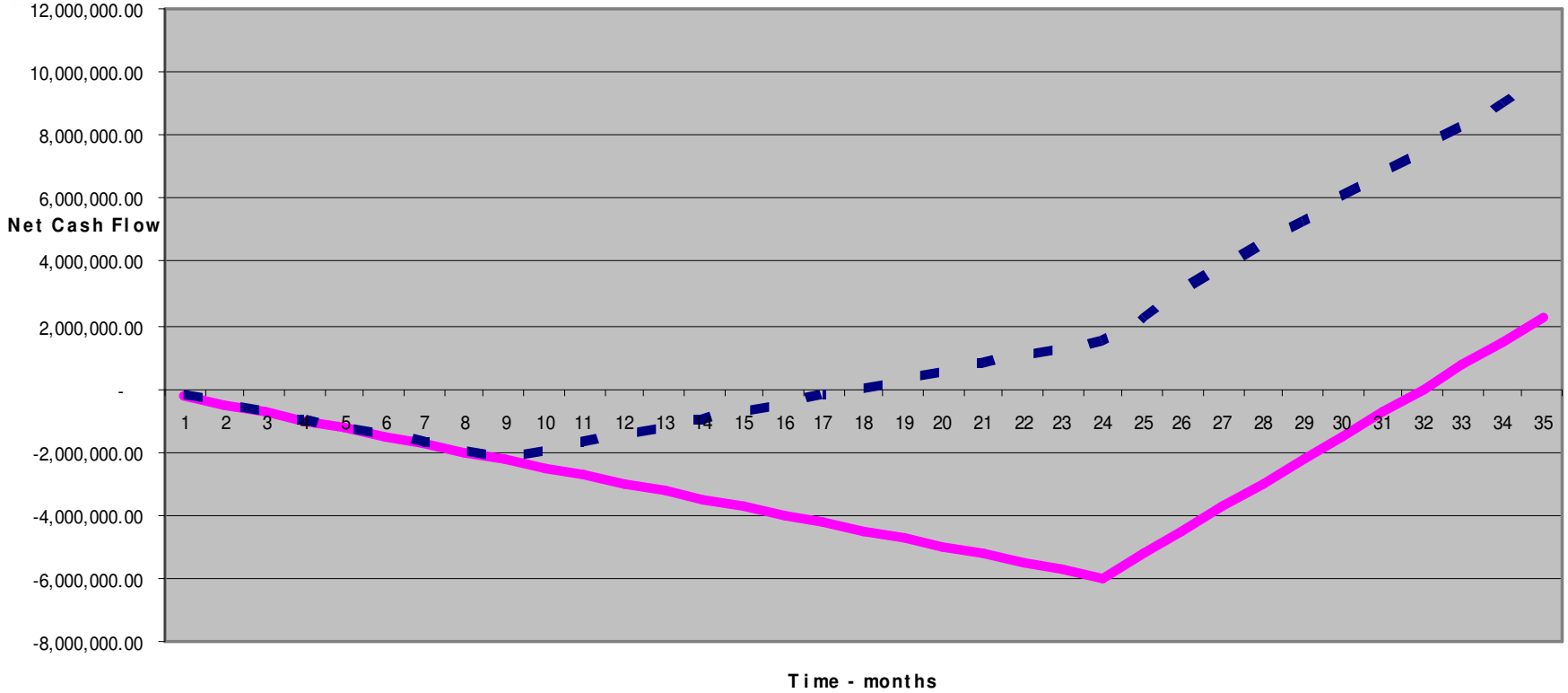
Faster cash flow and higher profits



You : "Single Release"
Revenue, starts month 24,
£1,000,000 / month (£12m / year}

** Both have development costs of £250,000 per month*

Faster cash flow and higher profits

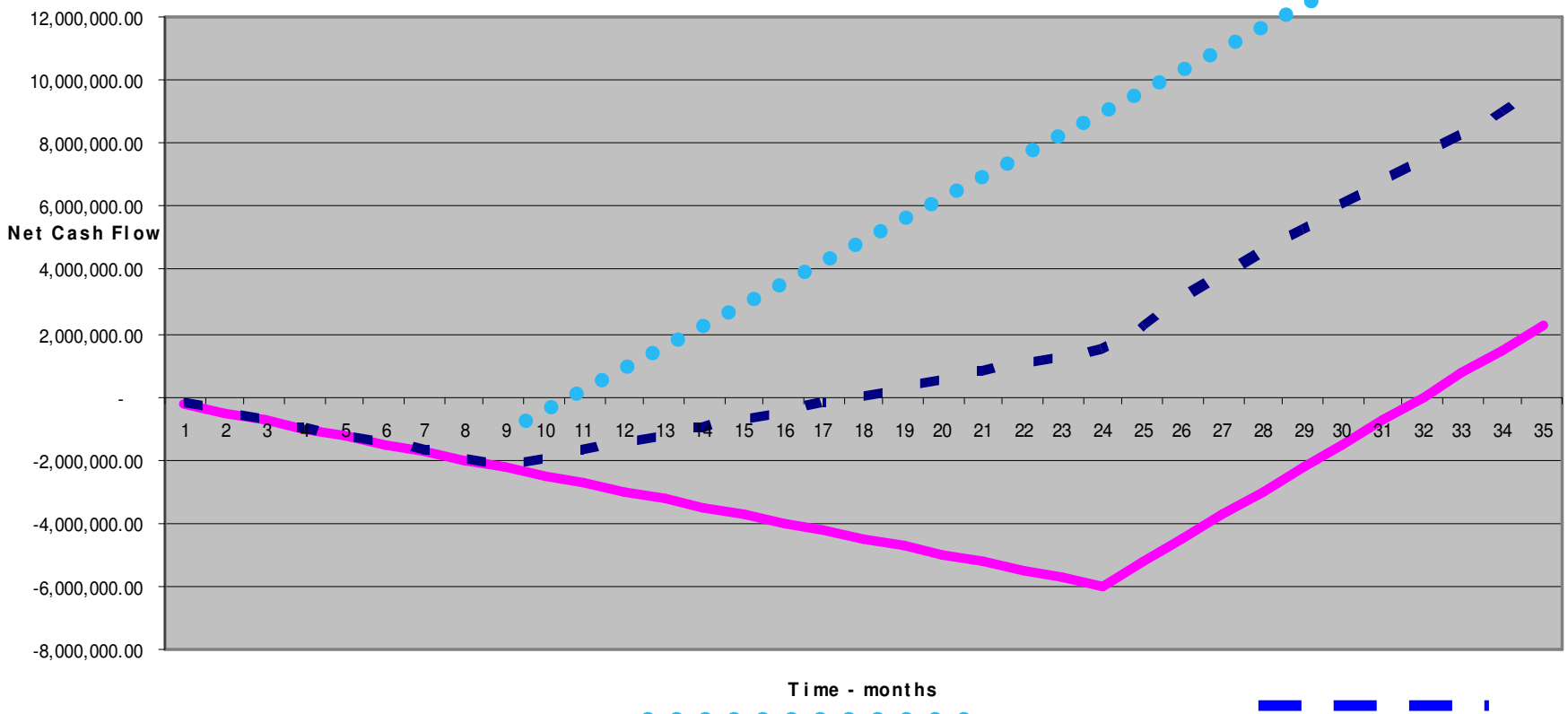


You : "Single Release"
Revenue, starts month 24,
£1,000,000 / month (£12m / year}

Your competitor: "Two releases"
Revenue, starts month 9
£500,000 / month (£6m / year},
Then grows to £12M at month 24

** Both versions have development costs of £250,000 per month*

Faster cash flow and higher profits



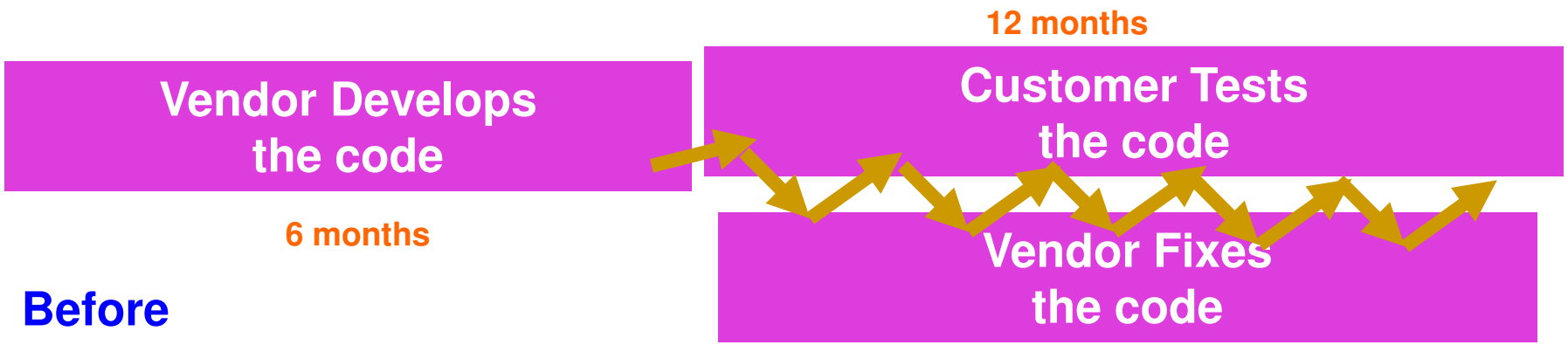
You : "Single Release"
Revenue, starts month 24,
£1,000,000 / month (£12m / year}

More realistic:
Your competitor learns from customer feedback & does multiple releases.

Your competitor: "Two releases"
Revenue, starts month 9
£500,000 / month (£6m / year},
Then grows to £12M at month 24

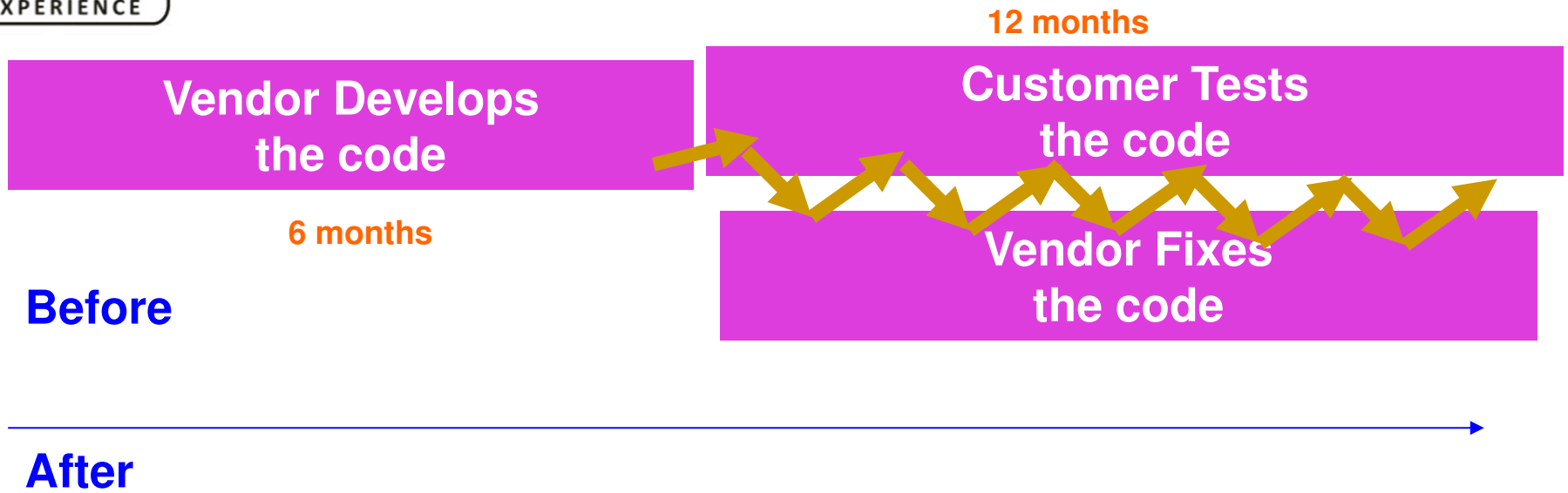
** Both versions have development costs of £250,000 per month*

3. Parallel Testing

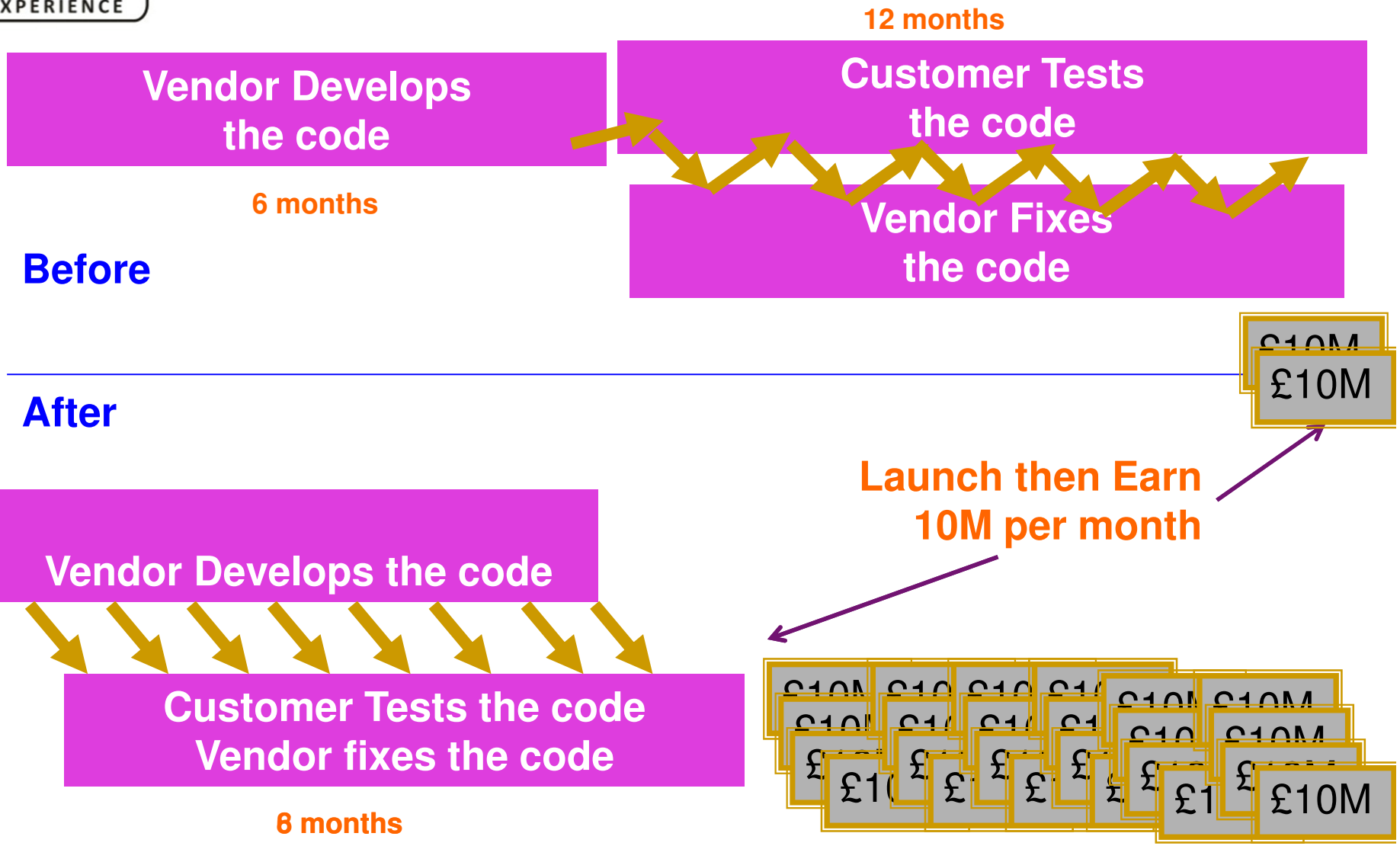


Before

Parallel Testing



Parallel Testing



Sixth International TOCPA Conference
18-19 May, 2013, Moscow, Russia

Agile 101 – Part 2

Kanban and Scrum and other Agile Practices and Techniques

Features are *broken down* into Stories
Stories are *built up* into become Features

Features

- You already know what these are.
- <> tasks or technical things.

Stories

- Requests to build a feature or (more likely) one small slice of a feature.
- A story needs to be small enough it fits into a 2 week sprint.
  - If it's too big for that, it's called an epic.
  - An epic will be broken down into several stories.

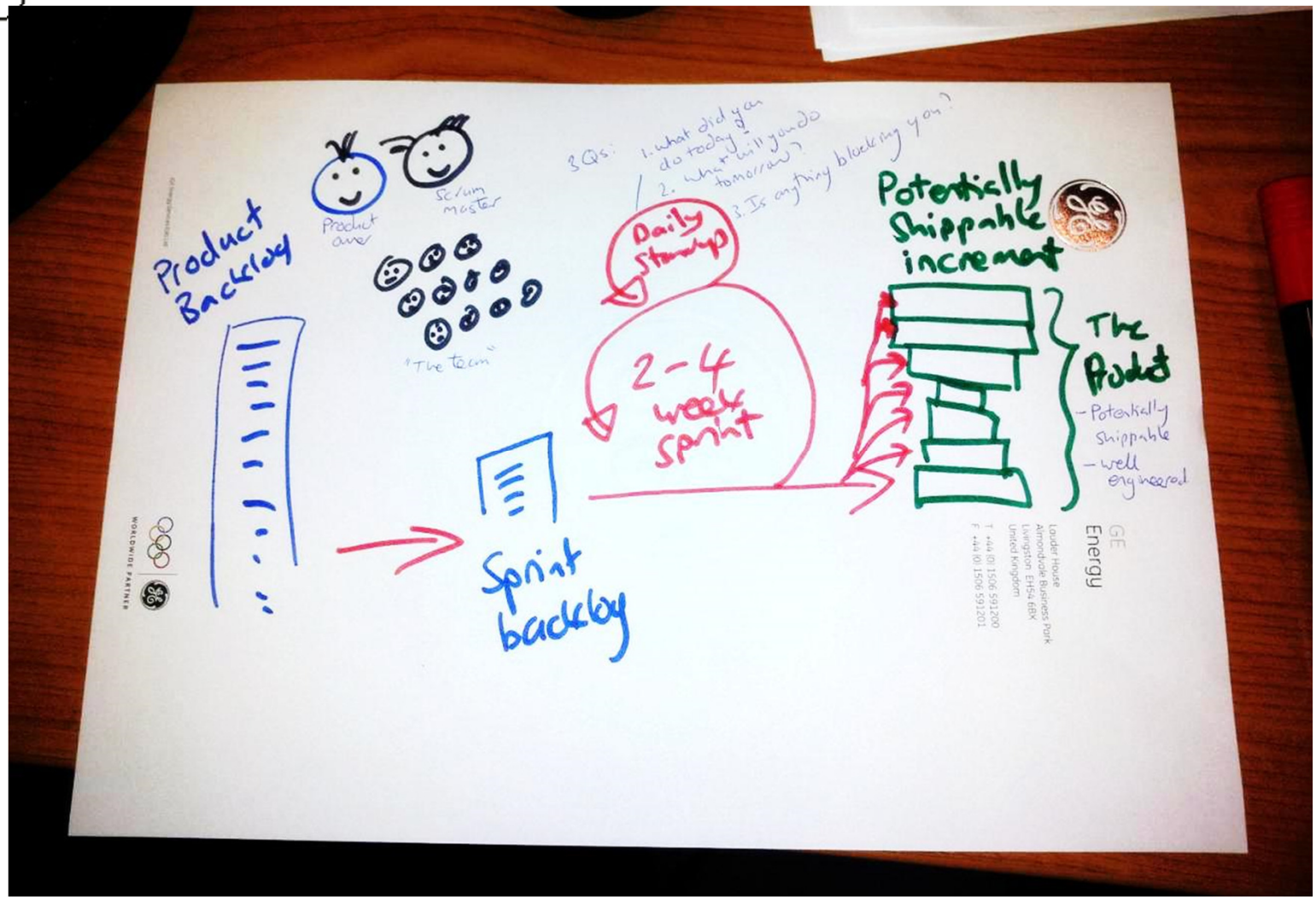
Hopefully:

- Not all stories get built.

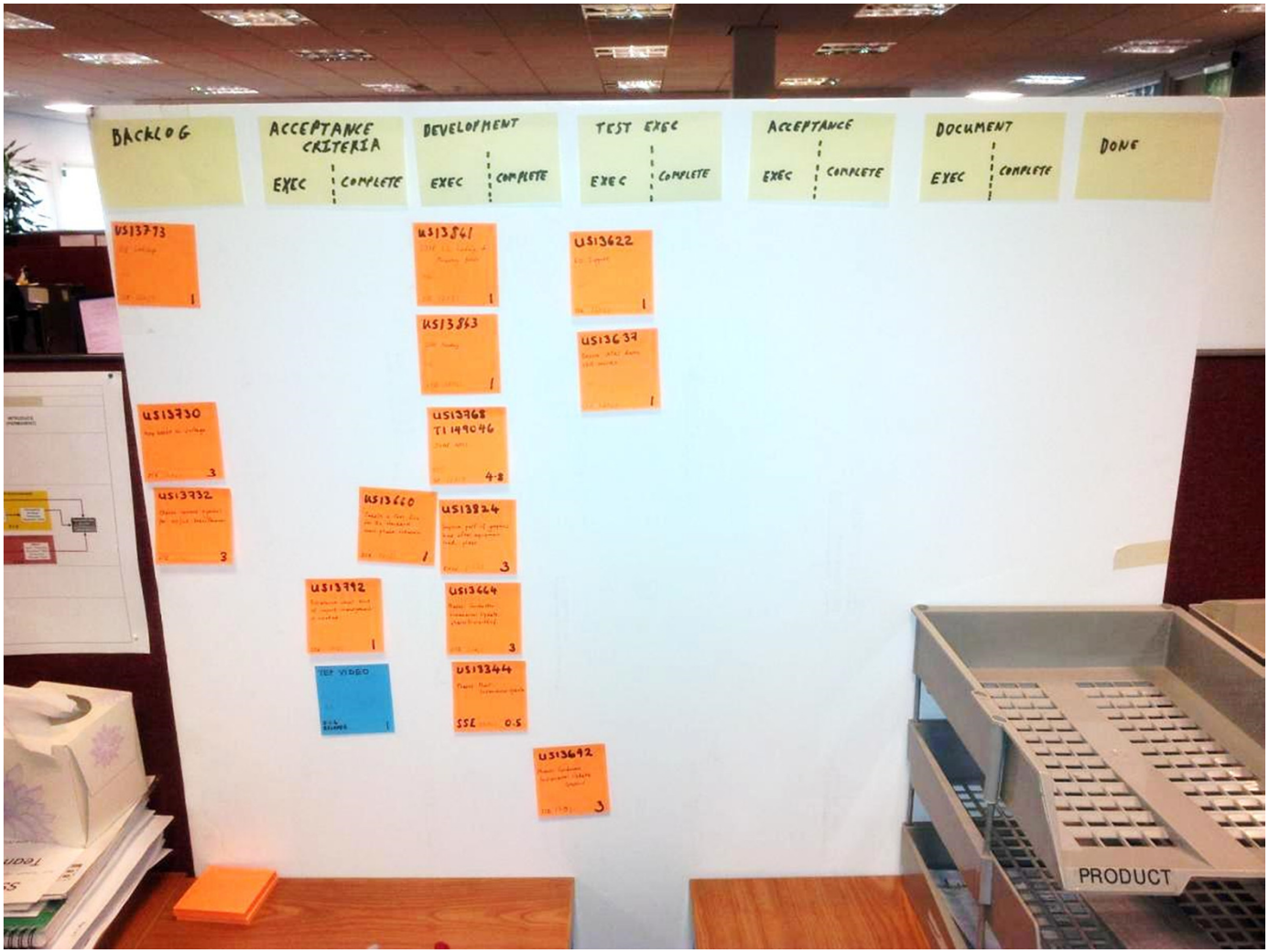
Story 112

As a TV-repair-person
I need to know my next appointment
So that I can arrive on time.

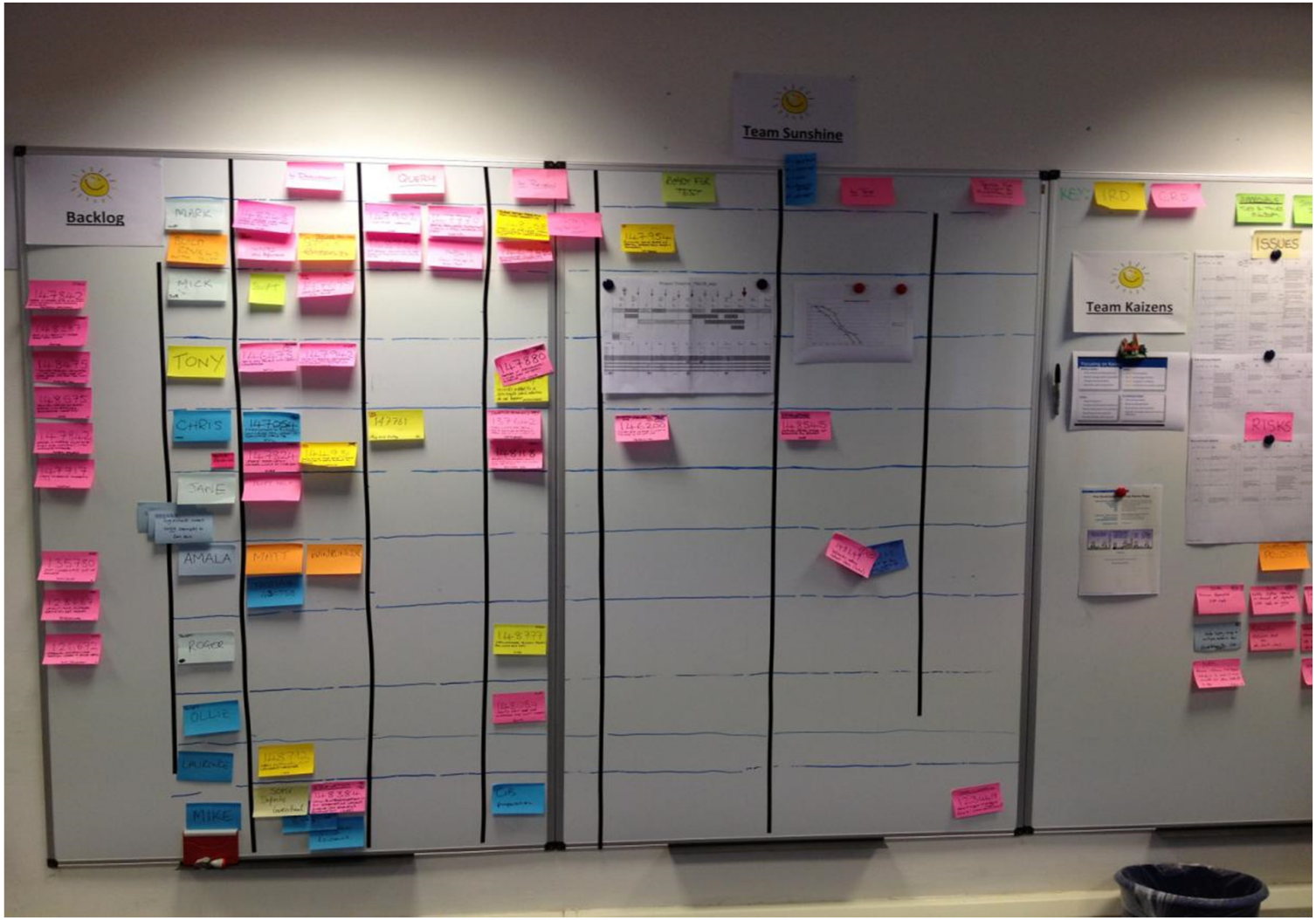
Scrum on a page



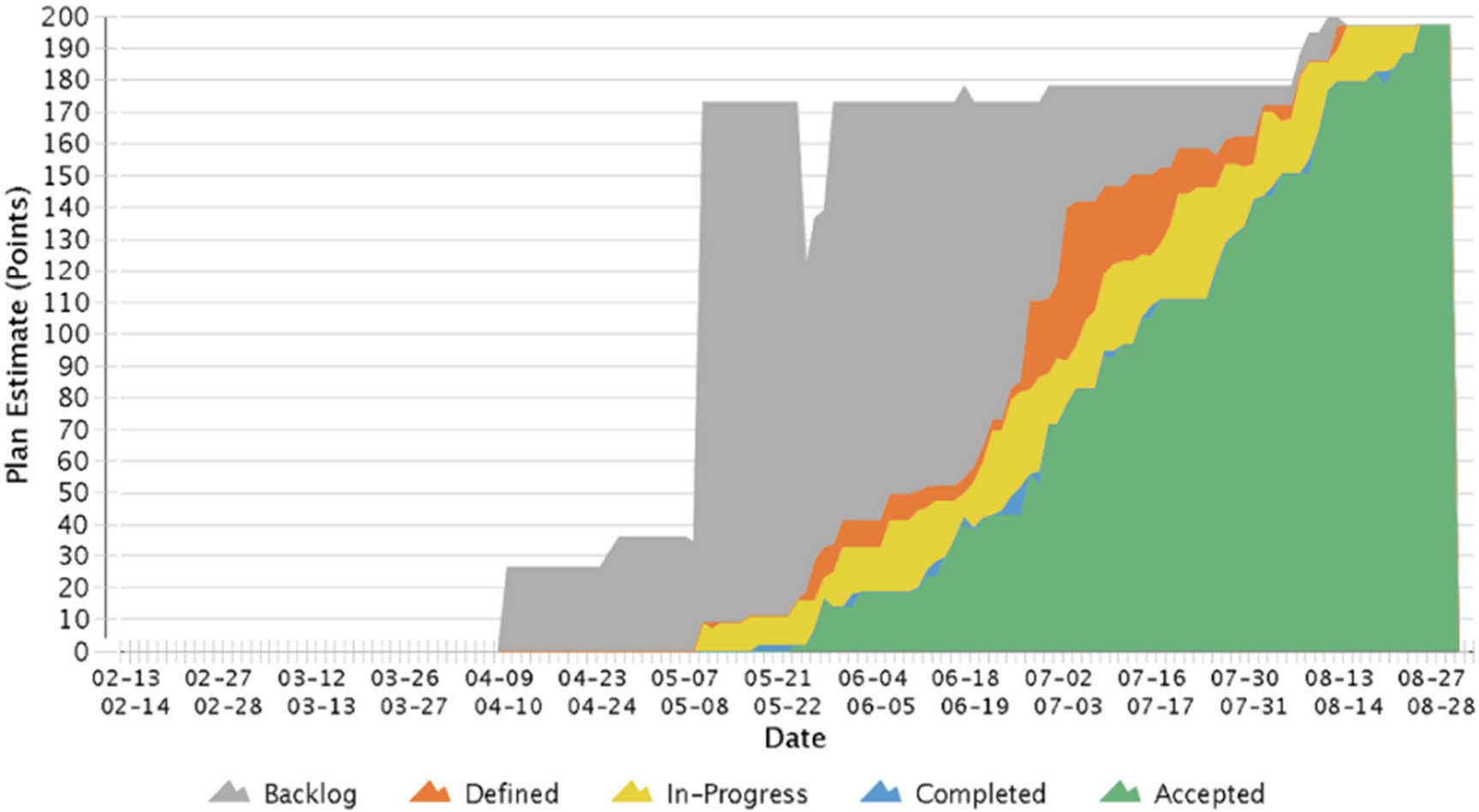
A Scrum board.



A Kanban board



Release Cumulative Flow



Final thoughts:

1. If you're ~~TRULY~~ VERY Agile then you are always ready to ship with a day's ~~week's~~ notice.
(Provided it makes business sense).

2. Agile works but it is hard work.

Questions?