February 2020 PLUG Meeting:

KANBAN ... what is it ?

Peter Northrop I.S.P., Cert.APM, SMC

Adjunct Faculty
Department of Computing and Information Systems

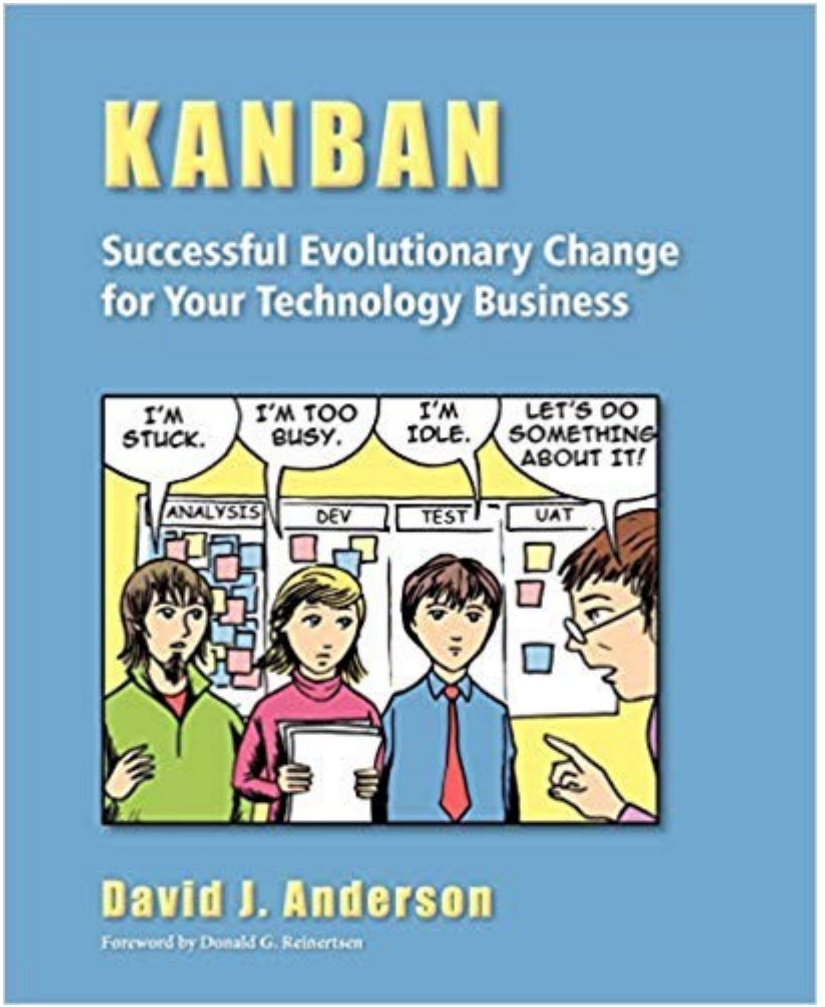
Course Instructor
COIS-ADMN 3850H - Fundamentals of Project Management

PLUG Member since 2004

QUESTION

How does a project come to be late ?

One day at a time !



Why KANBAN ?

Many longstanding PM Issues:

Fundamentally, its about

Greater productivity

Higher quality work

Sustainable pace

Scaling of Agile adoption

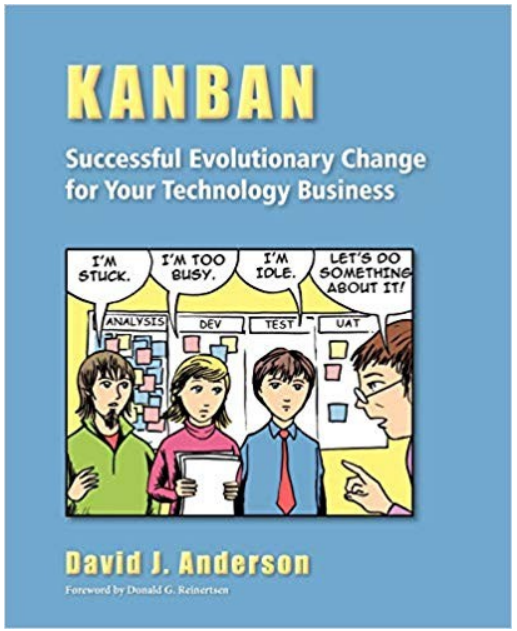
Why “one size fits all” development methodologies don't always work

TEAMS have different ...
- Skill Sets
- Levels of Experience
- Levels of Capability

PROJECTS have different ...
- Budgets
- Schedules
- Scopes
- Risk Profiles
- Changing Requirements

ORGANIZATIONS have different ...
- Value Chains
- Target Markets
- **Corporate Culture**

In some way, USUALLY every situation is unique ...
Building HWY 407 or the Hoover Dam is not the same as creating a new software application



Takeaways:

Kanban systems are from a family of approaches known as "pull" systems.

Eliyahu Goldratt's Theory of Constraints is one. [*The Goal,* 1984]

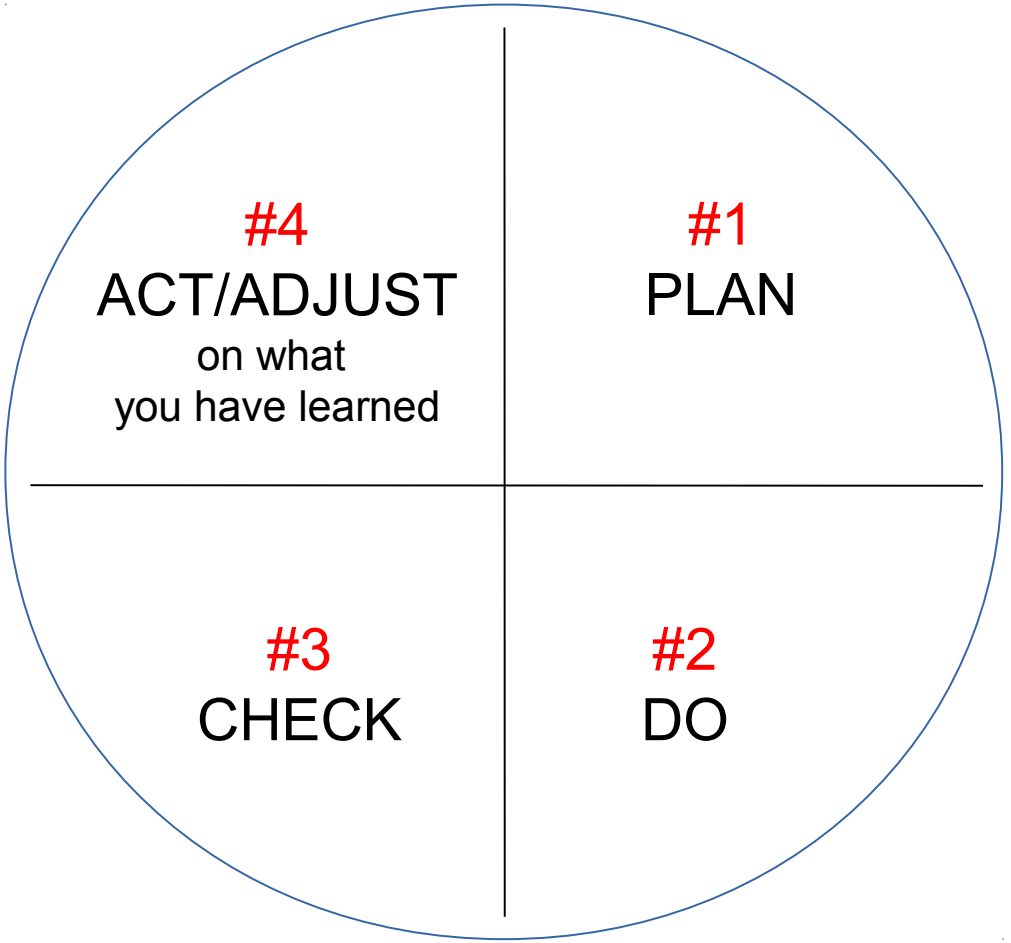
Motivation:
- to achieve a sustainable pace of work
- introduce process changes with minimal resistance.

Underpins the Toyota Production System and its *kaizen* approach to continuous improvement.

Microsoft implemented the first virtual system for software engineering in 2004.

Growth in adoption began after the Agile conference in Washington, D.C. in August 2007.

~ 100 years in the making ...



- **Remember:** **The newer IID processes have evolved from the Shewhart PDCA cycle**
  developed by Walter Shewhart in the **1920s** for the improvement of general business processes.
  He is the father of the concept of “continuous improvement” which was further developed in the 1950s by Edwards Deming who applied it in Japan after the war to produce low cost goods for the rest of the world.

Kan-ban is a Japanese word for "**signal card**" ...
a new piece of work can only be started when a card is available.

This free card is attached to a piece of work and follows it
as it flows through the system.

When there are no more free cards, no additional work can be started.

Any new work must wait in a queue until a card becomes available.

This mechanism is known as a **pull system** because new work is pulled into the system **when there is capacity to handle it** ...
rather than being pushed into the system based on demand.

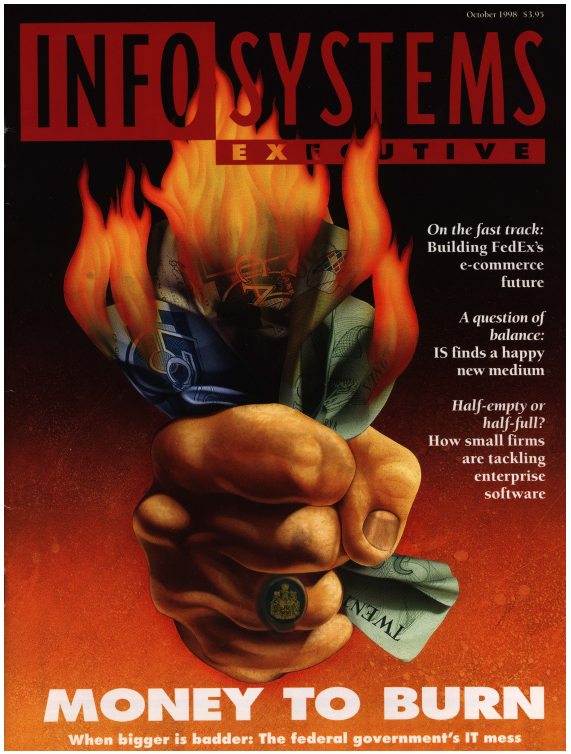
A pull system cannot be overloaded if the capacity, as determined by the number of signal cards in circulation, has been set appropriately.

Kaizen

is the Sino-Japanese word for “**improvement**”

But, we've gotten a bit ahead of ourselves at the moment ...

What follows next is important background information



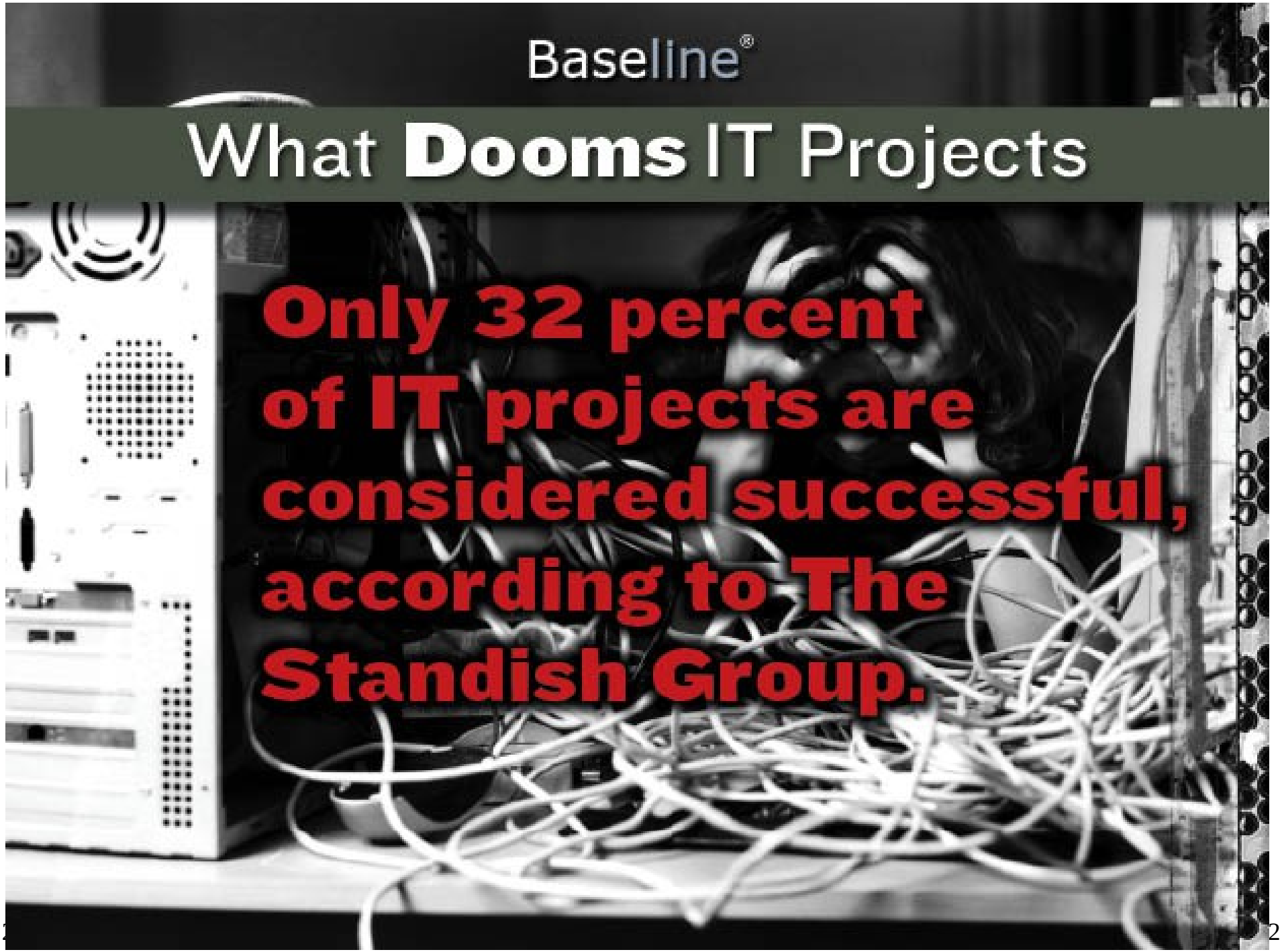
[well documented]

70%

FAILURE RATE

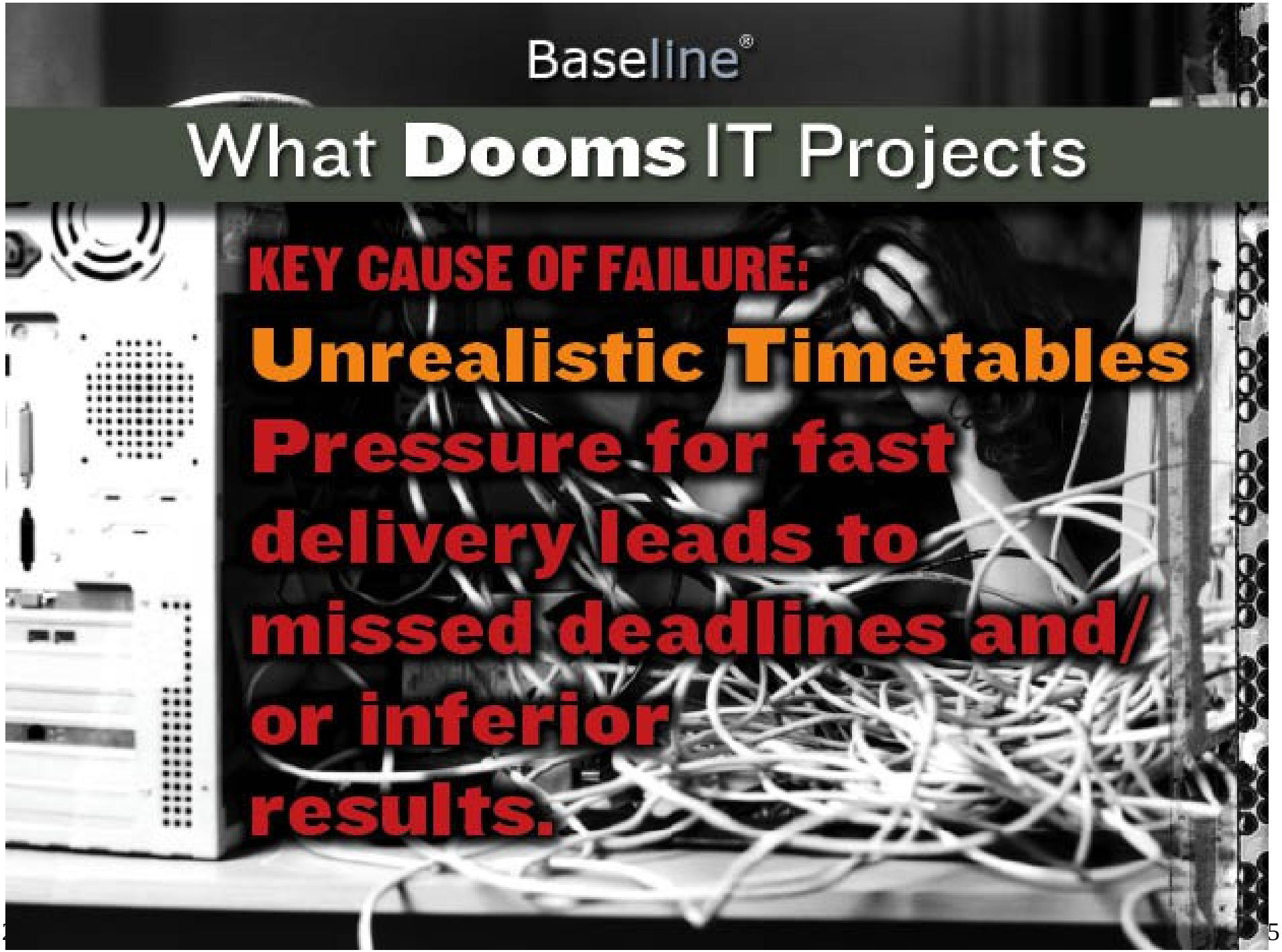
IN THE SOFTWARE INDUSTRY

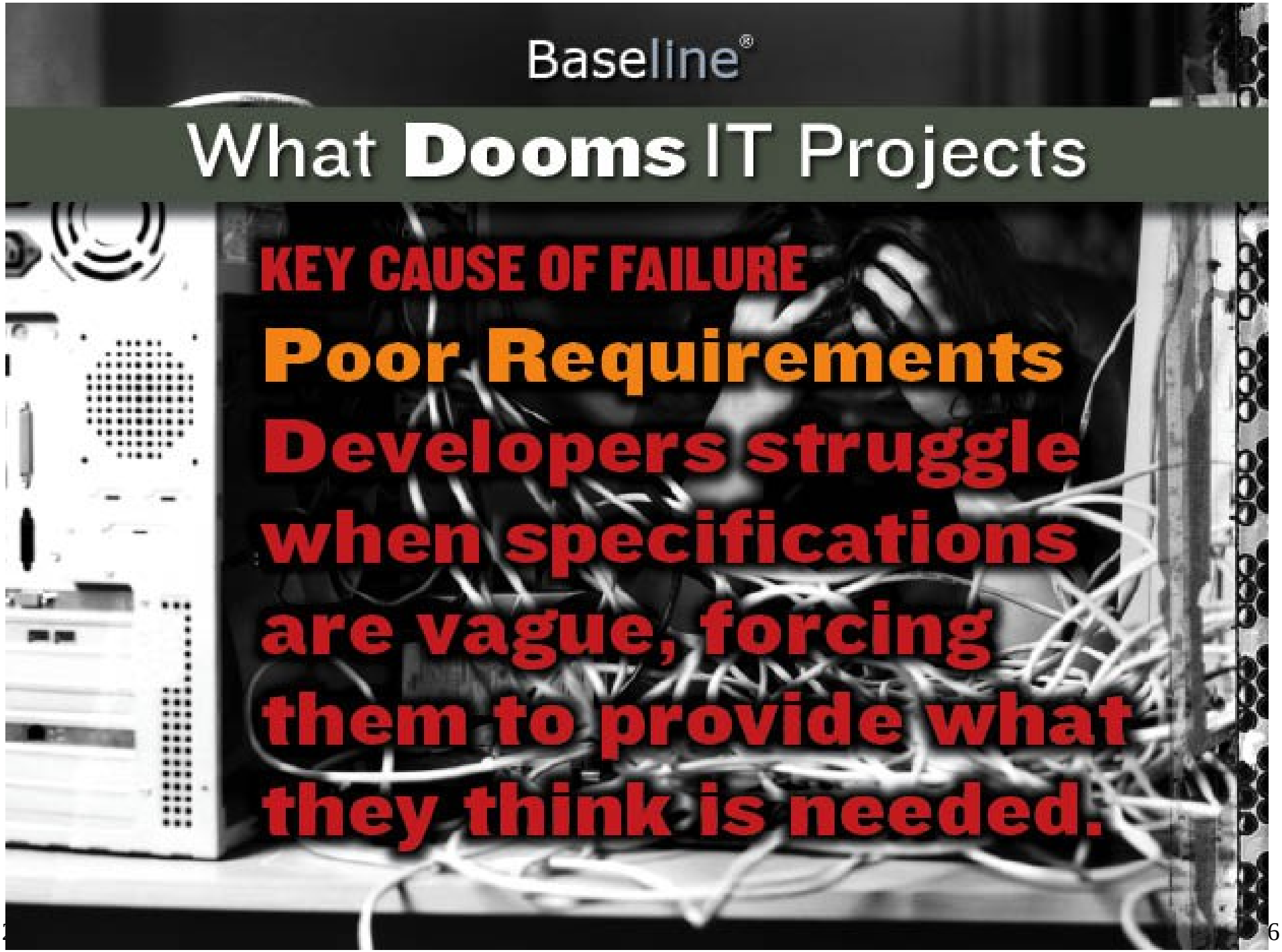
What's worse: **CLOSE TO 80% OF ALL ORGANIZATIONAL CHANGE INITIATIVES FAIL TO ACHIEVE THEIR INTENDED RESULTS**



Baseline®

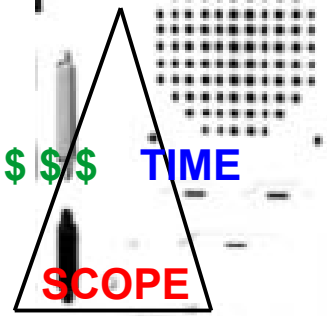
What **Dooms** IT Projects





Baseline®

What **Dooms** IT Projects



"THE IRON TRIANGLE"

KEY CAUSE OF FAILURE

Scope creep

When projects take add requirements on the fly, deadlines and resources often are not properly adjusted accordingly.

Indeed ...

The project management profession is in transition and at a major turning point in its history

This course introduces a much more adaptive approach and covers the benefits of Agile and SCRUM in detail.

Today, the emphasis is on matching the approach to the project rather than using the same old plan-driven approach for every project.

It will help you to

BREAK THROUGH and **better understand**

"the chasim in Project Management philosophies"

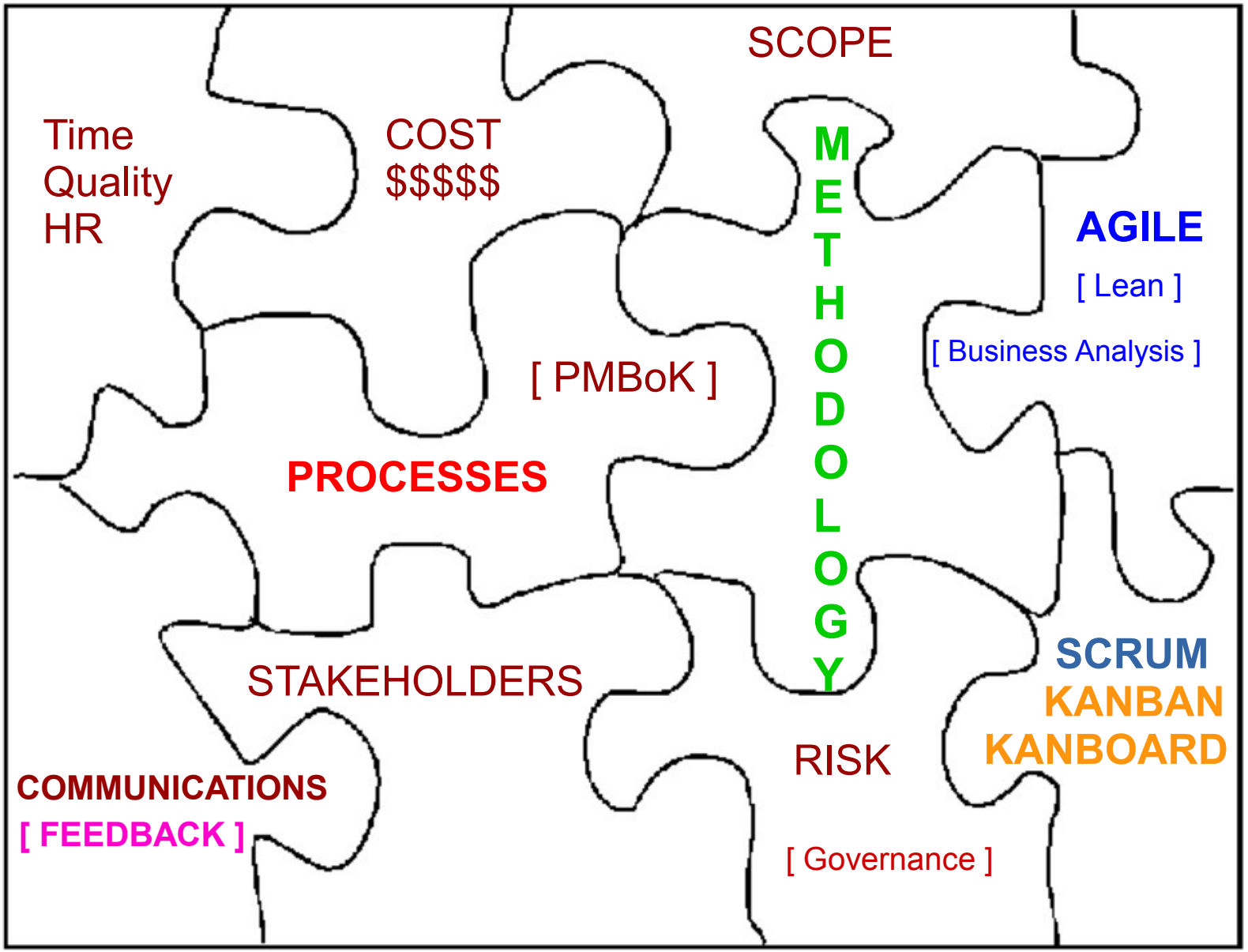
1.5 Scrum vs. Traditional Project Management

Table 1-3 summarizes many of the differences between Scrum and traditional project management models.

	Scrum	Traditional Project Management
Emphasis is on	People	Processes
Documentation	Minimal—only as required	Comprehensive
Process style	Iterative	Linear
Upfront planning	Low	High
Prioritization of Requirements	Based on business value and regularly updated	Fixed in the Project Plan
Quality assurance	Customer centric	Process centric
Organization	Self-organized	Managed
Management style	Decentralized	Centralized
Change	Updates to Productized Product Backlog	Formal Change Management System
Leadership	Collaborative, Servant Leadership	Command and control
Performance measurement	Business value	Plan conformity
Return on Investment (ROI)	Early/throughout project life	End of project life
Customer involvement	High throughout the project	Varies depending on the project lifecycle

Table 1-3: Scrum vs. Traditional Project Management

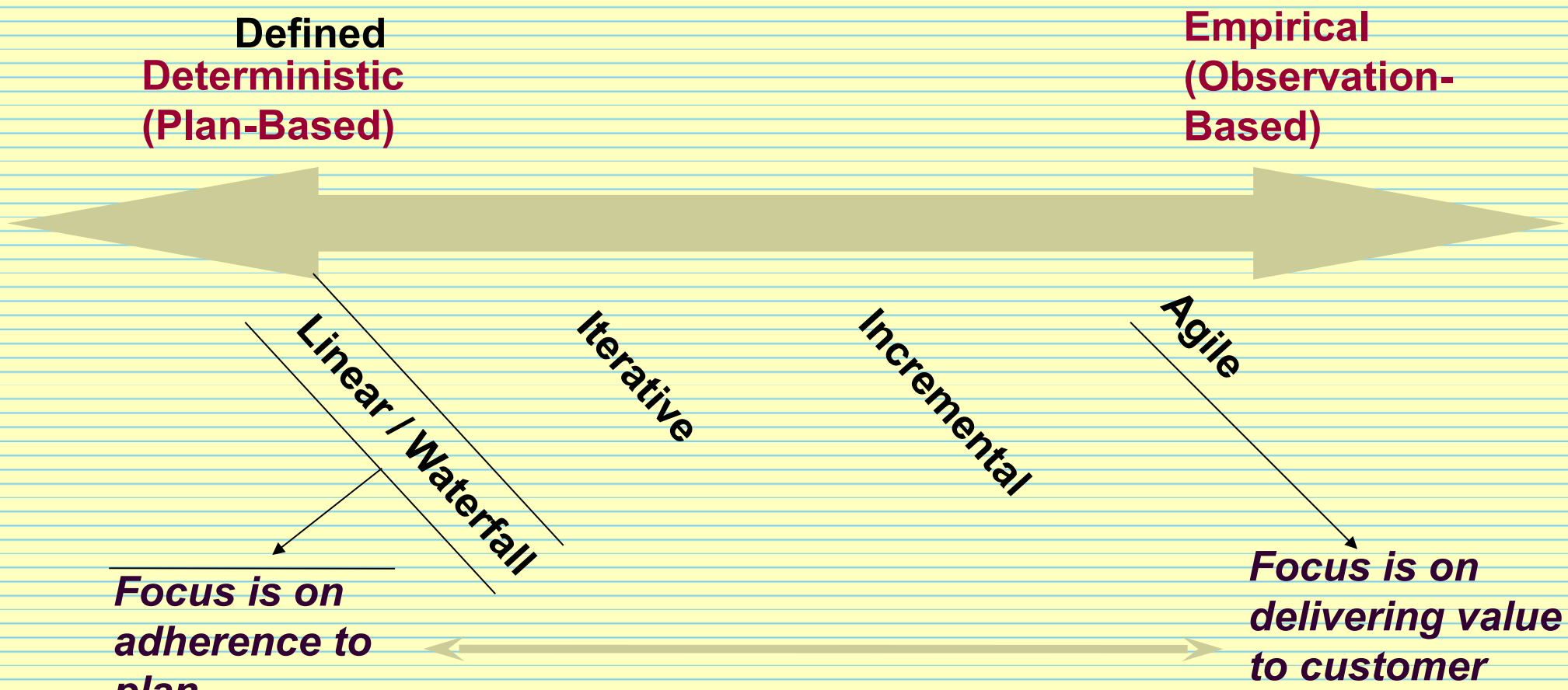
"Adaptive" vs a "Plan-Driven" approach



3850H Course Content – How things fit

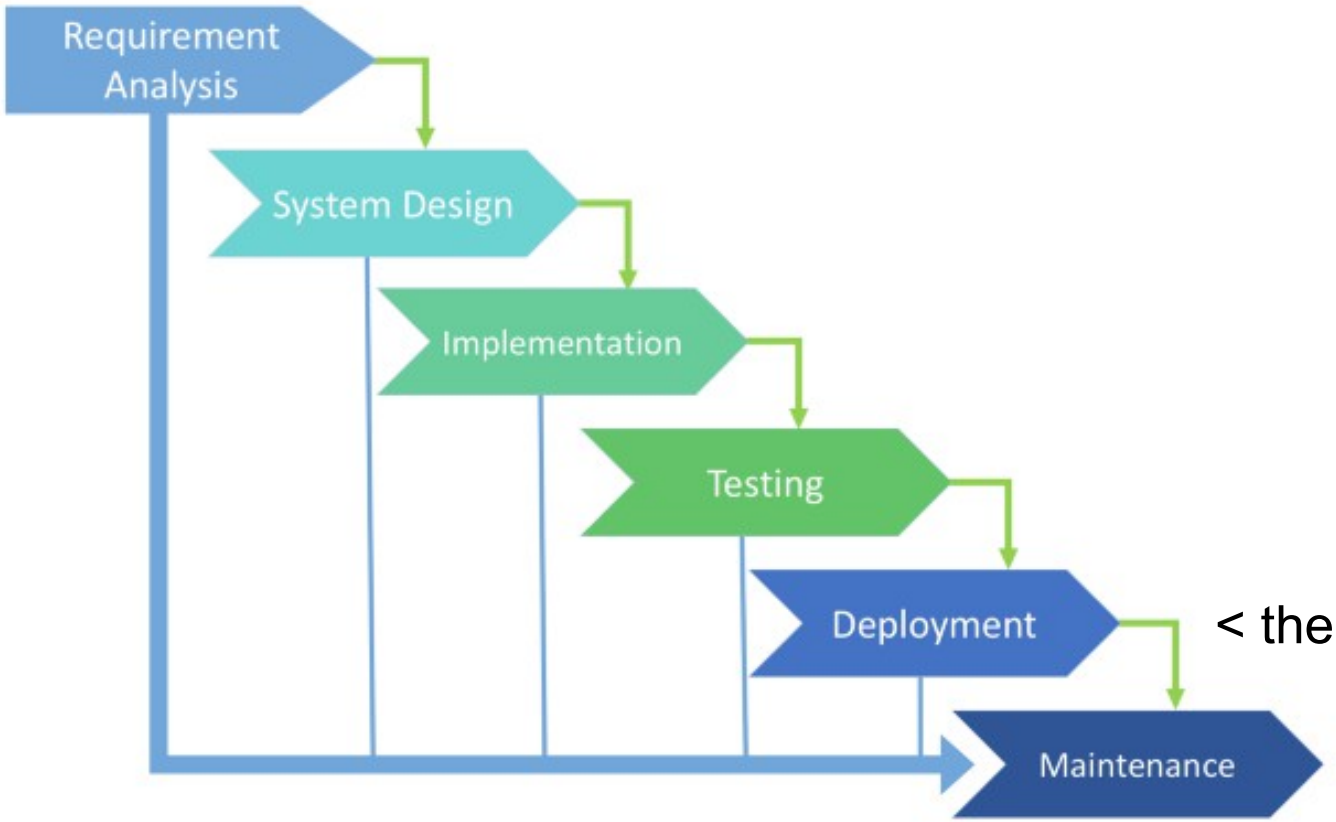
PMI was founded in 1969

We have a Range of PM Methods



"BIG BANG" vs "PHASED" approach

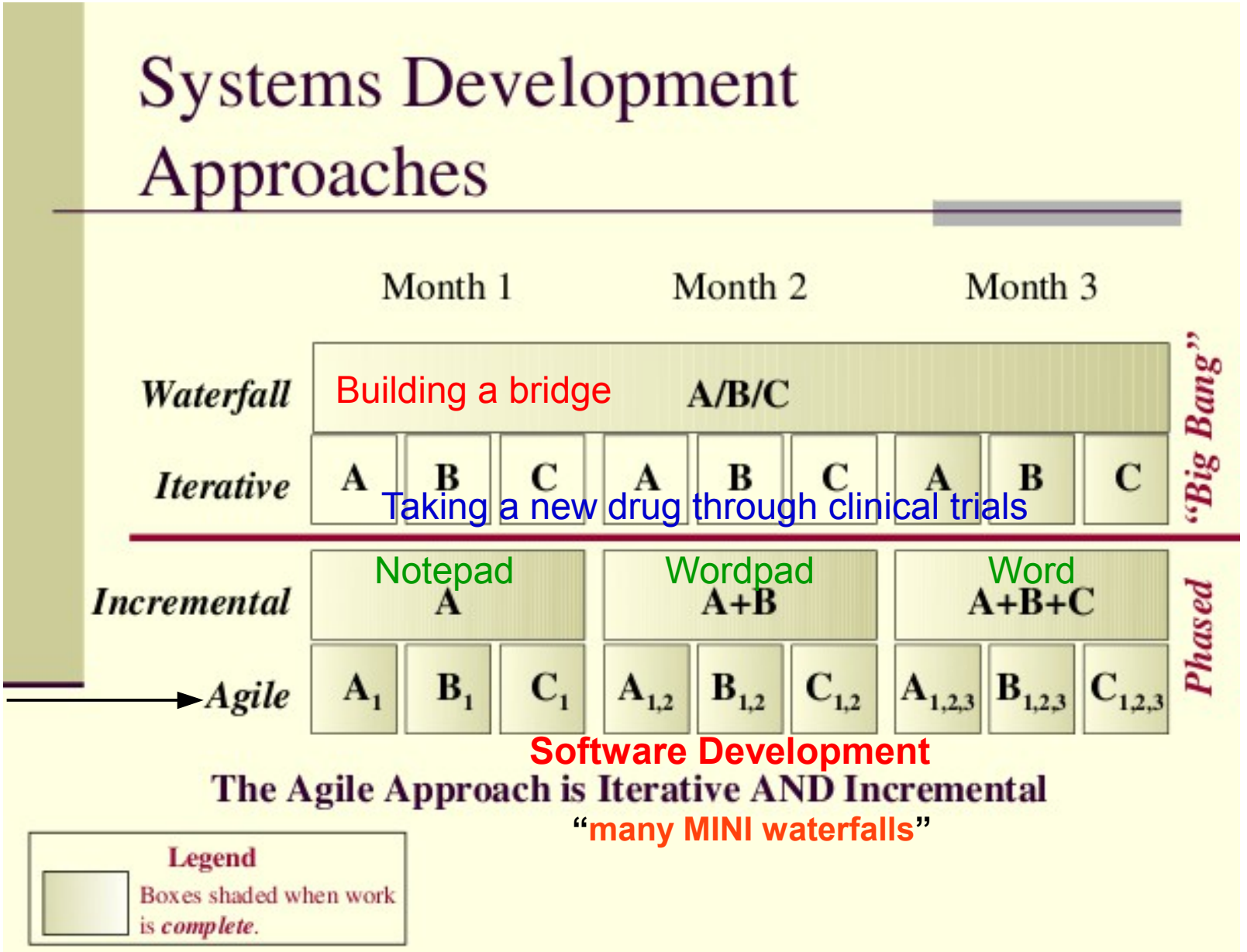
- **“Waterfall” - typically how things are done:**



< the “Big Bang”

Little room for “feedback” ... until the end

Systems Development Approaches



Defined versus Empirical Process Models

Defined = "Rifle" Approach

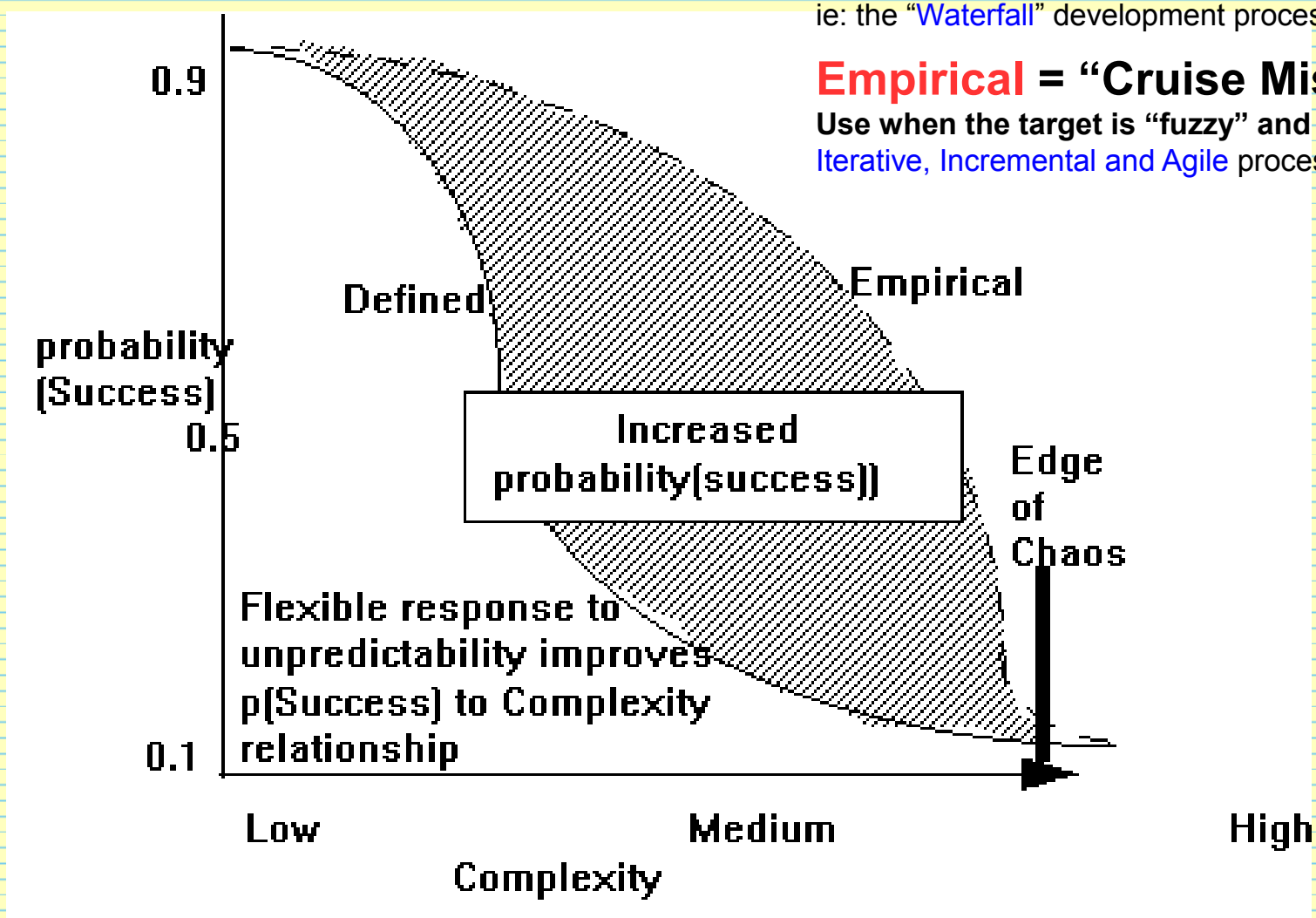
Use when the target is known and well defined

ie: the "Waterfall" development process.

Empirical = "Cruise Missile" Approach

Use when the target is "fuzzy" and not clear.

Iterative, Incremental and Agile processes are good options.



To recap:

Agile methods, at their core, are all about managing the impact of change on a project.

- **Iterative and Incremental Development**
  Each iteration builds upon the last, adding additional layers of functionality.
- **Short Iterations**
  More responsiveness to change, and reducing the risk of building "the wrong thing" based on unclear or changing requirements.
- **Progress Measured via Completed Features**
- **Open, Flexible Design**
  Designs are flexible and extensible using open standards wherever possible.
- **Empowered Teams:** Teams of specialists who know their jobs well.
- **Personal Communications**
  Speaking face to face, with the support of a whiteboard for drawing diagrams, is a much more efficient way of working out design details.

It's all about:

Reducing Risk

- Building the Wrong Thing
- Building the right thing **Wrong**
- Getting Stuck in “Design Churn”

Improving Control

- Frequent delivery of working code means progress is objectively measurable
- More chances for stakeholders to provide early feedback and redirect project priorities where necessary
- Misunderstanding surfaces earlier
- The sponsor has the ability to end a project early and still get measurable benefits.

- **The "Agile Manifesto":**

  - **Individuals and Interaction**
    over process and tools
  - **Working software**
    over comprehensive documentation
  - **Customer collaboration**
    over contract negotiation
  - **Responding to change**
    over following a plan

Agile Thinking

Don’t do it if you don’t *have* to.

Do it as informally as possible.

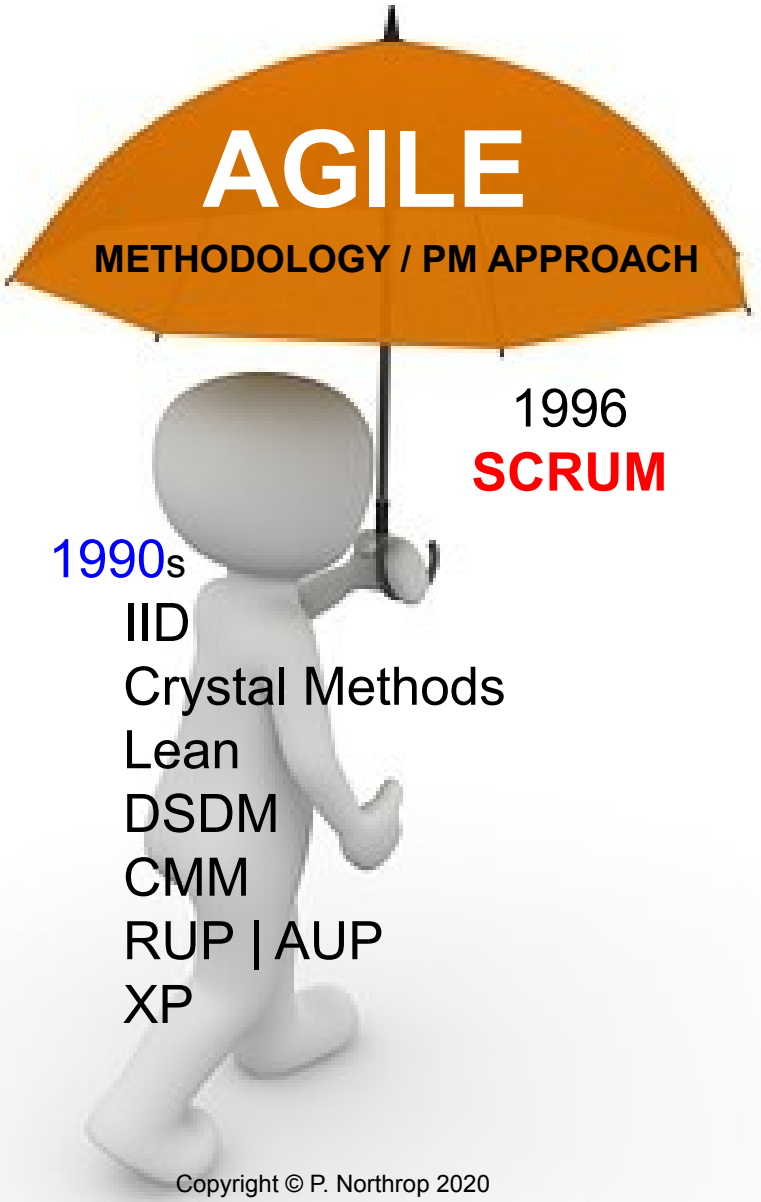
Procrastinate!

Continuously improve the process.

MUST HAVE | **GOOD TO HAVE** | **NICE TO HAVE**

SCRUM ...

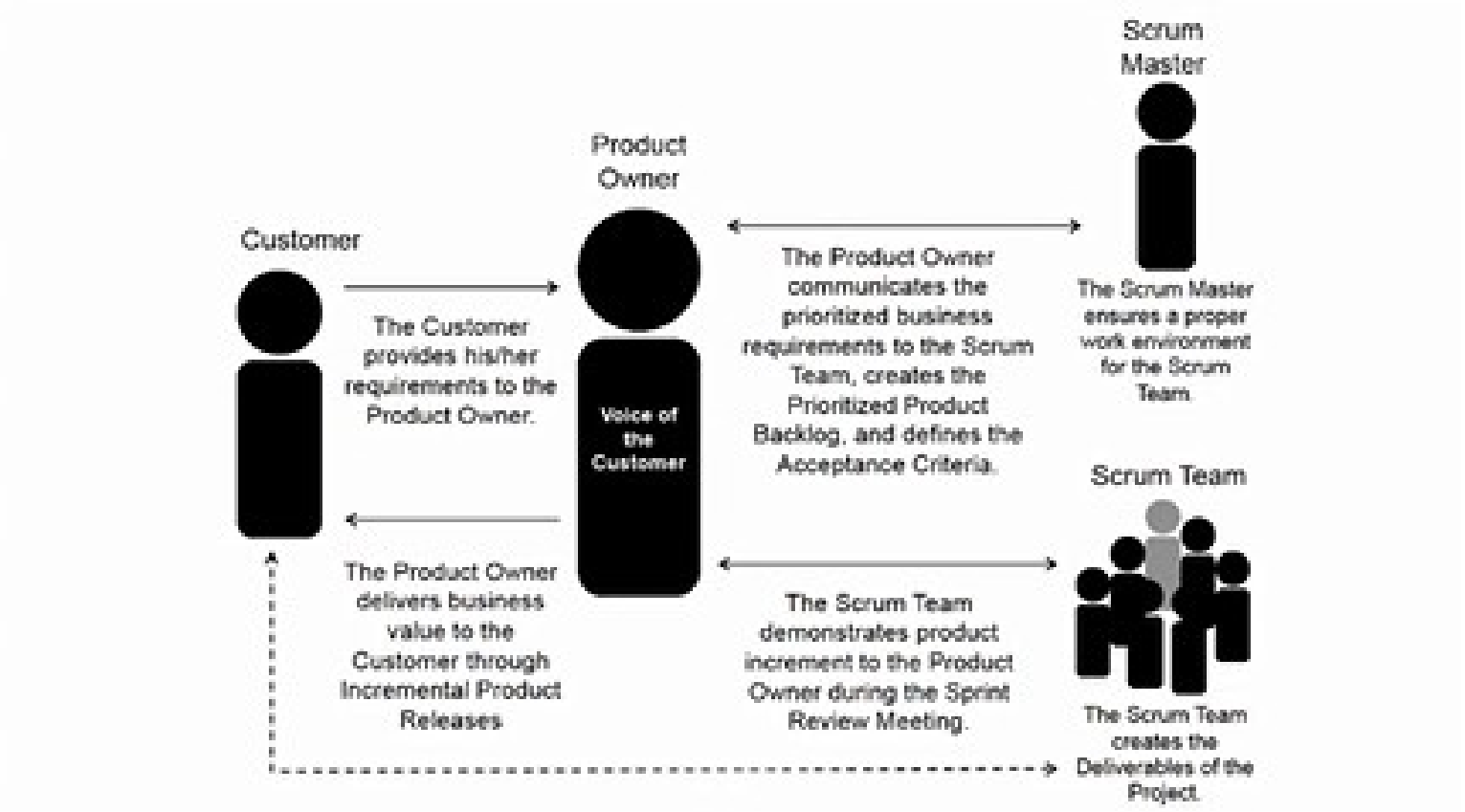
a management framework



Systems Development Timeline

- 1920s: **Walter Shewhart** – **"Father of Continuous Improvement"**
- 1950s: **Edwards Deming** – **applied Shewhart's work in Japan**
- 1950s:Iterative Incremental Development (IID)
- **1970: Waterfall Model** **Dr. Winston Royce** IEEE August 1970
- 1975: Iterative Enhancement
- 1988: Spiral Development Model
- Early 1990s: RAD, Crystal Methods, Lean, DSDM
- 1993: Capability Maturity Model
- **1996: Scrum**
- 1998: Extreme Programming (XP)
- **2001: Agile Manifesto**
- 2002: Agile Modeling
- **2004: Microsoft's first use of Kanban for software engineering**
- 2007: Substantial Kanban growth following 2007 Agile conference

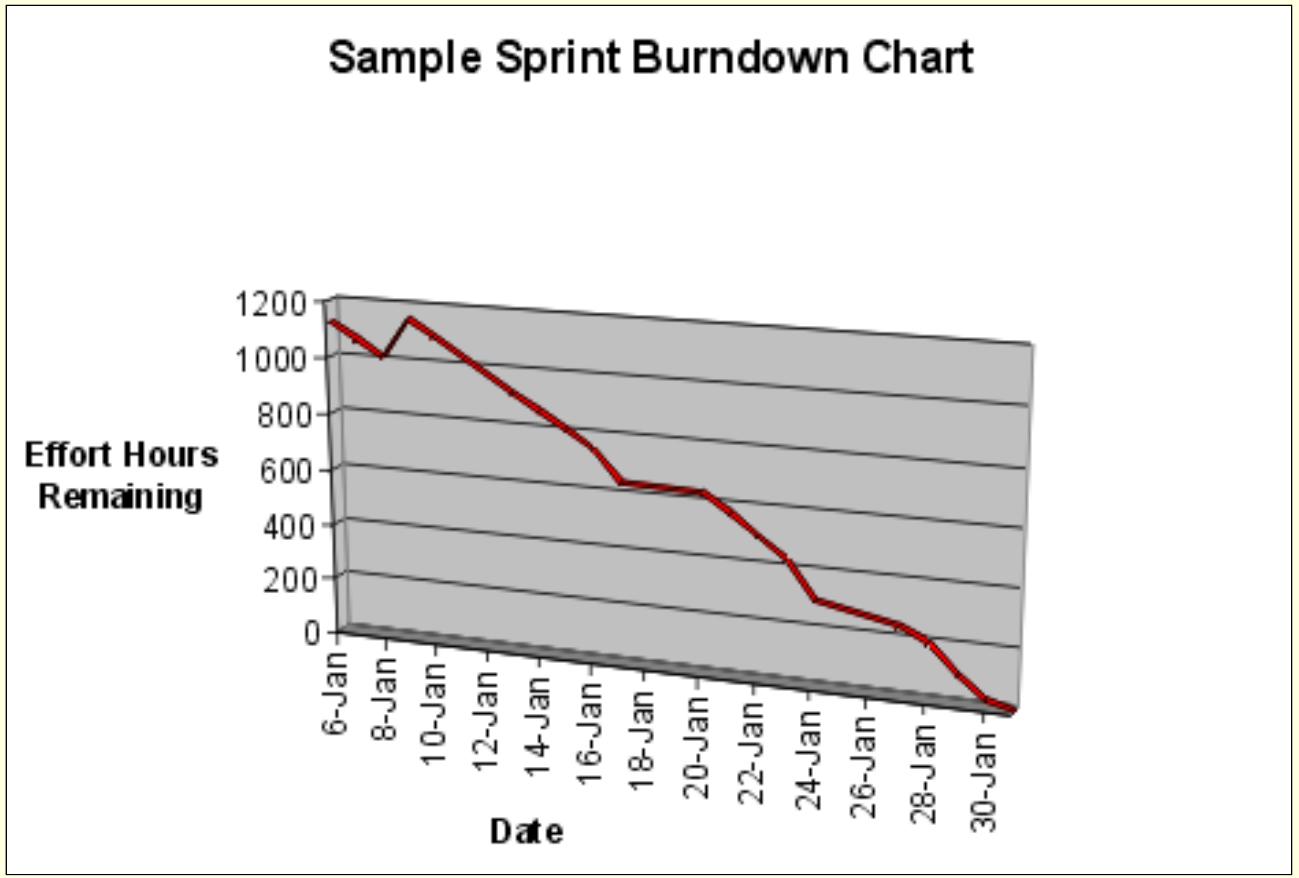
Check out the "**SCRUM in under 10 minutes**" video on the internet

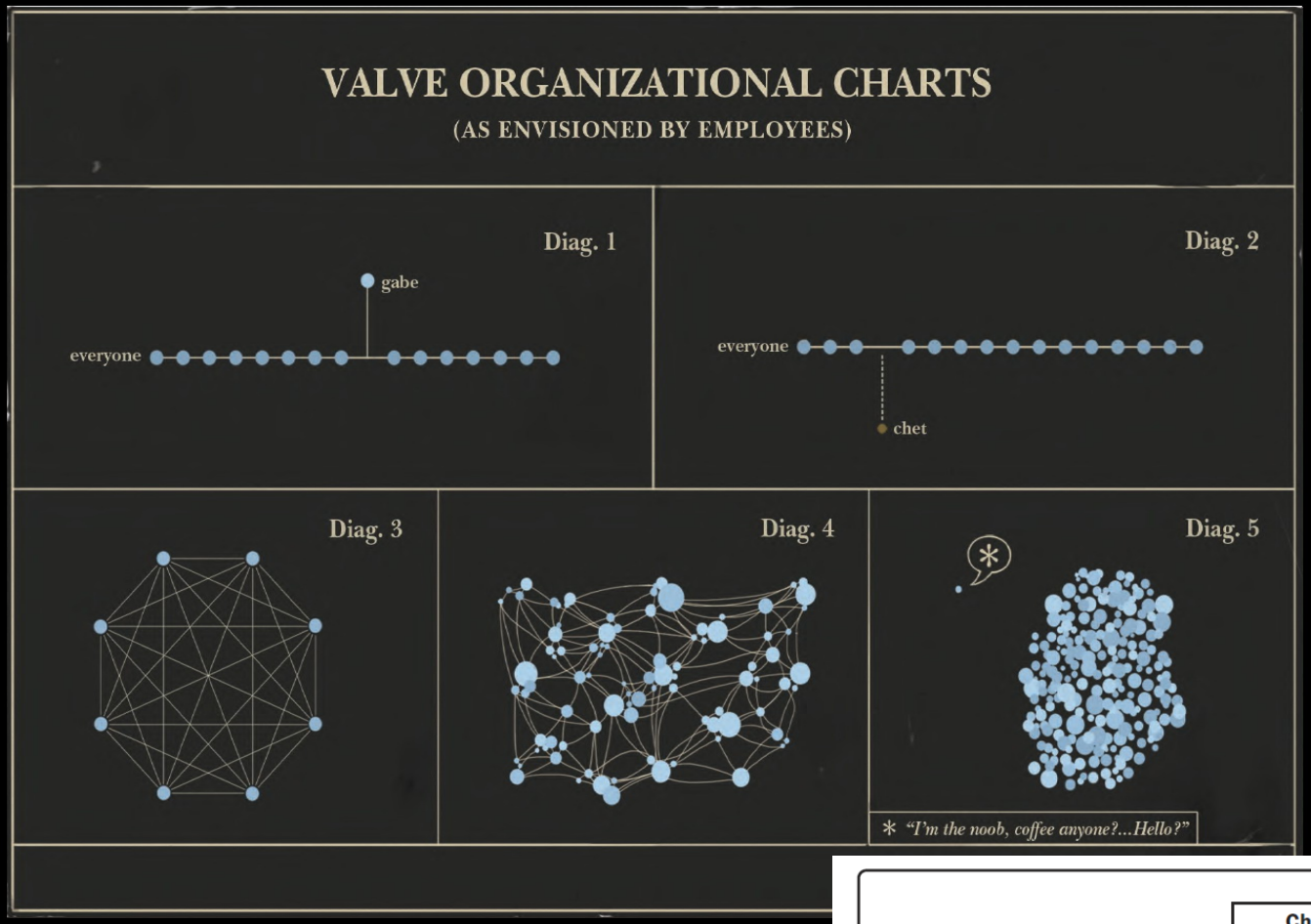


Ideally, self-organizing

Sprint Burndown Charts

- We update this chart periodically (every day is ideal) and use the slope of the line to forecast a completion date.





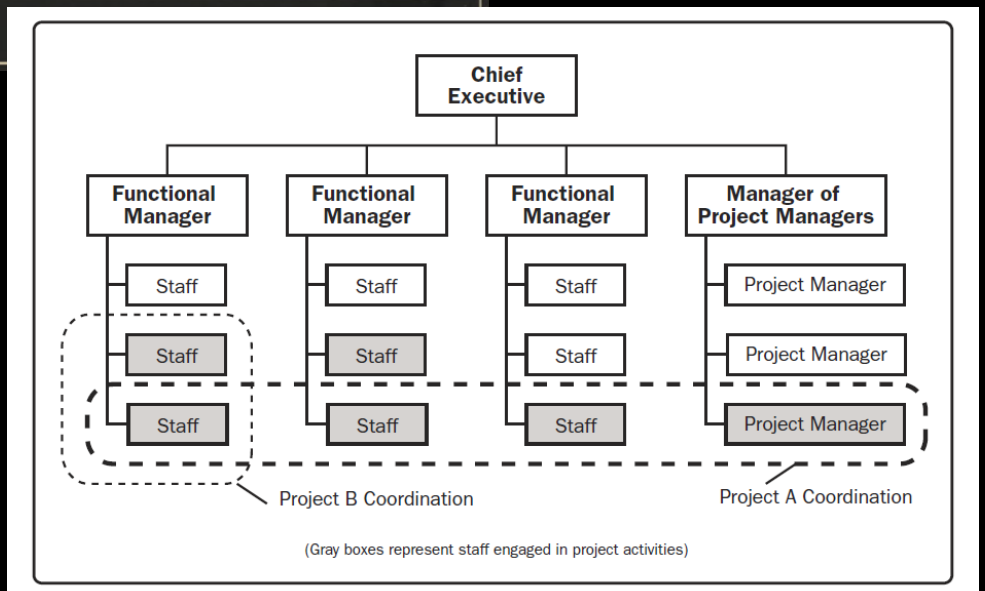


Figure 2-6. Composite Organization

lean

The 7 Wastes

Identified in the Toyota Production System

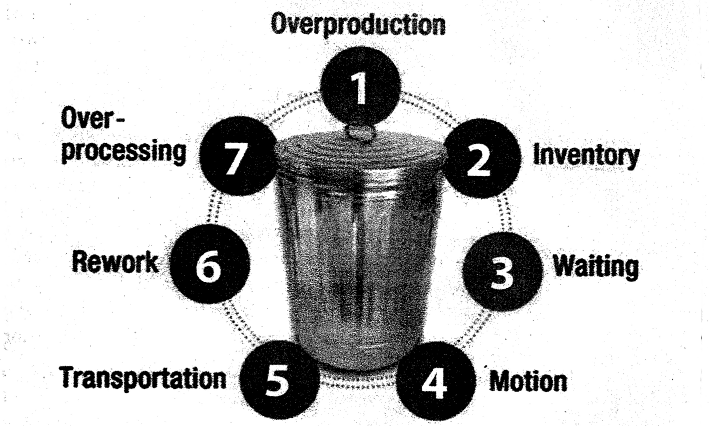


Figure 1.1 The seven lean wastes of software development: the seven wastes were identified in manufacturing in the Toyota Production System by Shigeo Shingo, then *translated* to software development by Mary and Tom Poppendieck

Table 1.1 Lean wastes: Shigeo Shingo (1981) identified seven wastes of manufacturing and Mary and Tom Poppendieck (2007) translated these into waste in software development. For an agile BA with keen eyes toward eliminating waste, this list offers a gold mine of opportunity

The Seven Wastes of Manufacturing	The Seven Wastes of Software Development
Overproduction	Partially done work
Inventory	Extra features
Extra processing	Relearning
Transportation	Handovers
Motion	Task switching
Waiting	Delays
Defects	Defects

Combating Waste

To better understand agile principles and concepts, it is important to review the ways lean attempts to fight waste. For that, we need to check the underlying reasons behind the waste and think about ways to reduce or eliminate that waste.

The 10 key practices:

Eliminate Waste
– especially things that don't add value to the final deliverable

Minimize Inventory
– intermediate artifacts like requirements & design documents

Maximize Flow – use iterative development

Pull from Demand – support flexible requirements

Empower Workers
– tell developers what needs to be done not how to do it

Meet Customer Requirements

Do it right the first time – test early

Abolish Local Optimization – flexibly manage scope

Partner with Suppliers – avoid adversarial relationships

Create a culture of Continuous Improvement

THE CORE OF KANBAN – 6 Practices:

Visualize the Workflow

Do you have a workflow?

Make a station for each step.

Limit Work in Progress (WIP)

Where was unfinished work piling up?
Select a WIP Limit
Mark stations with each limit.

Limit Work in Progress (WIP)

Determine normal lead time.
Use a point system that rewards fiished products and punishes unfinished.

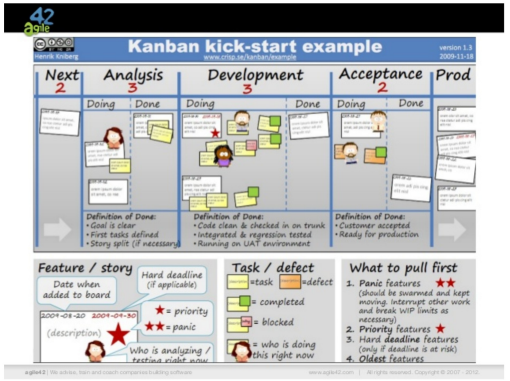
Implement Feedback Loops

Are you Regularly Reviewing Your Work and Processes?

Hold Standups & Retrospectives

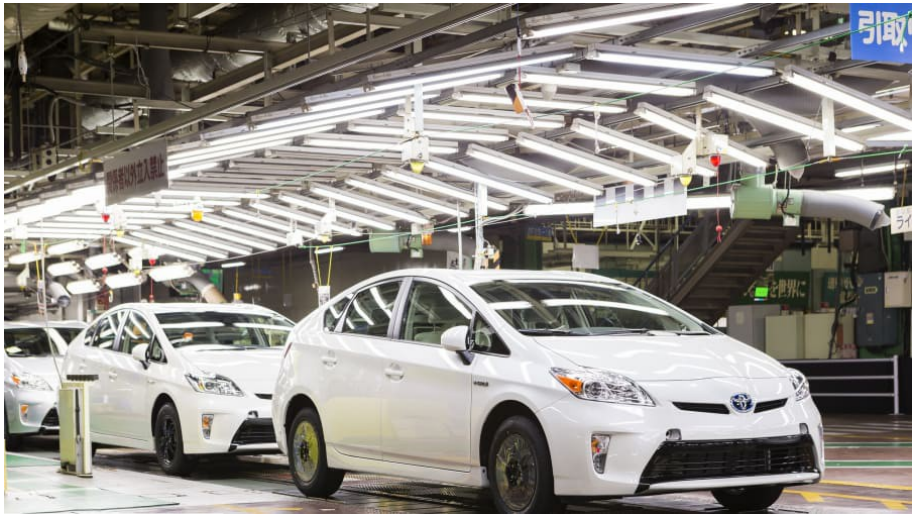
Make Process Policies Explicit

Make it easy for people to do the right things, and do them right.



Improve Collaboratively

Use metrics and models to continuously improve.



Questions ?

Quick Kanboard Demo

[next screen]

[the end]