Software Strategy

# Why Quality Must Come First

Allan Kelly
allan@allankelly.net
http://www.allankelly.net
Twitter: allankellynet

Skills Matter - ***In the Brain***
March 2011
60 minutes

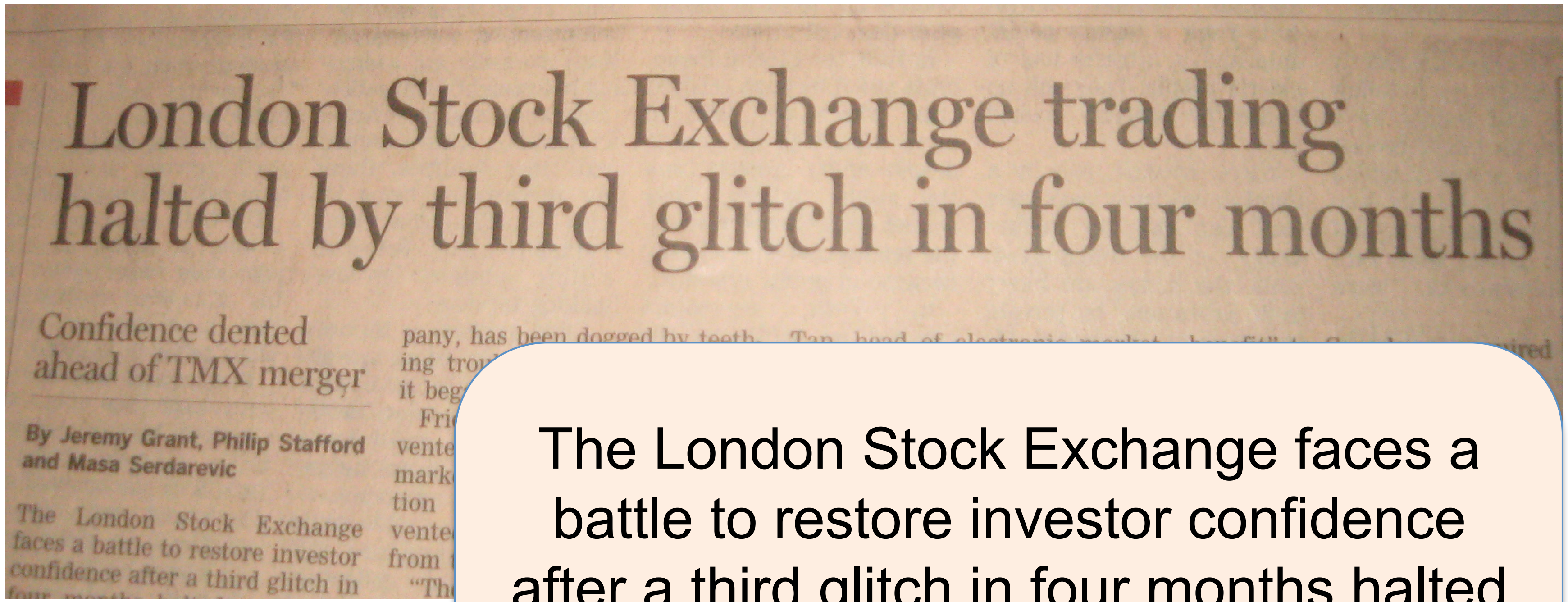
## Allan Kelly

- Training & Coaching for Agile adoption and deepening
- Software company specialist
- Author:
  - ***Changing Software Development: Learning to be Agile***, Wiley 2008.

*97 Things Every Programmer Should Know*, Henney, 2010

*Context Encapsulation* in *Pattern Languages of Program Design* volume 5, 2006

Collective Wisdom from the Experts
97 Things Every Programmer Should Know
O'REILLY



London Stock Exchange trading halted by third glitch in four months

Confidence dented ahead of TMX merger

By Jeremy Grant, Philip Stafford and Masa Serdarevic

The London Stock Exchange faces a battle to restore investor confidence after a third glitch in

The London Stock Exchange faces a battle to restore investor confidence after a third glitch in four months halted trading on Friday morning. …
The system failure came… new system… dogged by teething troubles

Financial Times, 6 February 2011

INSTITUTE FOR GOVERNMENT

SYSTEM ERROR

*Fixing the flaws in government IT*

- Modularity
- An iterative approach
- Responsiveness to change
- Putting users at the core

*Something missing… How do you do this with poor quality?*

## By the numbers

Capers Jones, 2008

*Applied Software Measurement*

But outranking both paper and code, the cost of repairing defects is the most expensive single activity.

For a large project, the cost of producing paper documents is more expensive than the code itself.

Projects with low defect potentials and high defect removal efficiency also have the shortest schedules, lowest costs and best customer satisfaction levels

## Quality, not Qualities

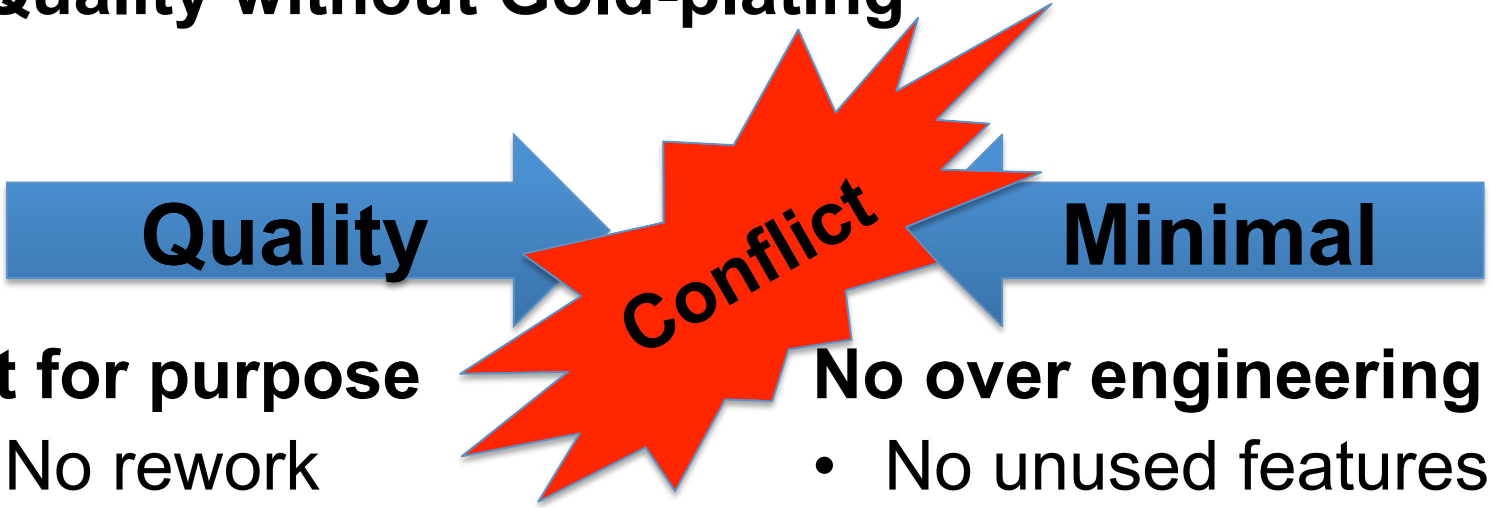
1970's Leyland Mini
•Rusts quickly
•Doesn't start well
•Engine floods
•etc. etc

1970's Rolls Royce
•Spacious
•Leather upholstery
•Low MPG

2000's BMW Mini
•Starts first time
•Engine just works
•Doesn't rust
•Nice to drive

Images from Wikipedia: Rolls-Royce public domain from Bull-Doser; Minis creative commons licenses, DeFacto (Leyland), BMW (IFCAR)

## Quality without Gold-plating



**Fit for purpose**

- No rework
- Free of bugs
- Features which work
- Fewer features make for more usability
- Maintainable
- Knife through butter testing

**No over engineering**

- No unused features
- No “reusable” code
- No “that would be cool”
- No half baked ideas

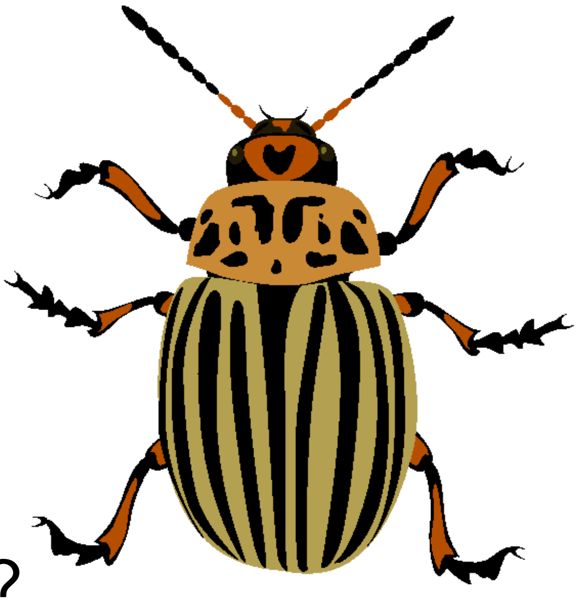
## Can you afford to reusable code?

Single Use

- Break even on third (re)use
- Profit on fourth (re)use
- How much of your code is (re)used four times?

"Reusable" - costs three times more

## Bugs



- How much time do you spend finding bugs?
- How many testers do you need?
- How many bugs do you have logged?
- How many bugs do you fix before shipping?
- How much time do you spend in meetings discussing bugs?

**How would your life change if there were no bugs?**

**Quality is Free$^{TM}$**

- Semi-conductors
- Missiles
- Cars
- Etc. etc.

Philip Crosby
1980

Quality is basis of Lean
Lean is the basis of Agile

Are we in danger of forgetting quality?

## An Old Idea

$\sum$

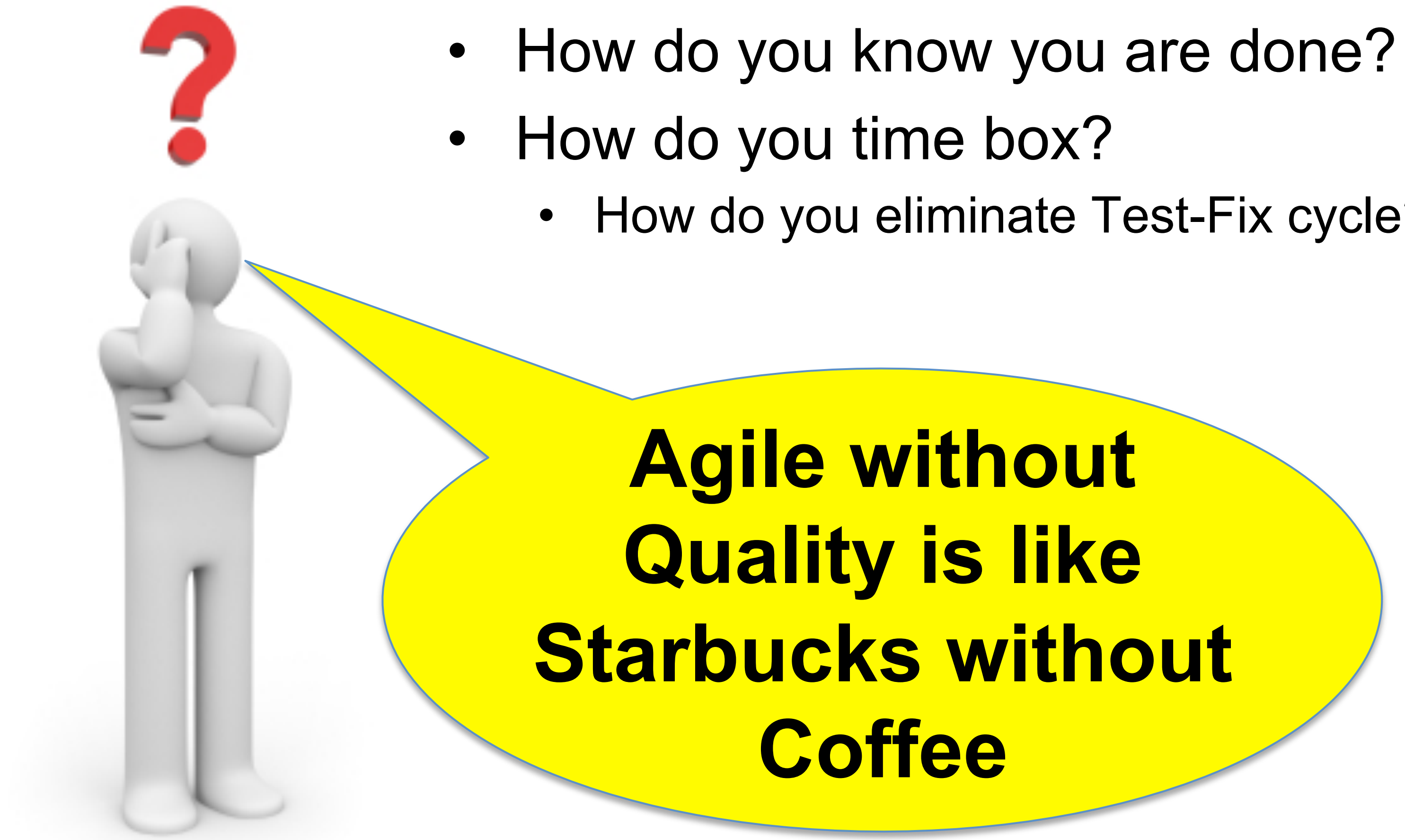
Original work
Finding defect
Scheduling fix
Fixing
Retesting
Customer
    inconvenience
Schedule
    disruption

$>$

Extra work
to prevent it

## Agile without quality?

- How do you know you are done?
- How do you time box?
  - How do you eliminate Test-Fix cycle?



STARBUCKS COFFEE
Careful, the beverage you're about to enjoy is extremely hot.

Starbucks image © Louis Abate, Creative Commons License, c/o Flickr

## Follow the Logic (iterations)

- Without quality you need test-fix
  - With test-fix you can’t close an iteration
- If you can’t close an iteration you can’t be done
  - Thus Iterations (Time-boxes) fall apart
- Without time-boxes delivery becomes random
  - People retreat to plans and demands

*How is this different to the old world?*

## Follow the Logic (design)

- Without quality you need test-fix
  - With test-fix you practice Refactoring
  - (Too expensive, too slow)
- Without Refactoring emergent design fails
  - Quality falls
  - More dependence on Big Up Front Design (BUFD)
- BUFD needs Big Up Front Requirements (BUFR)
  - BUFR prevents changes
- Without change, Agile is Not Agile

## Old idea – why didn't it work?

Σ

Original work
Finding defect
Scheduling fix
Fixing
Retesting
Customer inconvenience
Schedule disruption

<

Extra work to prevent defects

Old solutions made defect prevention <u>very expensive</u>

Copious documentation, Heavy weight code reviews, Manual testing

And <u>very very slow</u>

Detracted from ability to respond (Agility)

## Old Idea, New Tools

- Invest in quality
- Make defect prevention cheap
    - Continuous integration
    - Virgin install
    - Test Driven Development
    - Acceptance Test Driven Development
    - Lightweight code-reviews
    - Pair programming
    - Static analysis tools

$$\left.\begin{array}{c}\\ \\ \\ \\ \\ \\ \\ \end{array}\right\}$$ Feedback based

## Unit Testing on Steroids

Automated TDD is to Traditional Unit Testing what Amazon is to Great Universal Stores

## TDD works!

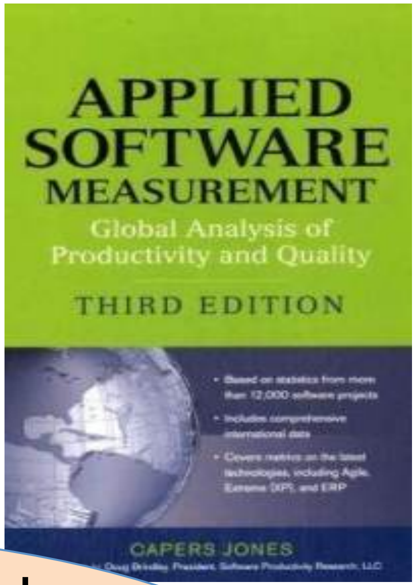
| | IBM drivers | Microsoft Windows | Microsoft MSN | Microsoft Visual Studio |
|---|---|---|---|---|
| Defect density (non-TDD) | W | X | Y | Z |
| Defect density (with TDD) | 61% of W | 38% of W | 24% of Y | 9% of Z |
| Increased time (with TDD) | 15-20% | 25-25% | 15% | 25-20% |

Nagappan, Maximilien, Bhat and Williams (Microsoft Research, IBM Research, North Carolina State University). Empirical Software Engineering journal 2008
http://research.microsoft.com/en-us/projects/esm/nagappan_tdd.pdf

## Code reviews

Capers Jones, 2008

*Applied Software Measurement*


APPLIED SOFTWARE MEASUREMENT
Global Analysis of Productivity and Quality
THIRD EDITION
CAPERS JONES

Formal reviews and inspections have the highest defect removal efficiency levels of any known kind of quality control activity

and are characteristic of "best in class" organizations

## Reprogram the mental model

Too many believe quality is negotiable

A bug here and a bug there, it soon adds up

**Quality is non-negotiable**

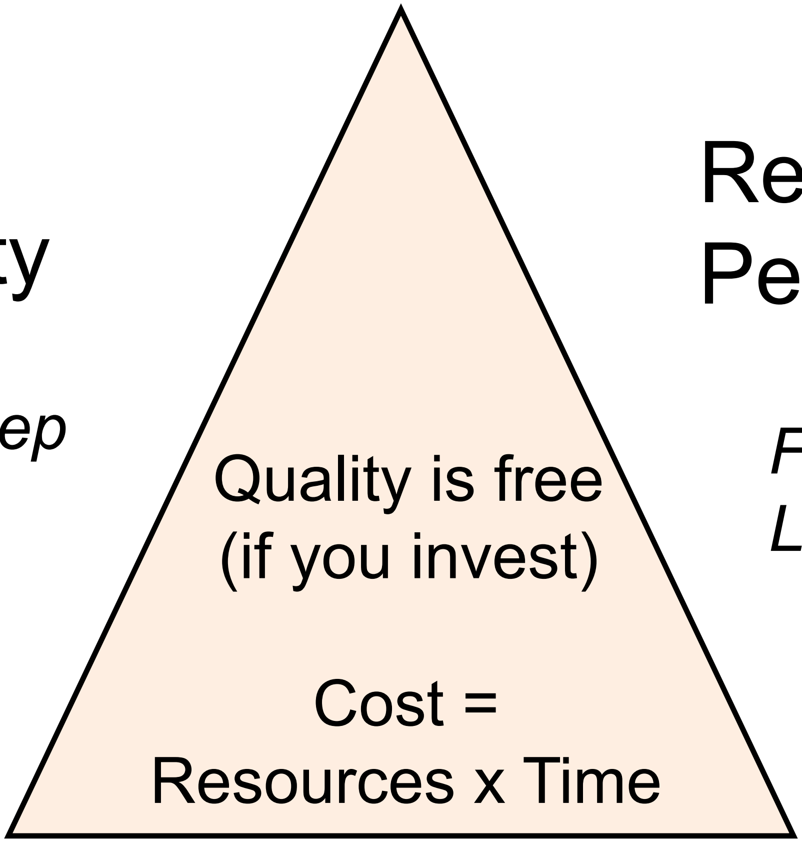
(if you want to be Agile)

Features &
Functionality

*Run scope creep backwards*

Resources –
People!

*Fixed – Brook's Law*

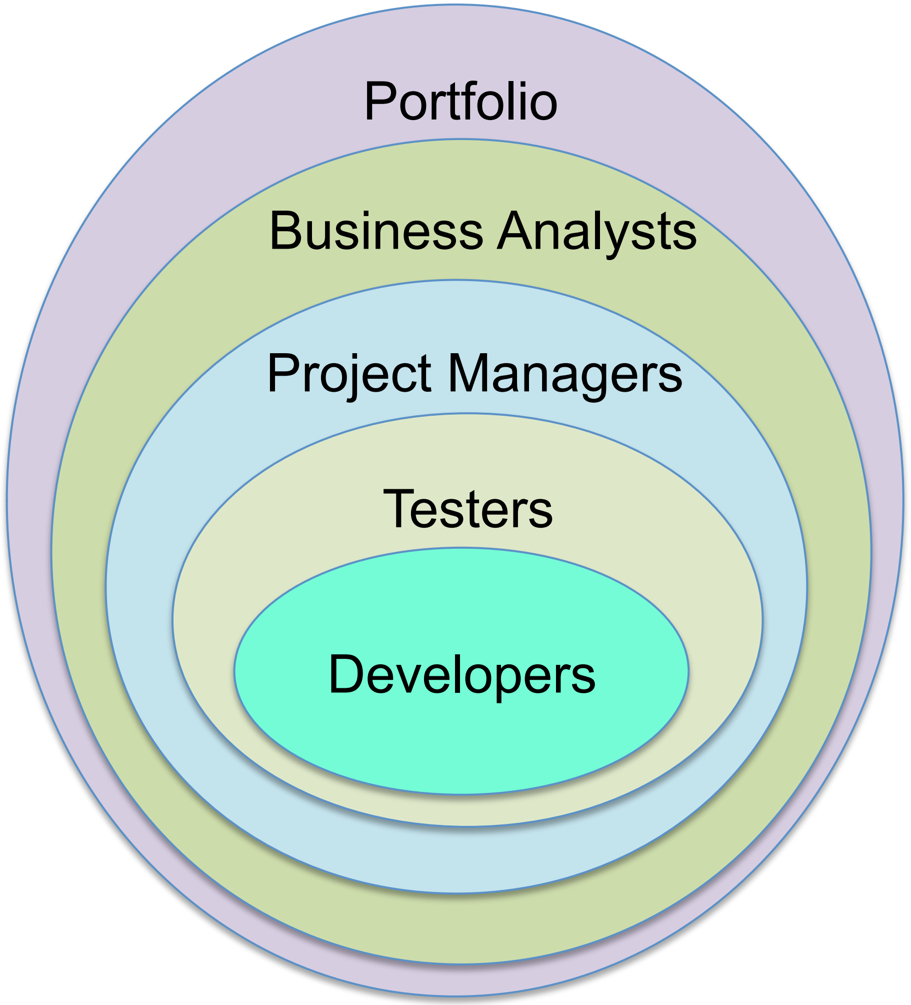


Time

*Time boxed*

2
1

## Agile change model



Drive quality

1 Interest Developers
- Improve quality

2 Enroll testers

3 Refocus Project Managers
- Deliveries over plans

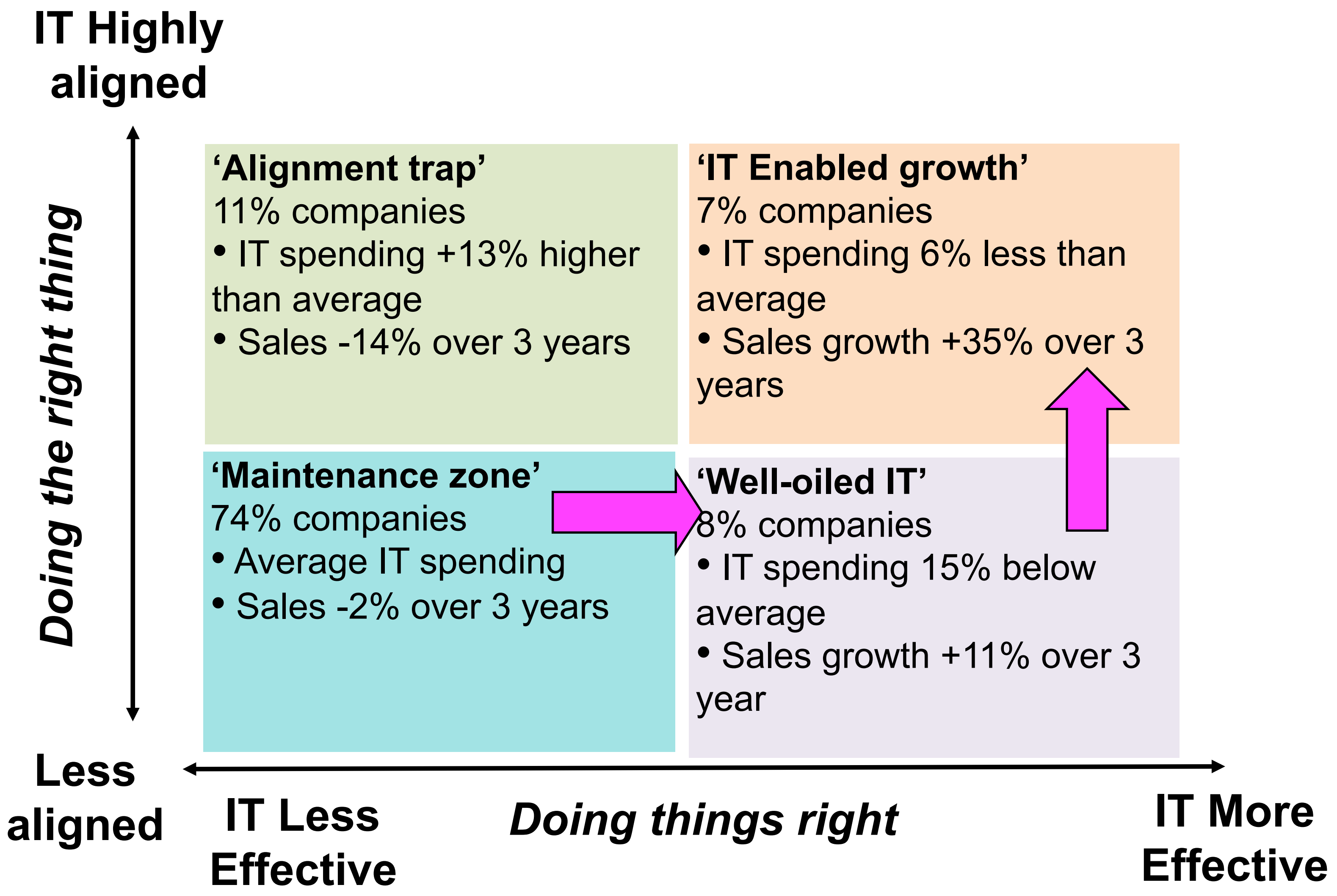
4 Change Business Analysis
- Goals over shopping lists

5 Change Portfolio parameters
- Delivering value over following plan

6. Realign Project Managers

Source: Shpilberg, Berez, Puryear, Shah: MIT Sloan Review, Fall 2007

## The Alignment Trap



**'Alignment trap'**
11% companies
- IT spending +13% higher than average
- Sales -14% over 3 years

**'IT Enabled growth'**
7% companies
- IT spending 6% less than average
- Sales growth +35% over 3 years

**'Maintenance zone'**
74% companies
- Average IT spending
- Sales -2% over 3 years

**'Well-oiled IT'**
8% companies
- IT spending 15% below average
- Sales growth +11% over 3 year

## *Can we build it?*

Job#1

- Improve quality
- Build an effective delivery machine

Job #2

- Move outwards and upwards

## How much quality can we afford?

- Lots
- Quality is free
  - If you invest in it

***Thank you!***
[allan@allankelly.net](mailto:allan@allankelly.net)
[http://www.allankelly.net](http://www.allankelly.net)
Twitter: allankellynet

***Questions?***