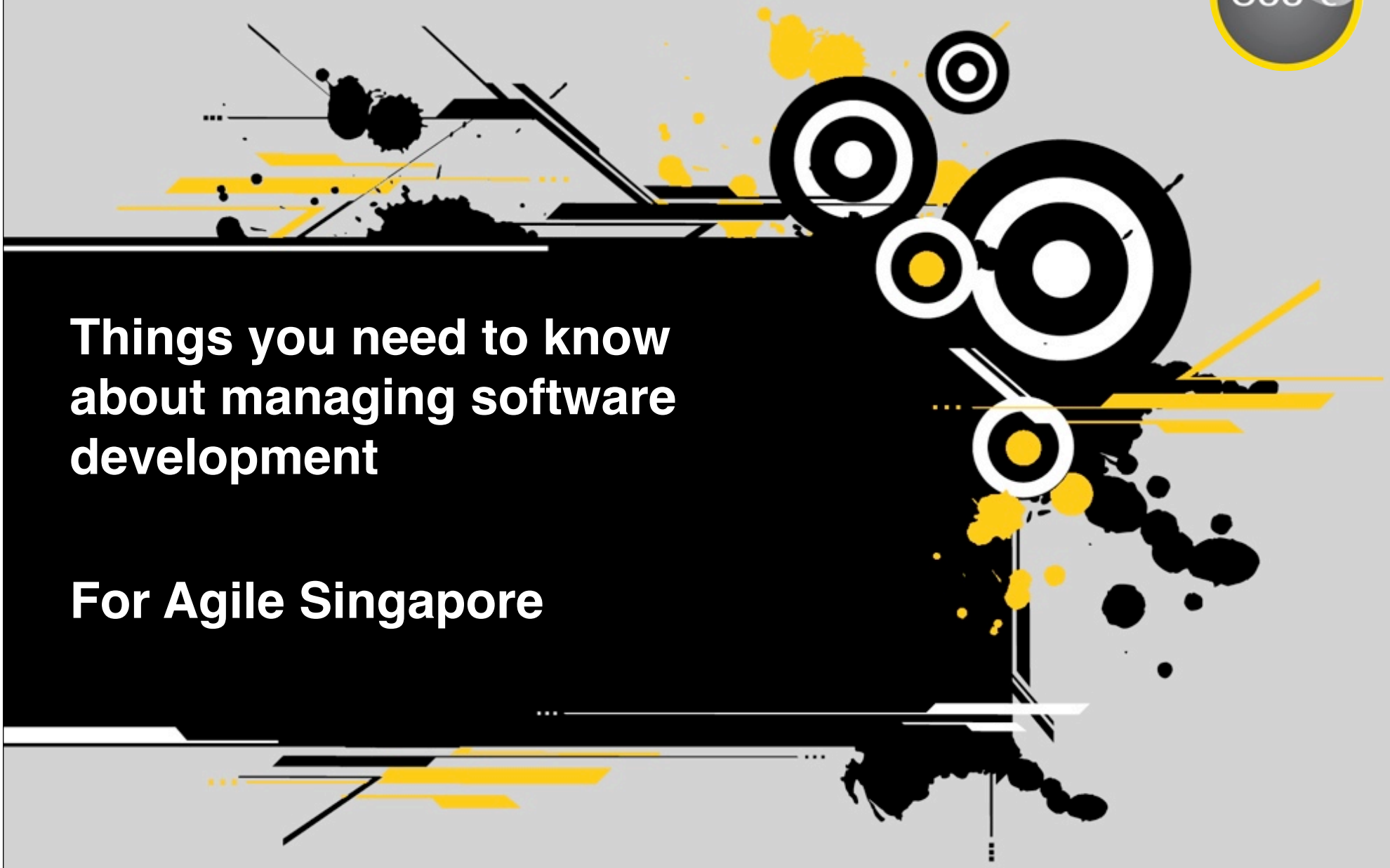
www.odd-e.com

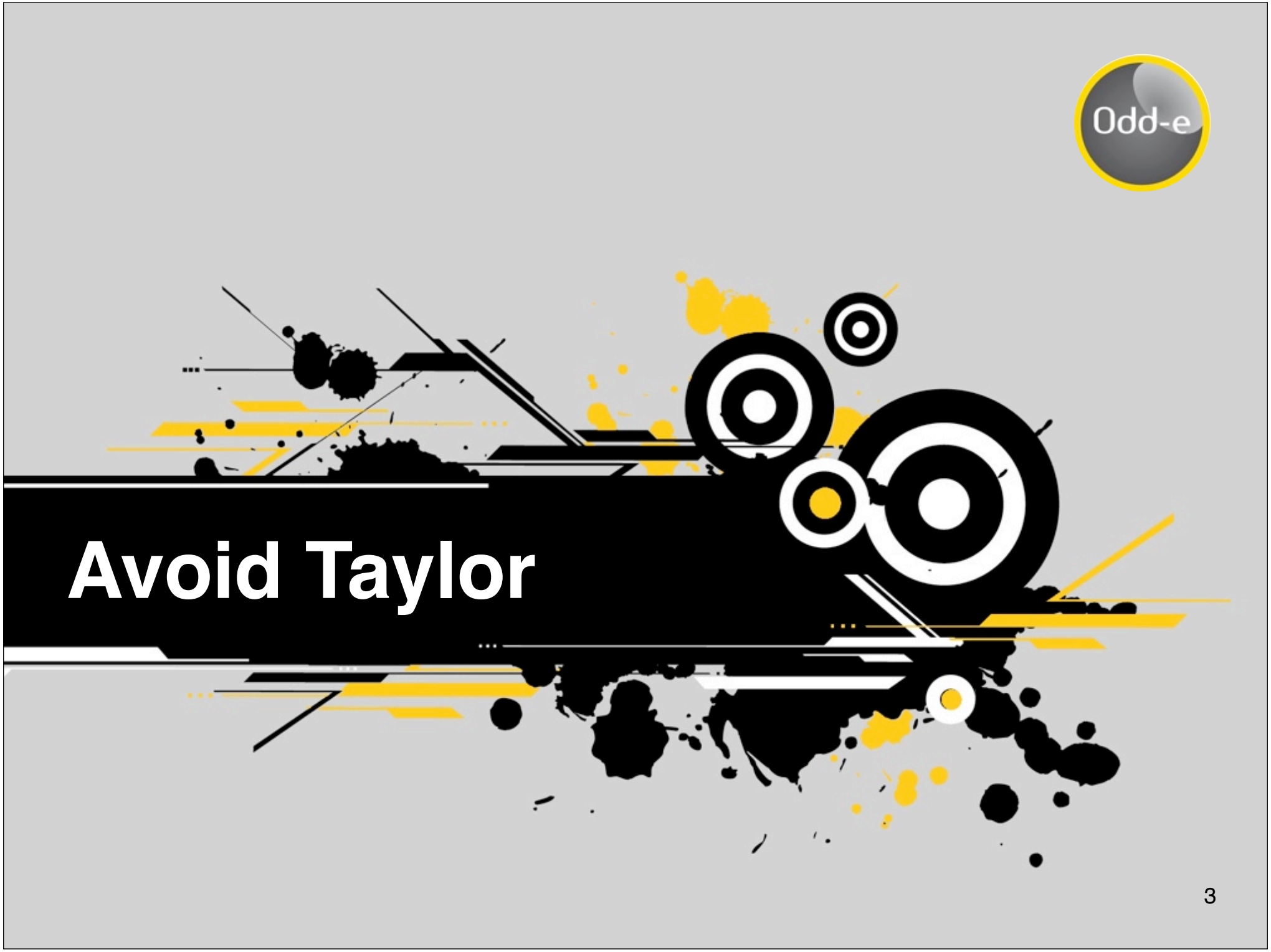


Things you need to know about managing software development

For Agile Singapore

Who am I?

- Name: Bas Vodde
- Originally from Holland
- Lives in Singapore
  - Lived in China and Finland
- Works for Odd-e
- Agile coach, SW developer
- Experience with large embedded products


Odd-e
Avoid Taylor

Konosuke Matsushita (1)

"We will win and you will lose. You cannot do anything about it because your failure is an internal disease. Your companies are based on Taylor's principles. Worse, your heads are Taylorized, too. You firmly believe that sound management means executives on one side and workers on the other, on one side men who think and on the other side men who can only work. For you, management is the art of smoothly transferring the executives' ideas to the workers' hands."

Konosuke Matsushita (2)

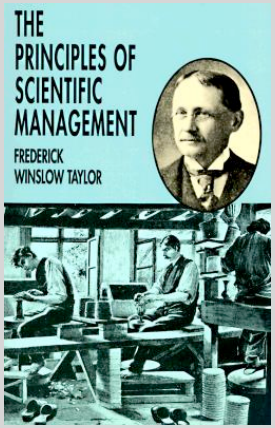
“We have passed the Taylor stage. We are aware that business has become terribly complex. Survival is very uncertain. Therefore, a company must have the constant commitment of the minds of all of its employees to survive. For us, management is the entire workforce's intellectual commitment at the service of the company.

We know that the intelligence of a few technocrats—even very bright ones—has become totally inadequate to face these challenges. Only the intellects of all employees can permit a company to live with the ups and downs and the requirements of its new environment. Yes, we will win and you will lose. For you are not able to rid your minds of the obsolete Taylorisms that we never had.”

Panasonic

There is **no question** that the cost of production is lowered by **separating** the work of **planning** and the **brain work** as much as possible from the **manual labor**

Scientific Management

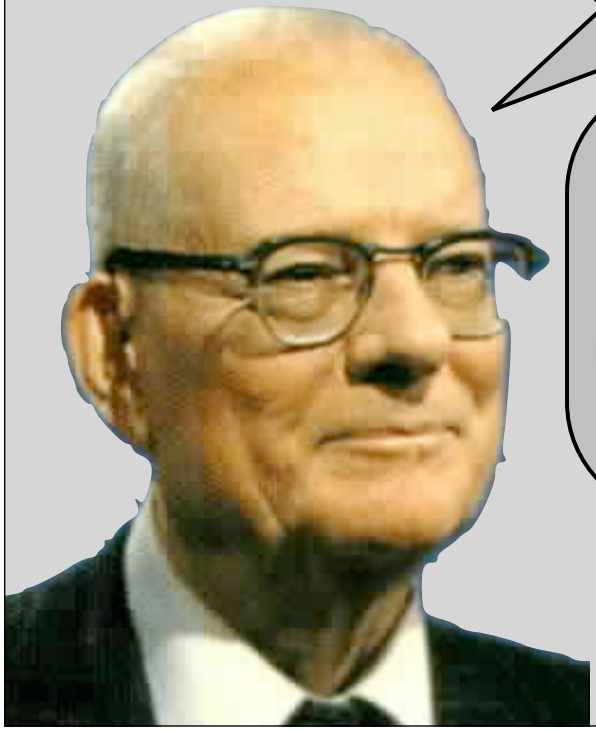


- Application of Science to find “one best method”
- Separation of “‘planning/improving” and execution

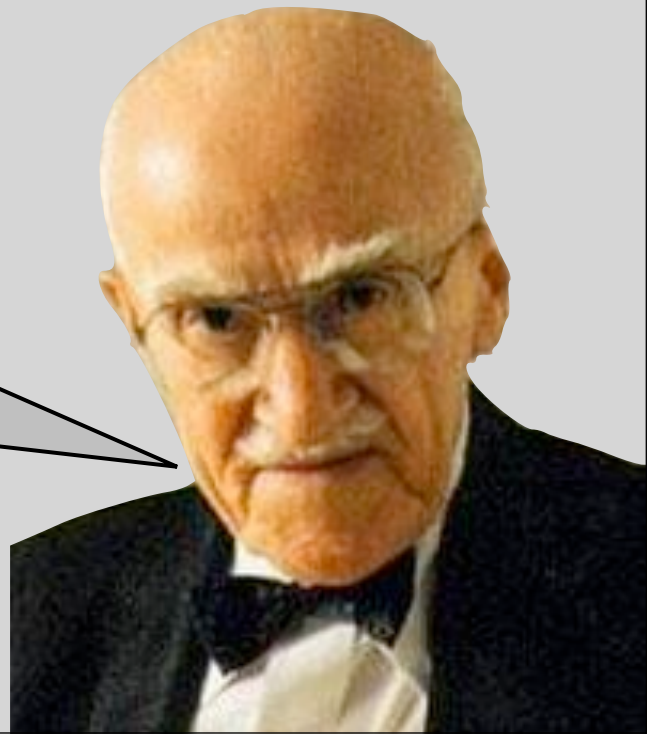
- Strongly influenced existing management practices:
  - Project Management
  - Management by Objectives
  - Incentive systems
  - Sig Sigma / CMMi

Deming / Juran

Remove barriers that rob people of their right to pride of workmanship

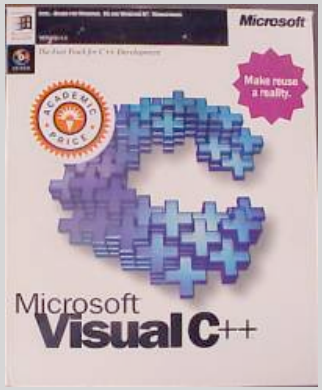


The Taylor System has become so obsolete that is should be replaced.




Odd-e
TEAM = PRODUCT

Jim McCarty



Anything you need to know about the team can be discovered by examining the product, and visa versa

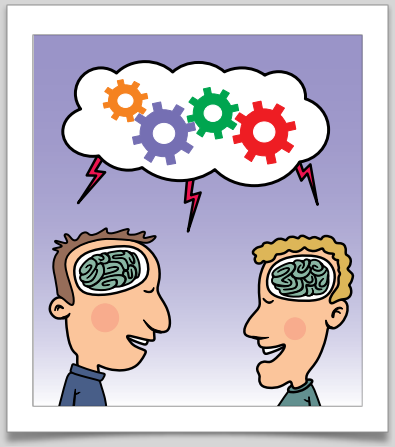
TEAM = PRODUCT

http://www.mccarthyshow.com/the-23-rules-of-thumb/

Team = Product

- Focus on improving people!
  - Working together, sharing work!
  - Books, movies, learning sessions.
  - Training, coaching.

- More than on:
  - Process (let the people do that!)
  - Metrics... also for people themselves



Avoid adding people!

People matter *most*!

Between fastest and slowest

28x

1

Original 1968 Sackman study

Between fastest and slowest

14x

1

Corrected Sackman study (2000)

Common difference

5x

1

Boehm study (1975)

Between top quarter and bottom quarter

4x

1

Prechelt (2000)

For Software teams. This difference is even bigger

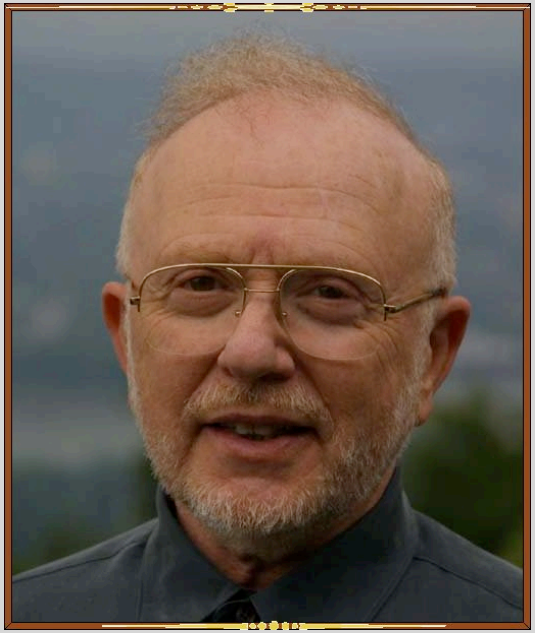
Joel Spolsky

It's not just a matter of "10 times more productive." It's that the "average productive" developer never hits the high notes that make great software.

http://www.joelonsoftware.com/articles/HighNotes.html

David Parnas

What is the most often-overlooked risk in software engineering?



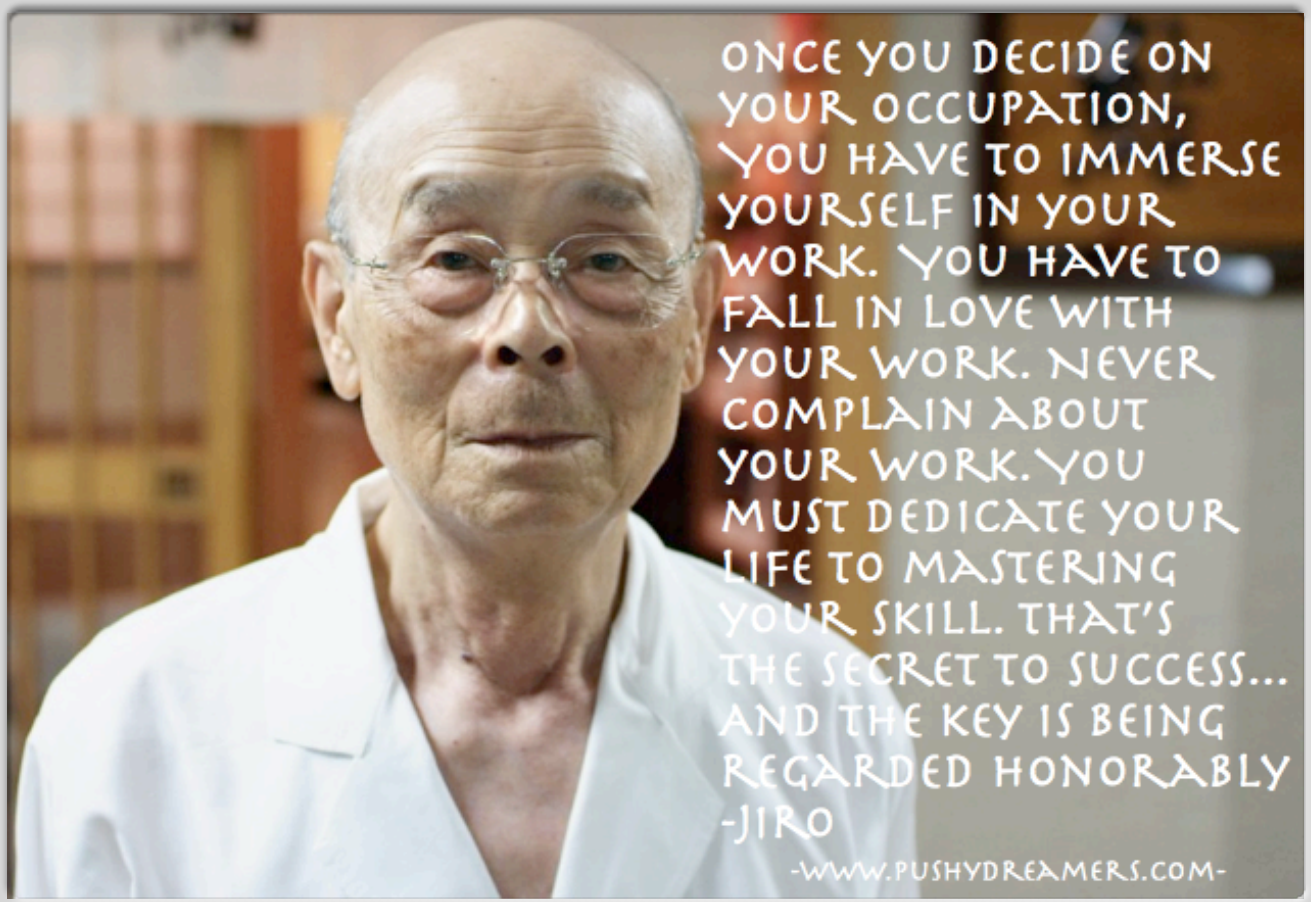
Incompetent programmers. There are estimates that the number of programmers needed in the U.S. exceeds 200,000. This is entirely misleading. It is not a quantity problem; we have a quality problem. **One bad programmer can easily create two new jobs a year**. Hiring more bad programmers will just increase our perceived need for them.

If we had more good programmers, and could easily identify them, we would need fewer, not more.

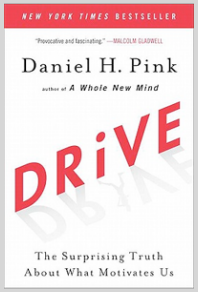
Understand: Motivation

Autonomy

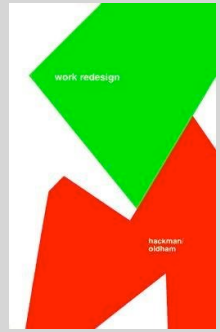
Purpose



Mastery

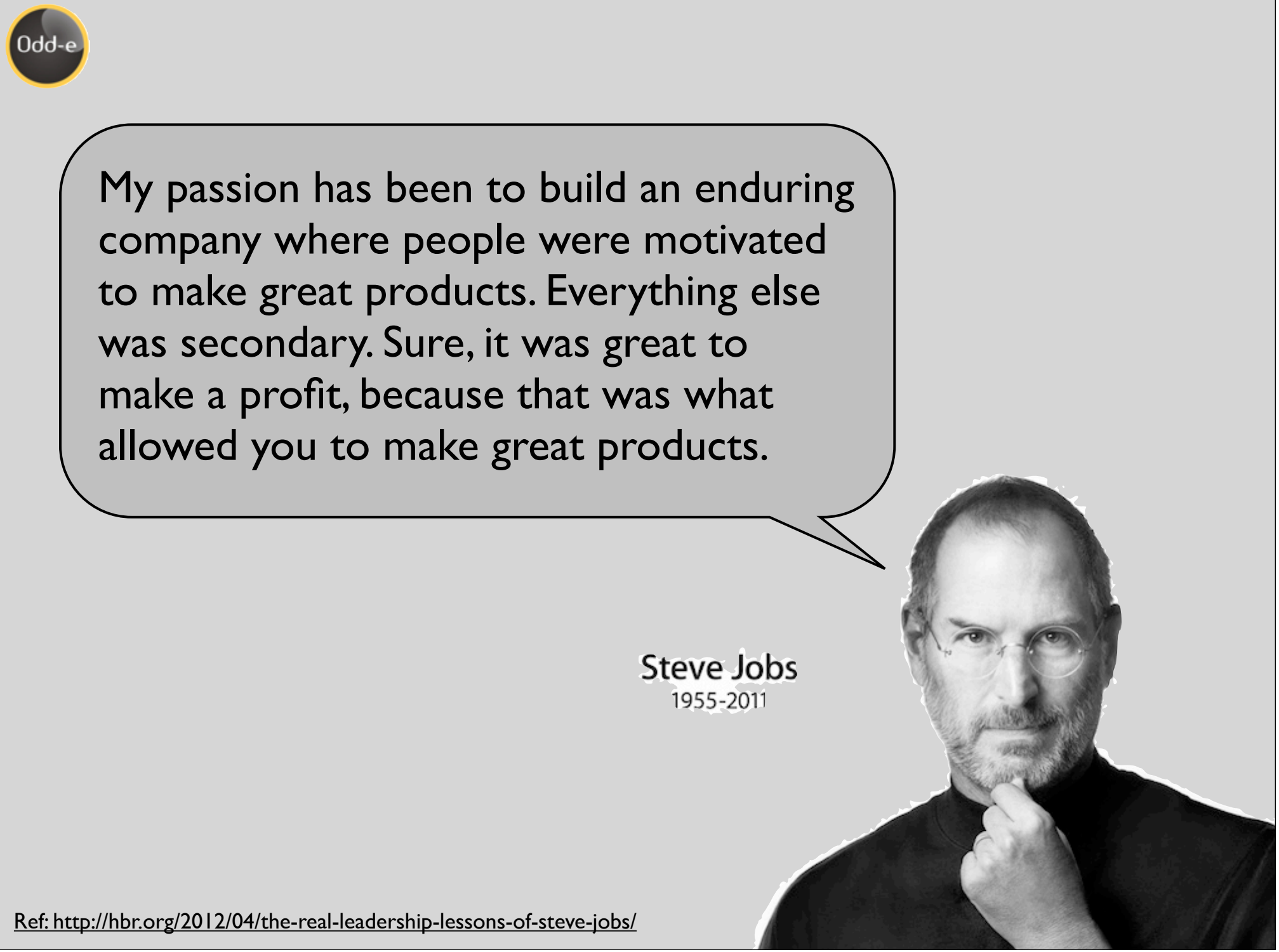


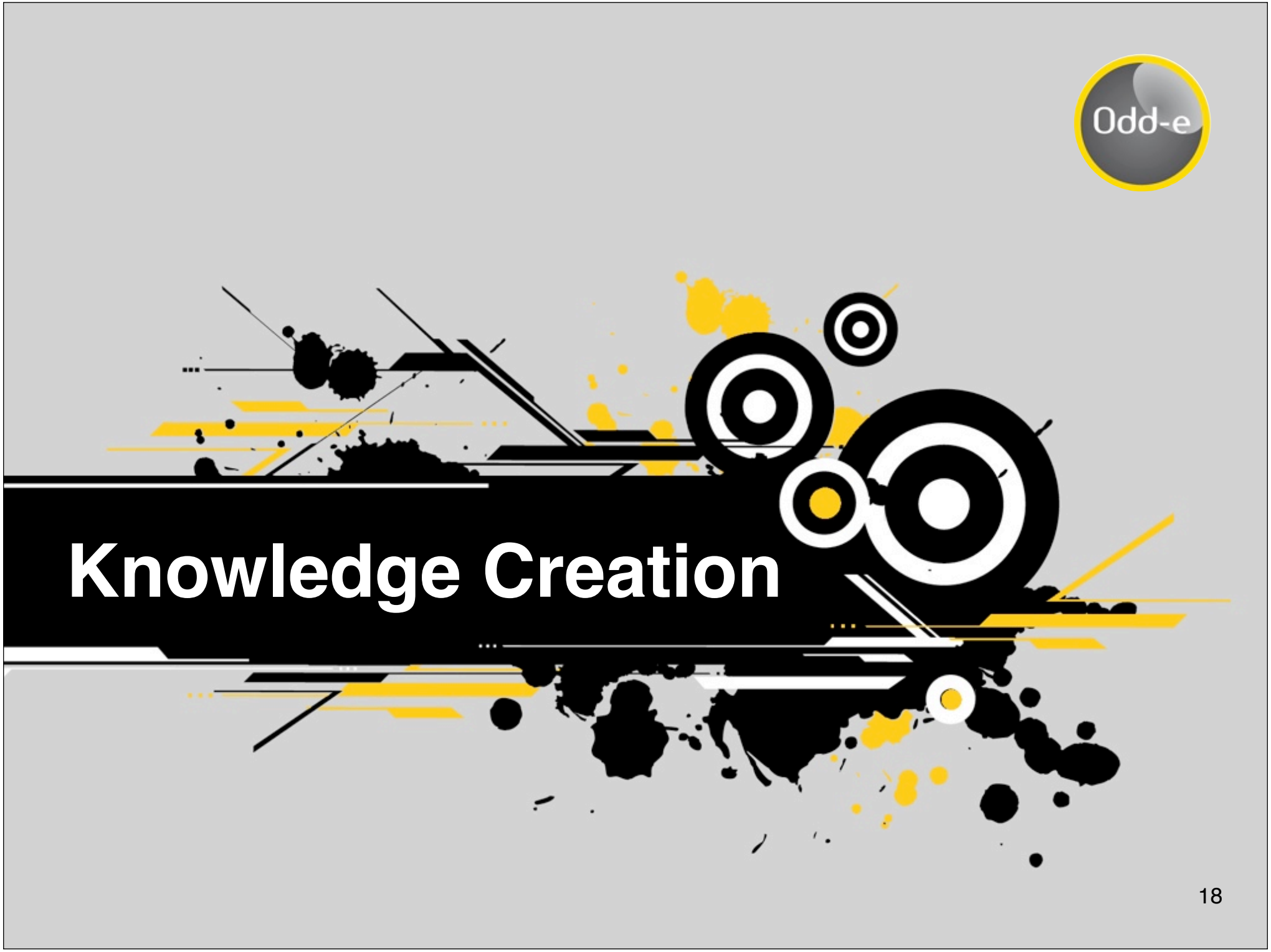
Work re-design



Principles of Job Enrichment:

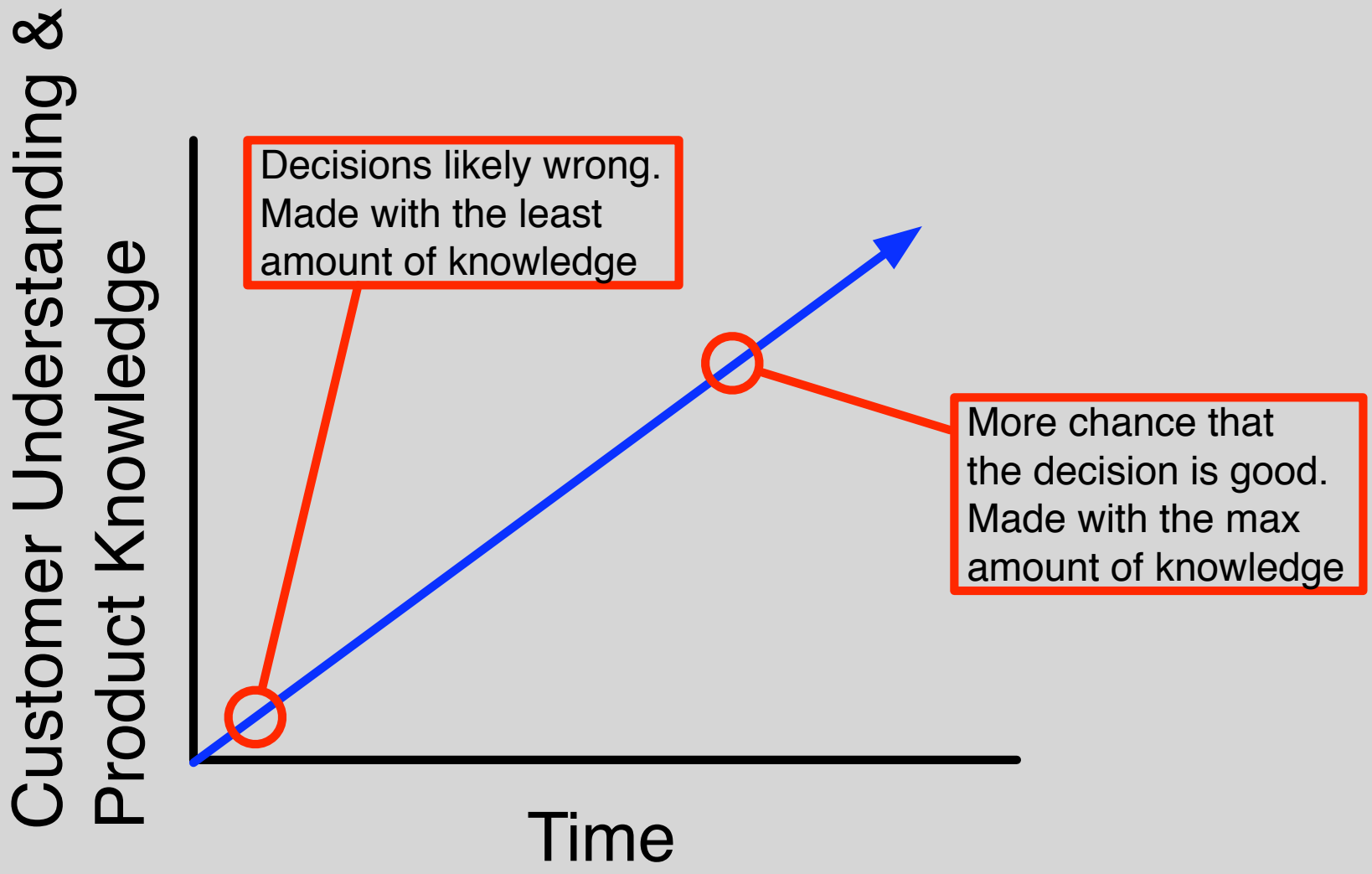
1. Combine tasks
2. Form **natural** work units
3. Client relationships
4. Vertically load the job
5. Feedback channels


Odd-e
My passion has been to build an enduring company where people were motivated to make great products. Everything else was secondary. Sure, it was great to make a profit, because that was what allowed you to make great products.
Steve Jobs
1955-2011
Ref: http://hbr.org/2012/04/the-real-leadership-lessons-of-steve-jobs/


Odd-e
Knowledge Creation

Knowledge Creation

Knowledge / Time



Accurate Early Estimagics

Odd-e

Improve
Software

Increase
Learning


Odd-e
Nature of Software

Gardening - Hunt / Thomas

Rather than construction, software is more like gardening -- it is more organic than concrete.

You constantly monitor the health of your garden, and make adjustments as needed.

People are comfortable with the metaphor of building construction: it is more scientific than gardening, it's repeatable.

But we're not building skyscrapers -- we aren't as constrained by the boundaries of physics and the real world.

Refactoring visualized

Without refactoring:

Original program:

Making changes:

More changes:

With refactoring:

Small change

Refactor

Etc

Cost of change increases rapidly!

Cost of change does not increase

Legacy Being Created

loginController.js' init()
Elapsed time of 1 month

loginUI.js-es
Elapsed time of 5 months

Refactoring

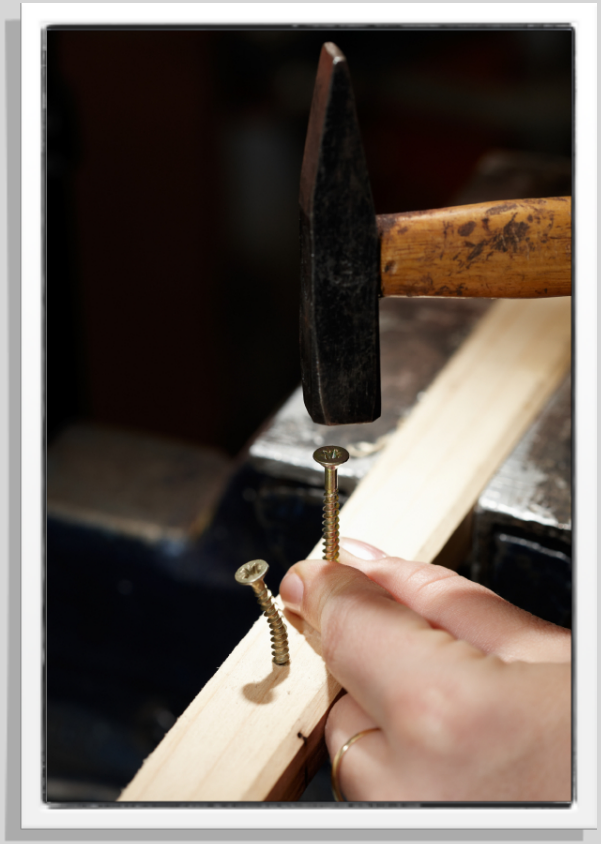
loginController.js' init()
Refactoring

loginUI.js-es
Refactoring

Safety net - Unit tests

2 main causes of legacy

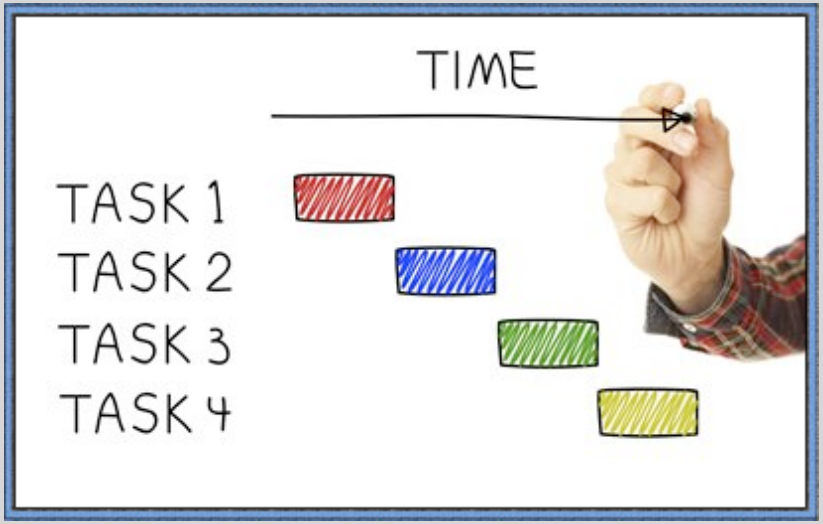
Pressure for unrealistic commitments



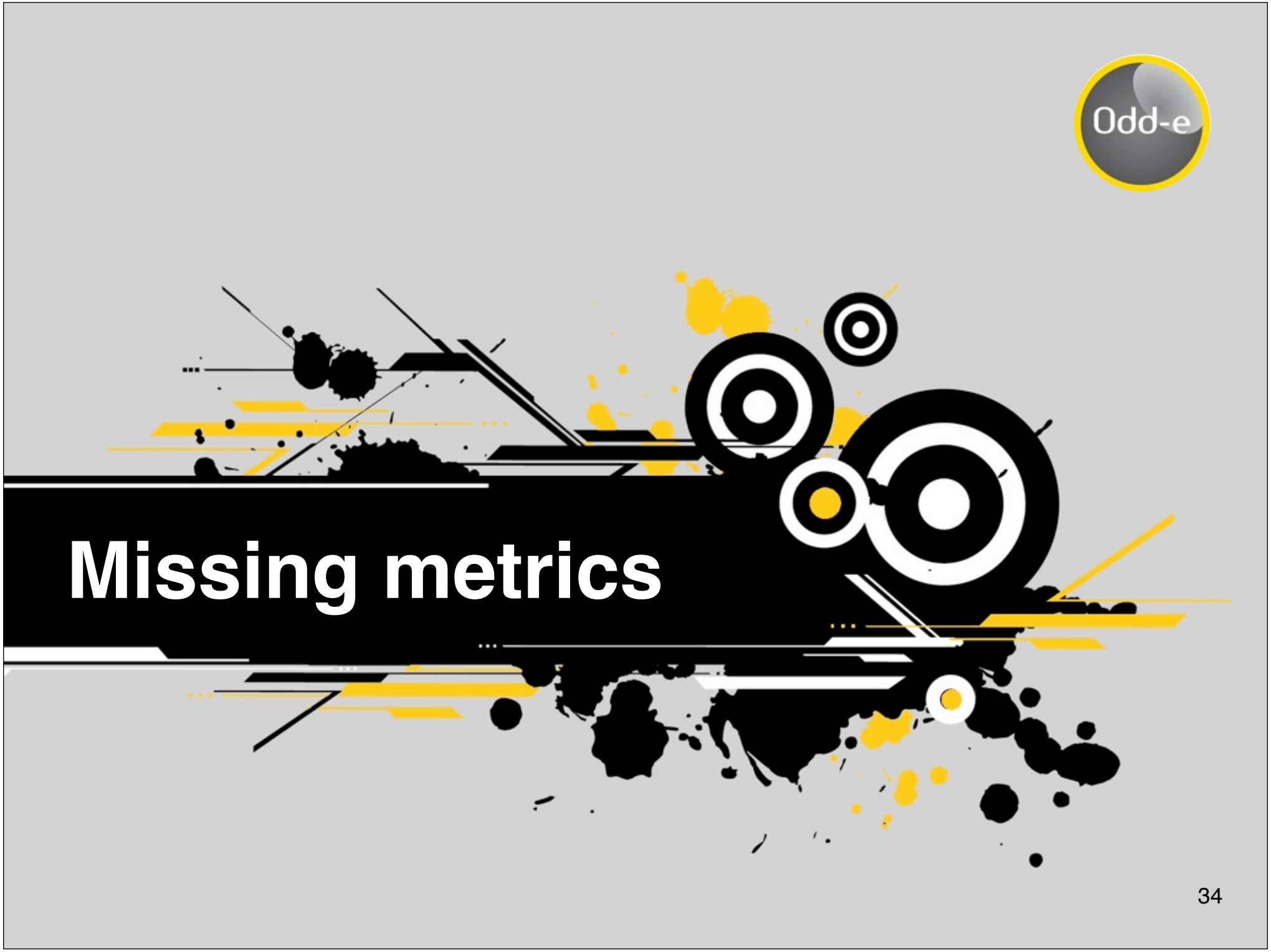
Lack of skill

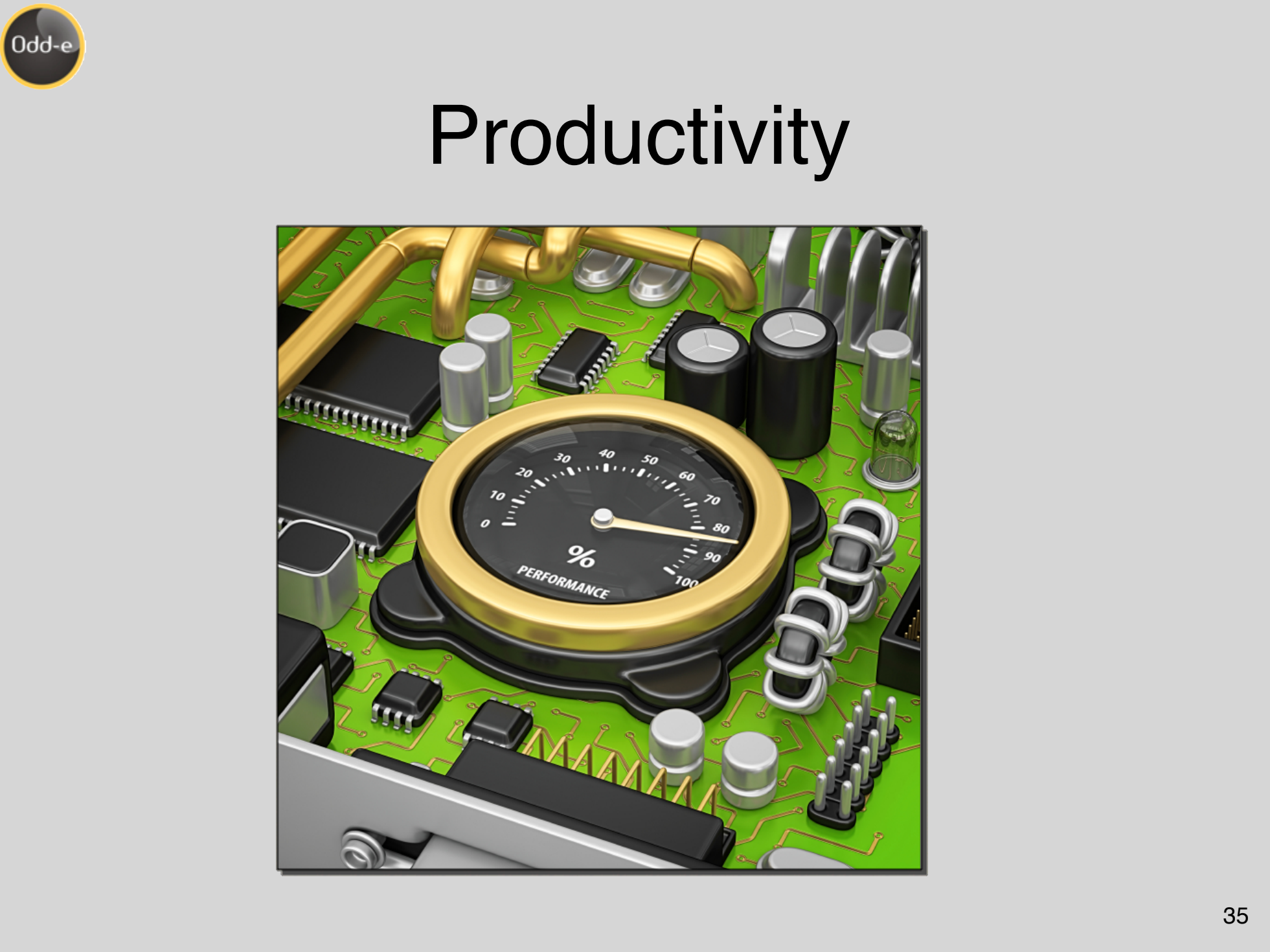
Manage
Products

Not
Projects



Not all work must be done in "projects."
That is simply one way of organizing work.


Odd-e
Missing metrics



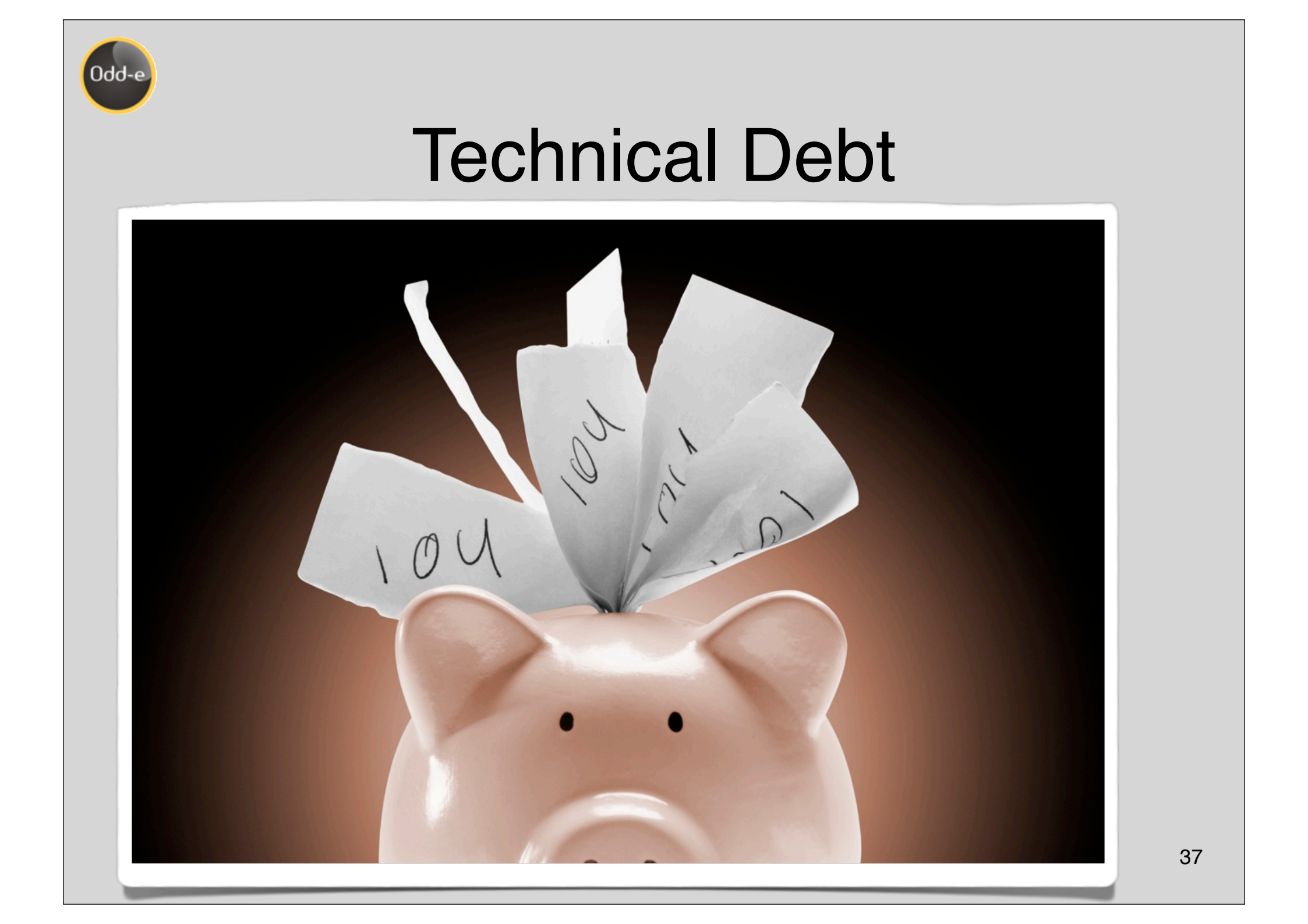
Productivity

Productivity - Martin Fowler

We see so much emotional discussion about software process, design practices and the like. Many of these arguments are impossible to resolve because the software industry lacks the ability to measure some of the basic elements of the effectiveness of software development. In particular we have no way of reasonably measuring productivity.

Productivity, of course, is something you determine by looking at the input of an activity and its output. So to measure software productivity you have to measure the output of software development - the reason we can't measure productivity is because we can't measure output.

http://martinfowler.com/bliki/CannotMeasureProductivity.html

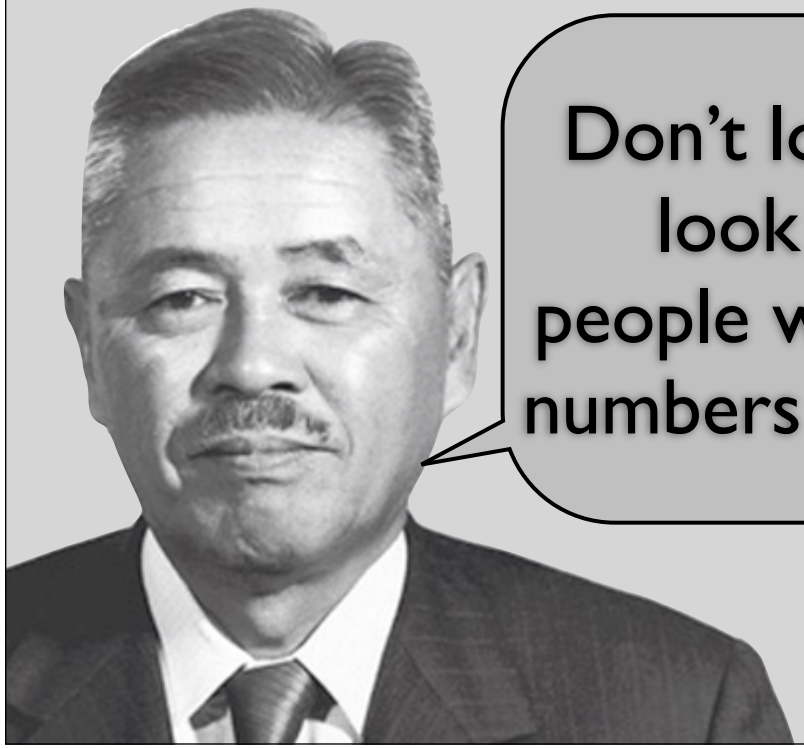


Technical Debt

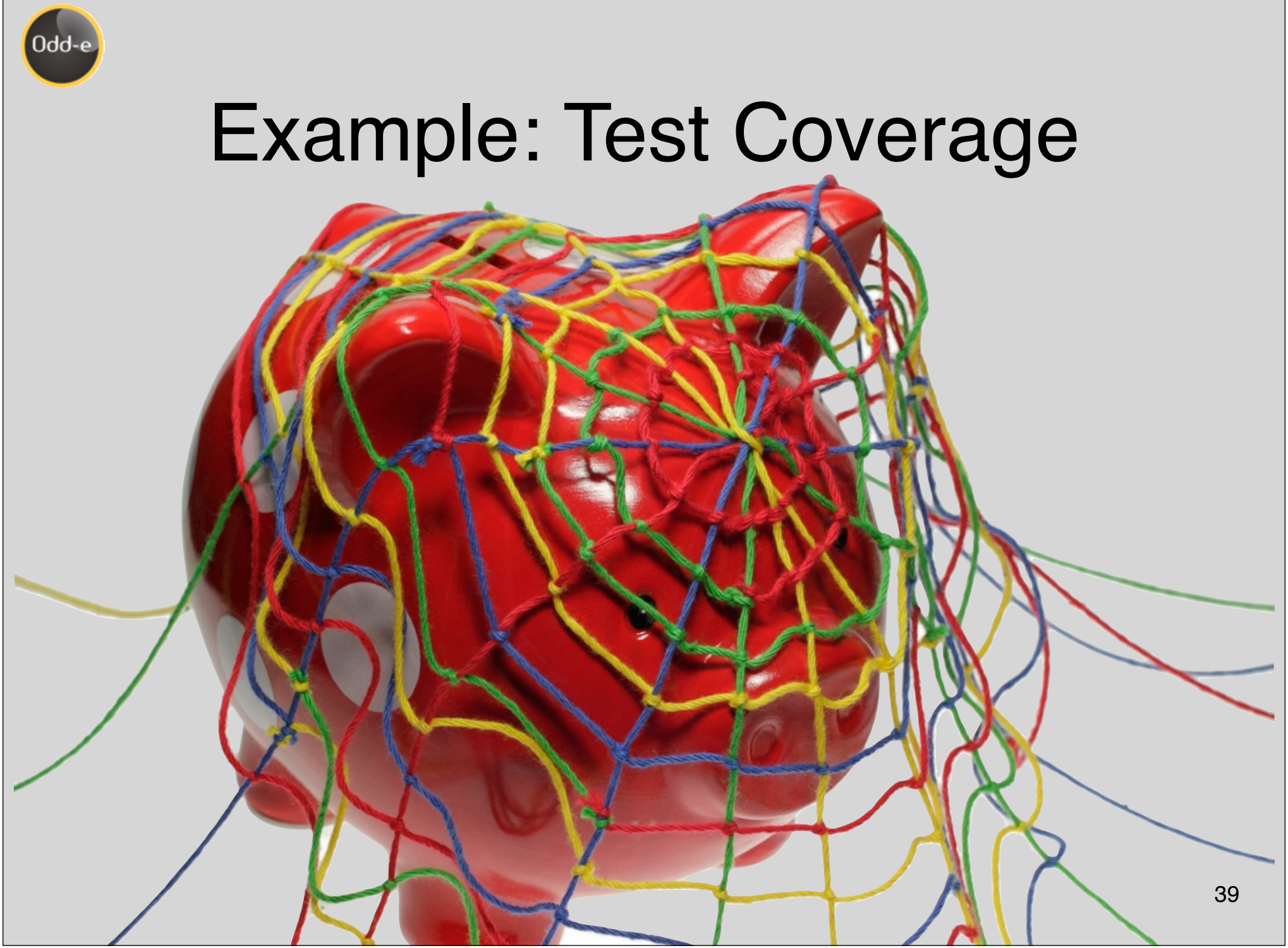
Ohno / Deming

Deadly Disease #5

Running a company on visible figures alone



Don't look with your eyes, look with your feet... people who only look at the numbers are the worst of all.



Example: Test Coverage

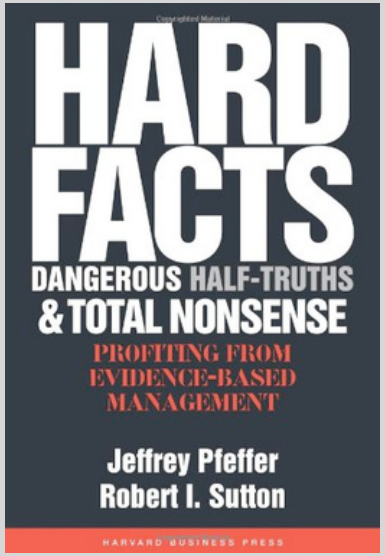
Metrics and their use...

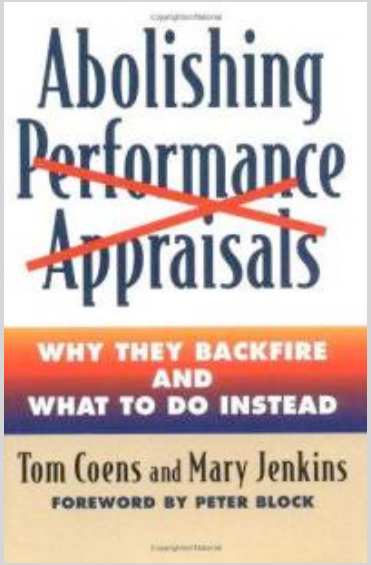
Don’t focus on the metric and the numbers

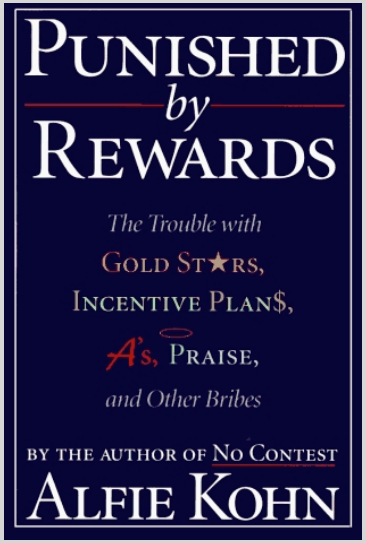
But on who sets them and how they use them

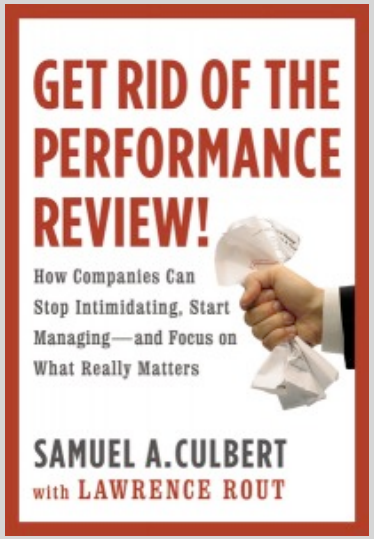
Go and See

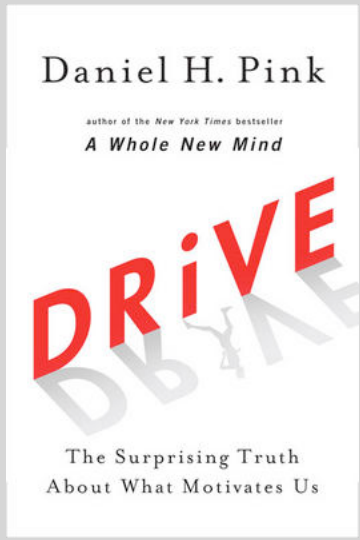
Performance Reviews









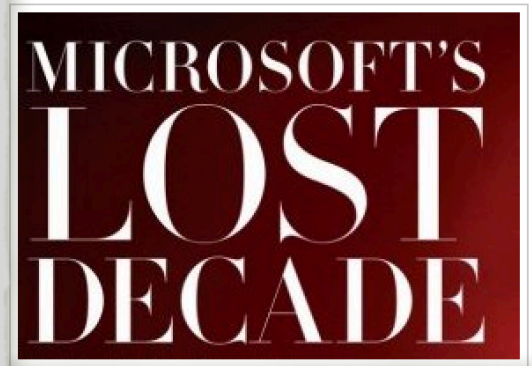


Adobe Systems set to scrap annual appraisals, to rely on regular feedback to reward staff

Devina Sengupta, ET Bureau Mar 27, 2012, 10.39AM IST

Tuesday, July 03, 2012

How to Kill Teams Through "Stack Ranking"



The newest Van
the Executive E
It starts with th

Analyzing one o
lost decade of M
V.F.'s newest co
"astonishingly
"could serve as

Relying on doze
including e-mail
Eichenwald offe

http://online.wsj.com/article/SB10001424052748704844904576189071859468086.html

http://www.vanityfair.com/online/daily/2012/07/microsoft-downfall-emails-steve-ballmer

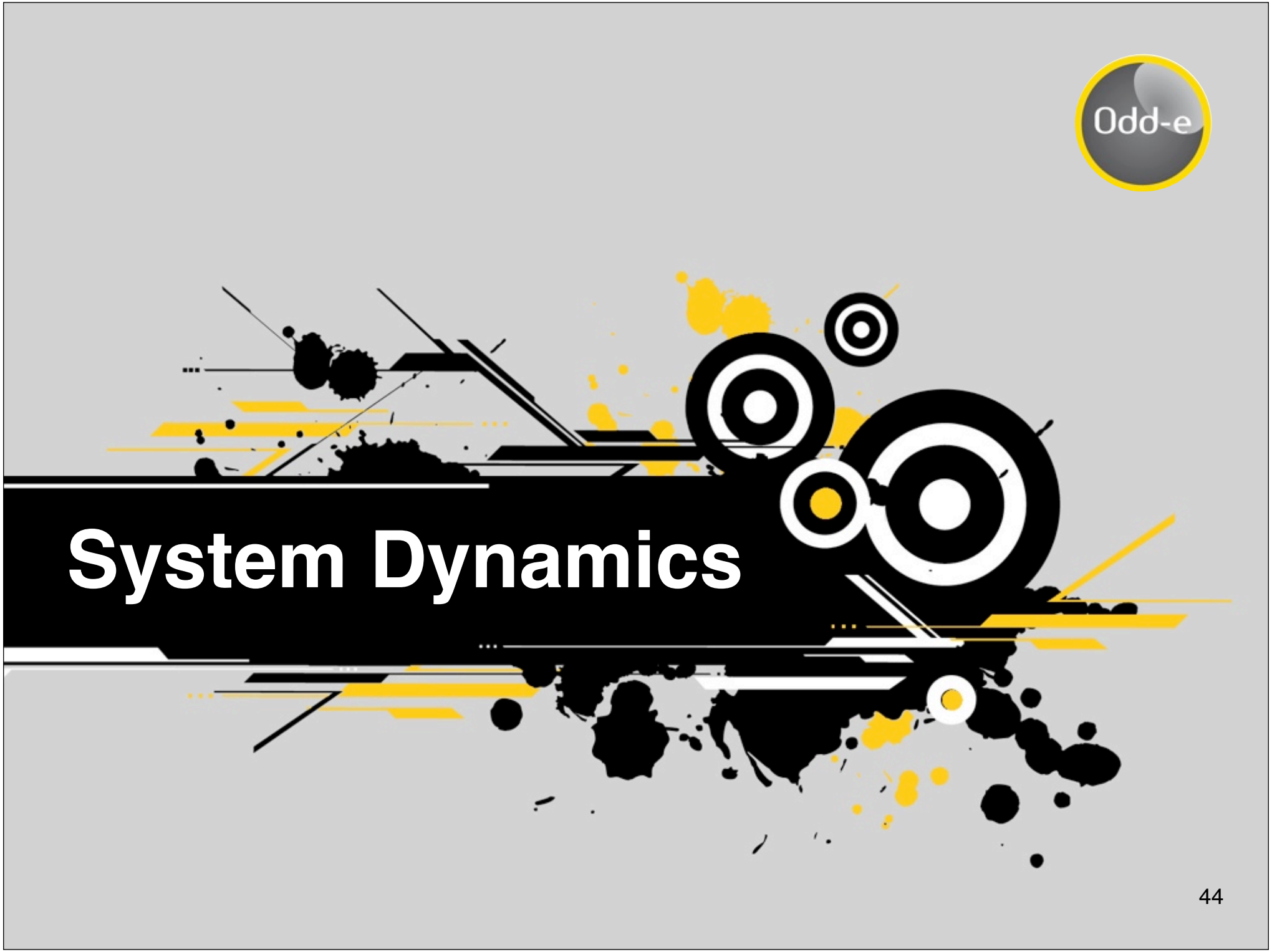
Deming

Point #12

Remove barriers that rob people of their right to pride of workmanship. This means abolishment of the annual or merit rating.

Deadly Disease #3

Evaluation by performance, merit rating, or annual review of performance


Odd-e
System Dynamics

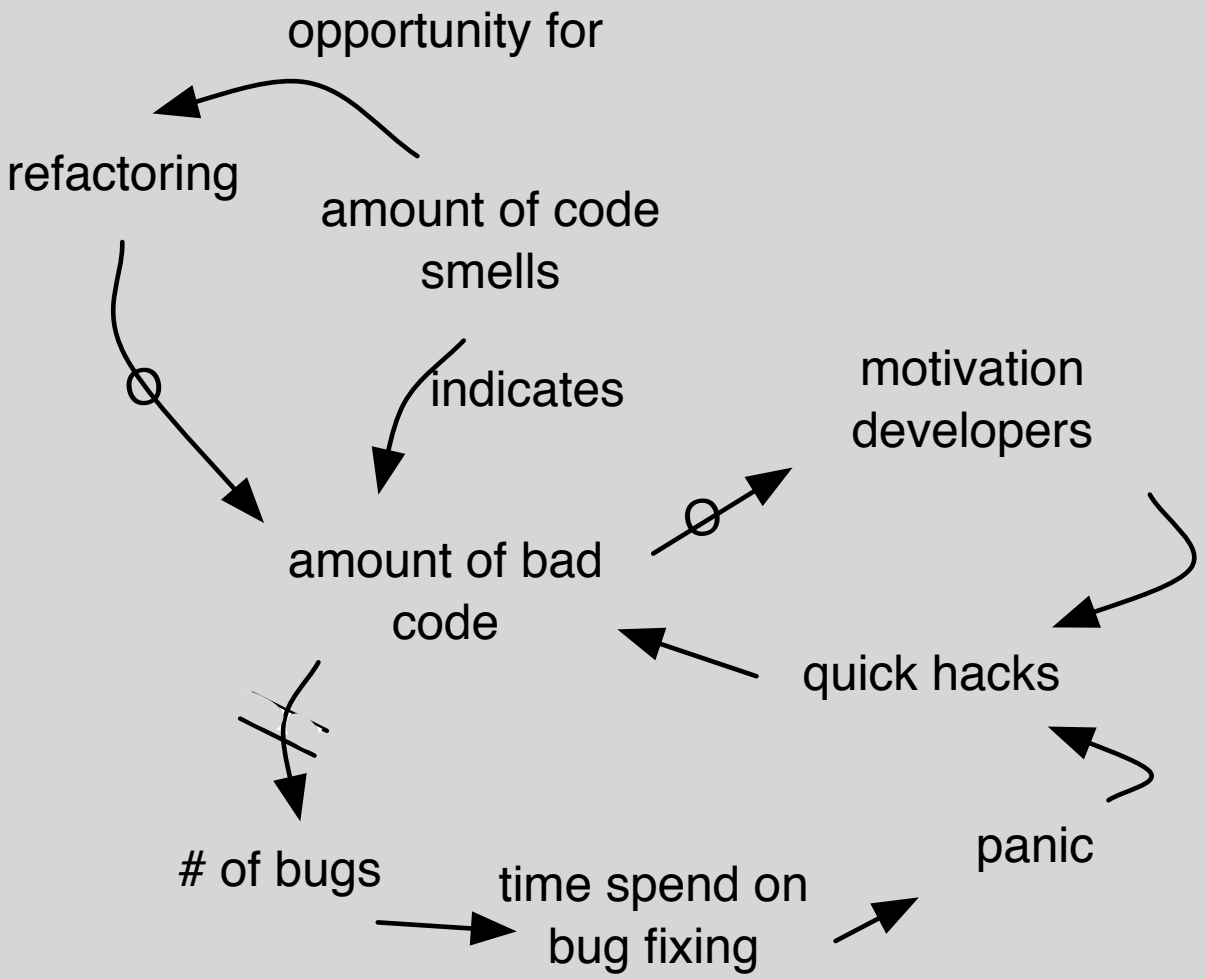
Systems Thinking

- Making better decisions by seeing:

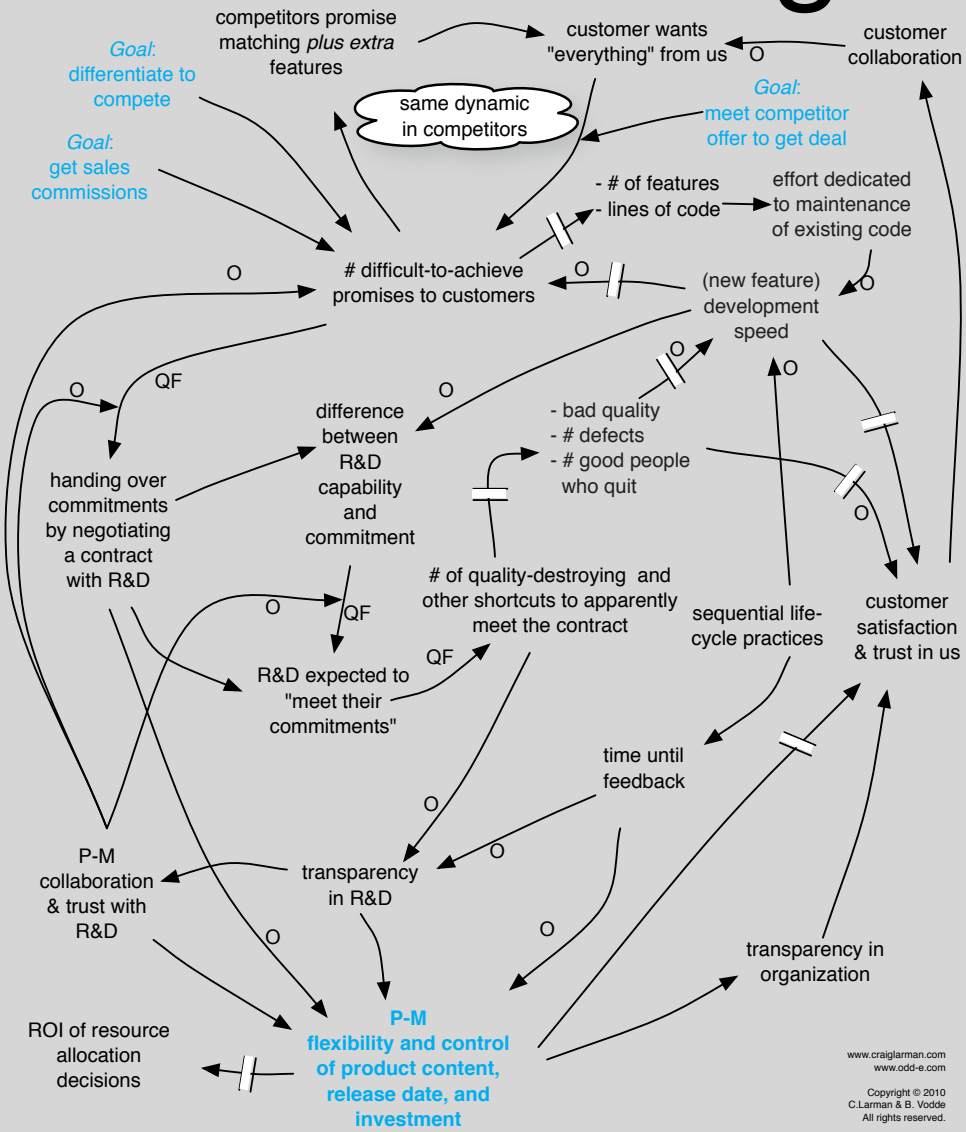
- Systems dynamics
  - What variables are there, how do they relate?
  - Especially considering **time**
- Mental models
  - What **assumptions** are there?
- **Root causes**
- Optimizations
  - and local optimizations

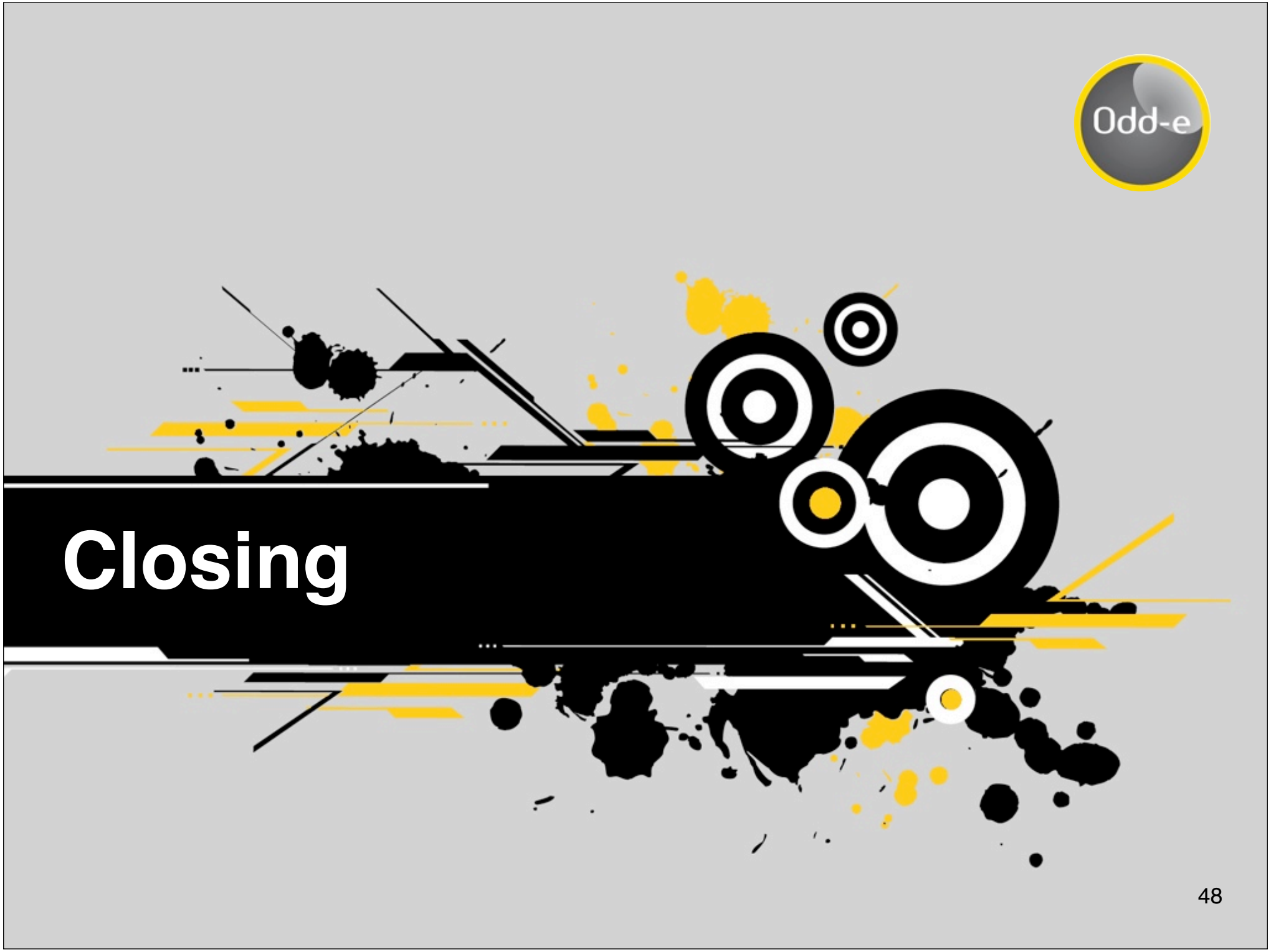
- And... **Go and See**

Example: Refactoring



Example: Product Management




Odd-e
Closing

Principles of Managing Software Development

- Be aware of Taylor
- Team == Product
- Software Development is Knowledge Creation
- The Nature of Software is to Grow Ugly
- The 2 Forever Missing Metrics

Meta-principle:
Systems Thinking

Questions