Lean & Agile

Diego Pacheco

About Me

- ❑ Cat's Father
- ❑ Principal Software Architect
- ❑ Agile Coach
- ❑ SOA/Microservices Expert
- ❑ DevOps Practitioner
- ❑ Speaker
- ❑ Author

diegopacheco

@diego_pacheco

http://diego-pacheco.blogspot.com.br/

Diego Pacheco

Building Applications with Scala

Write modern, scalable, and reactive applications with the power of Scala

Packt>

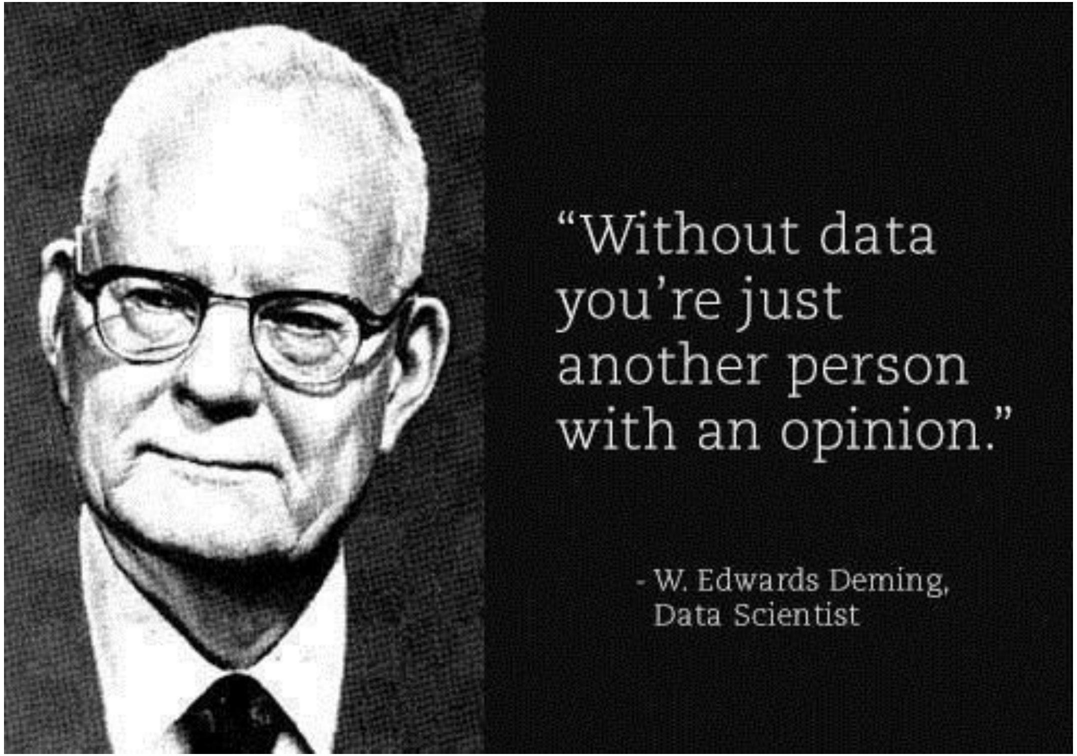
Diego Pacheco

Building Effective Microservices

Explore microservices and their implementation hands-on

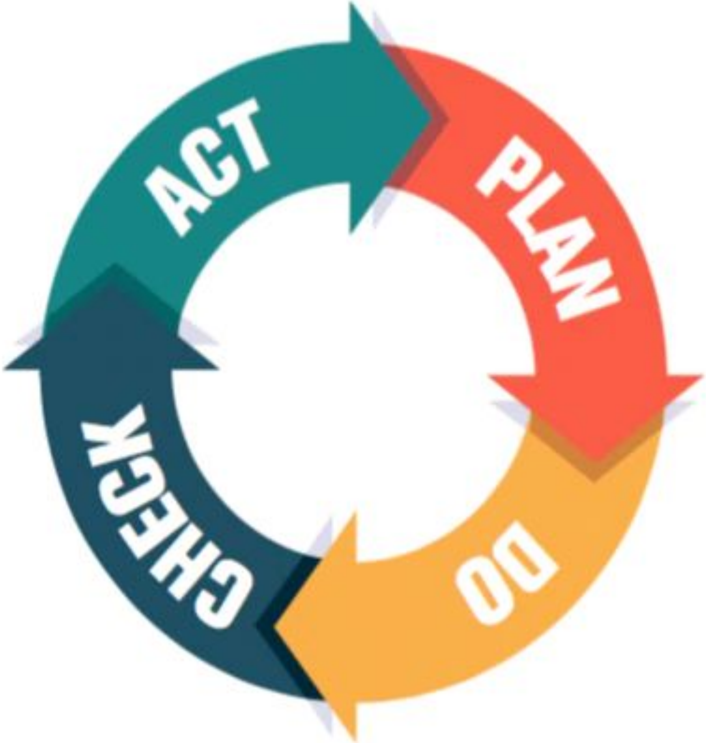
Packt>

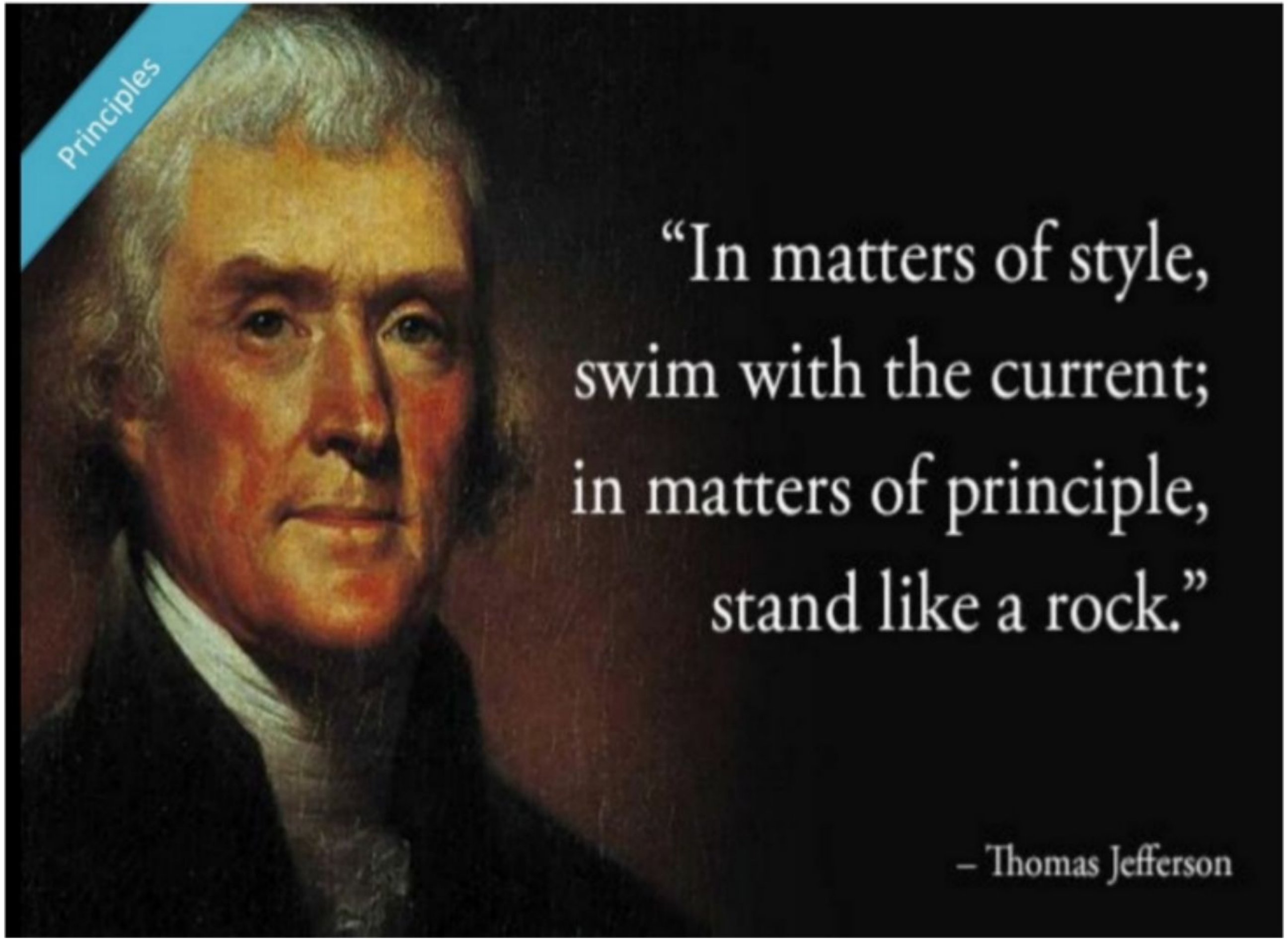
Lean == Deming



"Without data you're just another person with an opinion."

- W. Edwards Deming, Data Scientist

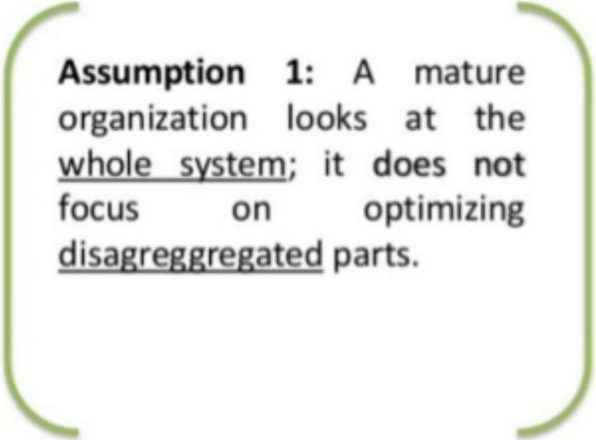




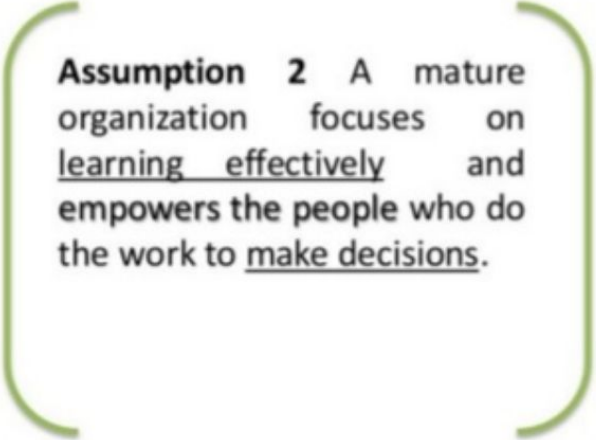
"In matters of style,
swim with the current;
in matters of principle,
stand like a rock."

– Thomas Jefferson

Lean: Mature Organization



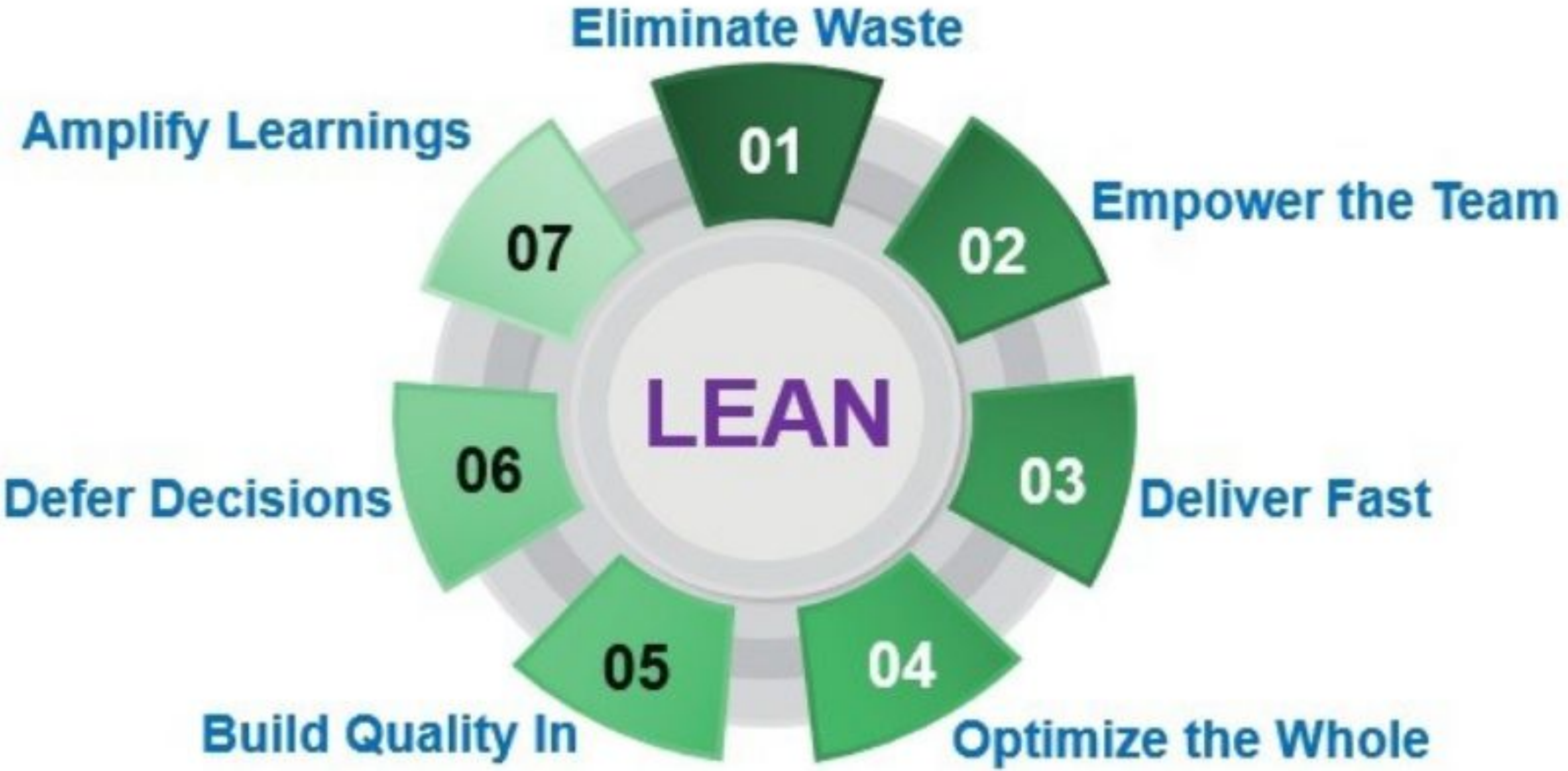
Assumption 1: A mature organization looks at the whole system; it **does not** focus on optimizing disagreggregated parts.



Assumption 2 A mature organization focuses on learning effectively and **empowers the people** who do the work to make decisions.

Lean

Why do it at all ?
Remove Waste



LEAN

01 Eliminate Waste

02 Empower the Team

03 Deliver Fast

04 Optimize the Whole

05 Build Quality In

06 Defer Decisions

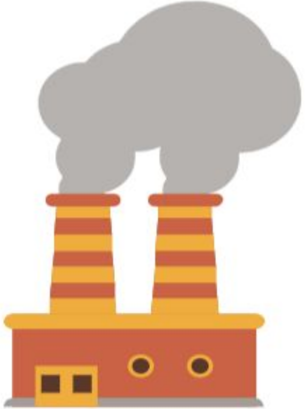
07 Amplify Learnings

7 Wastes of Lean

Inventory

Waiting

Defects

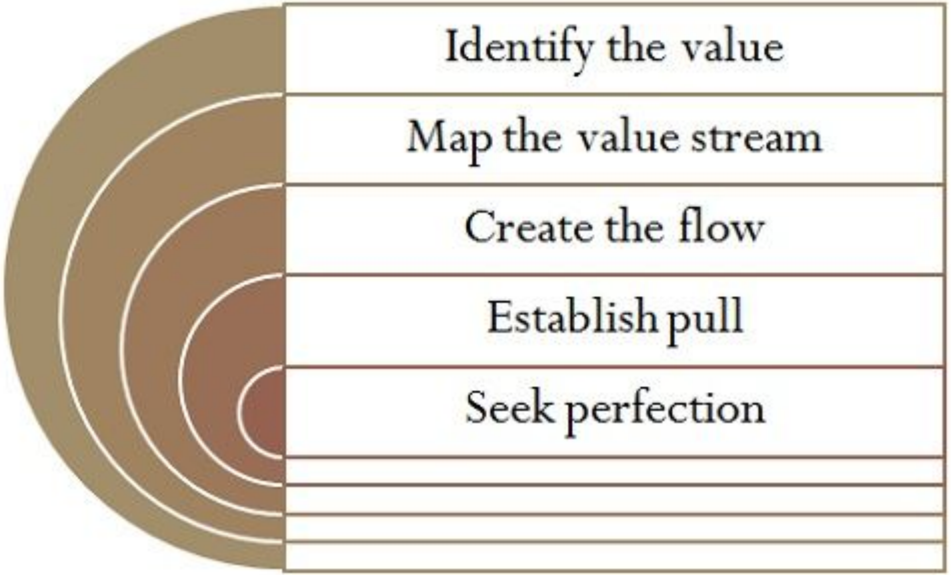


Overproduction

Motion

Transportation

Over-processing



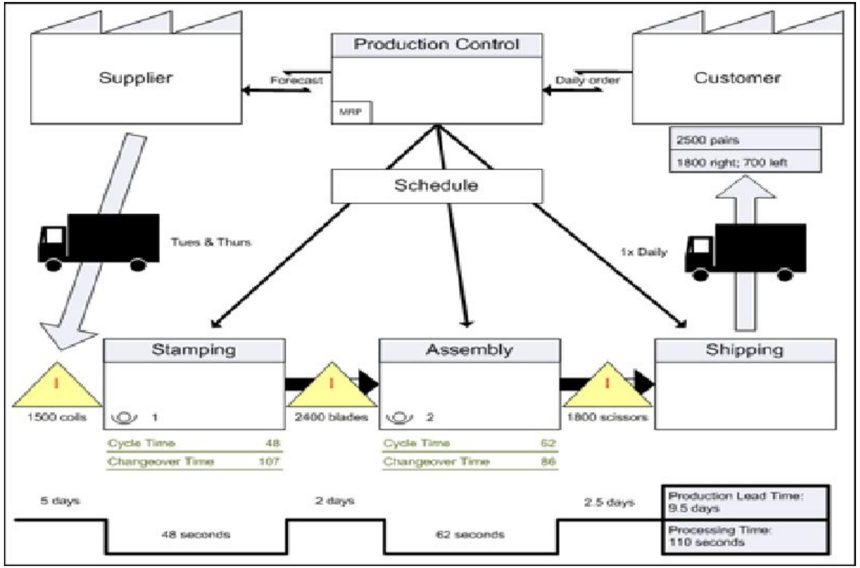
Identify the value

Map the value stream

Create the flow

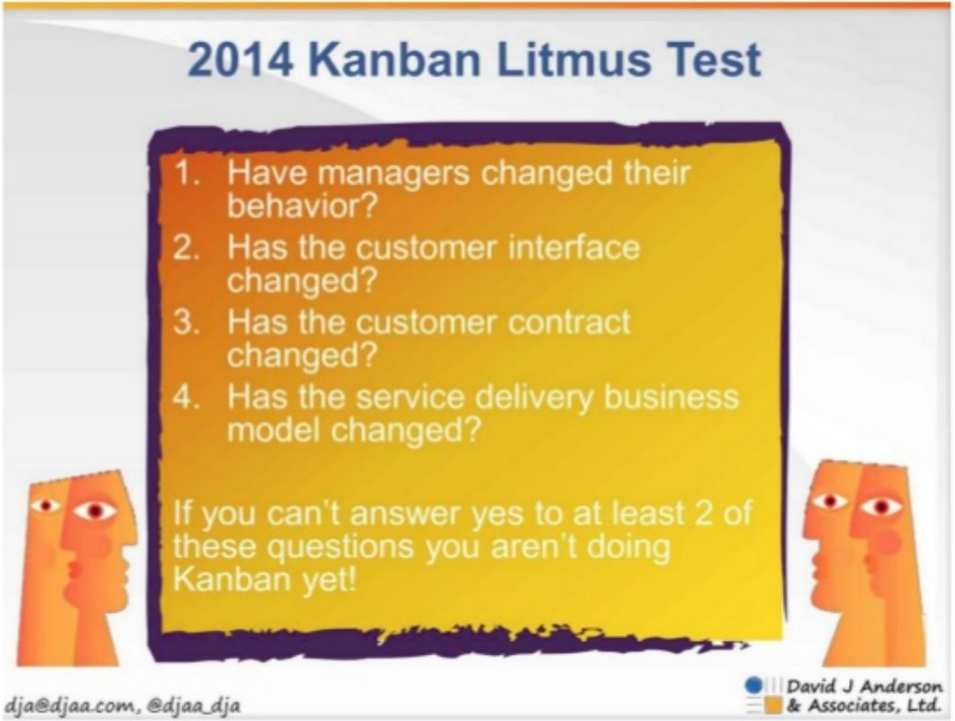
Establish pull

Seek perfection



How do I test If I'm doing Kanban?

ESSENTIAL
KANBAN
CONDENSED
KANBAN
CHANGE MANAGEMENT
Start with what you do NOW
Agree to pursue improvement through EVOLUTIONARY CHANGE
Encourage acts of LEADERSHIP at every level
SERVICE DELIVERY
Understand and focus on CUSTOMER needs and expectations
Manage the WORK let people self-organize around it
Evolve POLICIES to improve outcomes
Visualize
Limit WiP
Manage flow
Make policies explicit
Feedback loops
Improve & evolve
PRINCIPLES
PRACTICES
DAVID J ANDERSON
ANDY CARMICHAEL



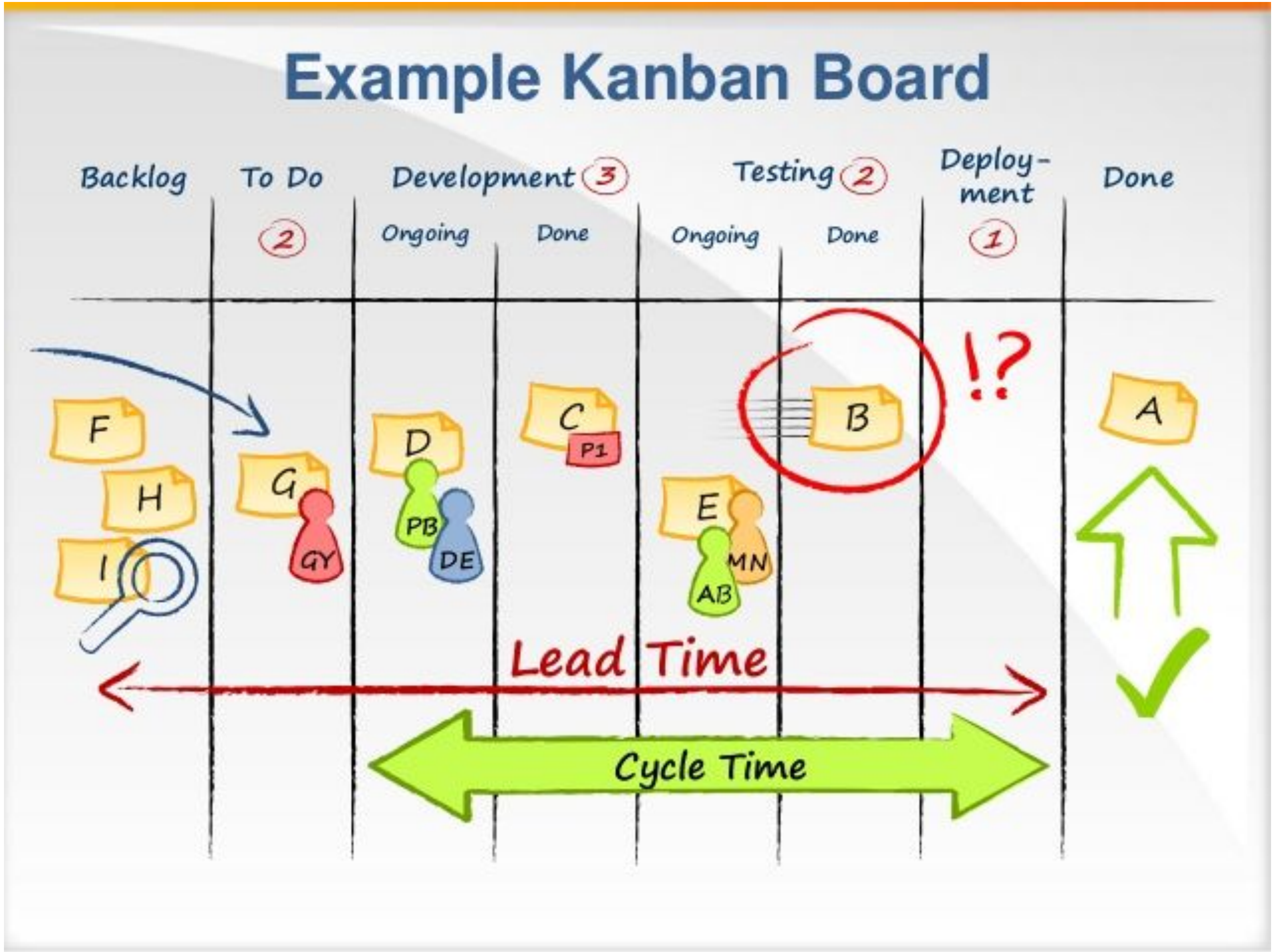
2014 Kanban Litmus Test

1. Have managers changed their behavior?
2. Has the customer interface changed?
3. Has the customer contract changed?
4. Has the service delivery business model changed?

If you can't answer yes to at least 2 of these questions you aren't doing Kanban yet!

dja@djaa.com, @djaa_dja

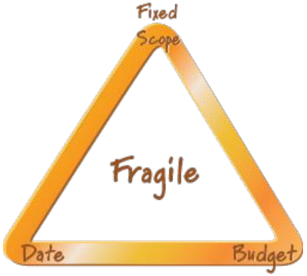
David J Anderson & Associates, Ltd.



Example Kanban Board

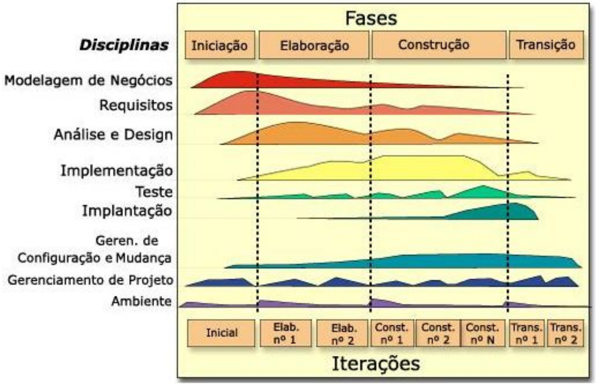
90s..2k

CMMI

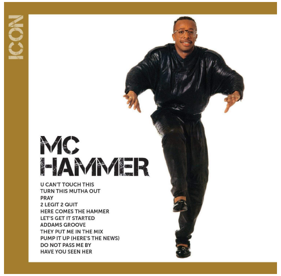


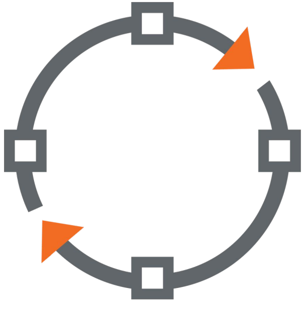
Software Factory

RUP



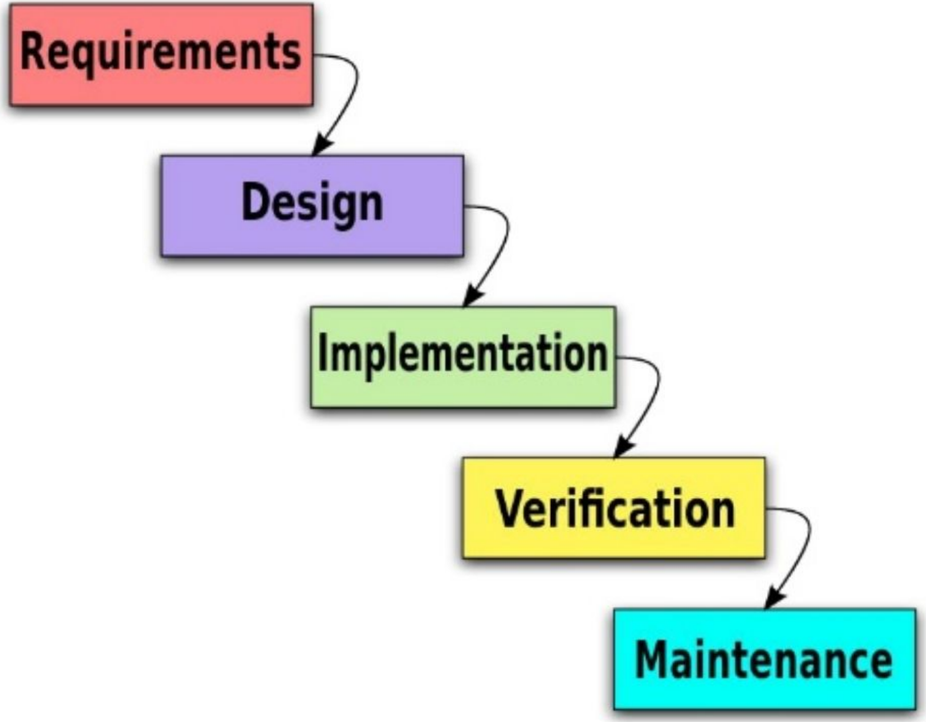
PMI®
Project Management Institute



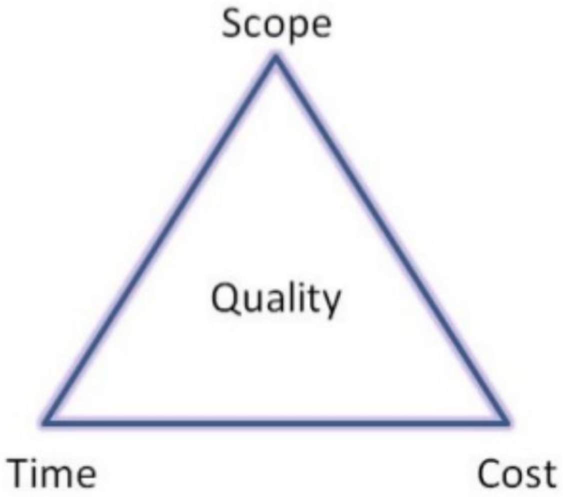


PROCESS

Waterfall



- ❑ What is software?
- ❑ Buy X Sell
- ❑ ALL FIXED
- ❑ Low Feedback
- ❑ BDUP
- ❑ Dev X Ops X ...

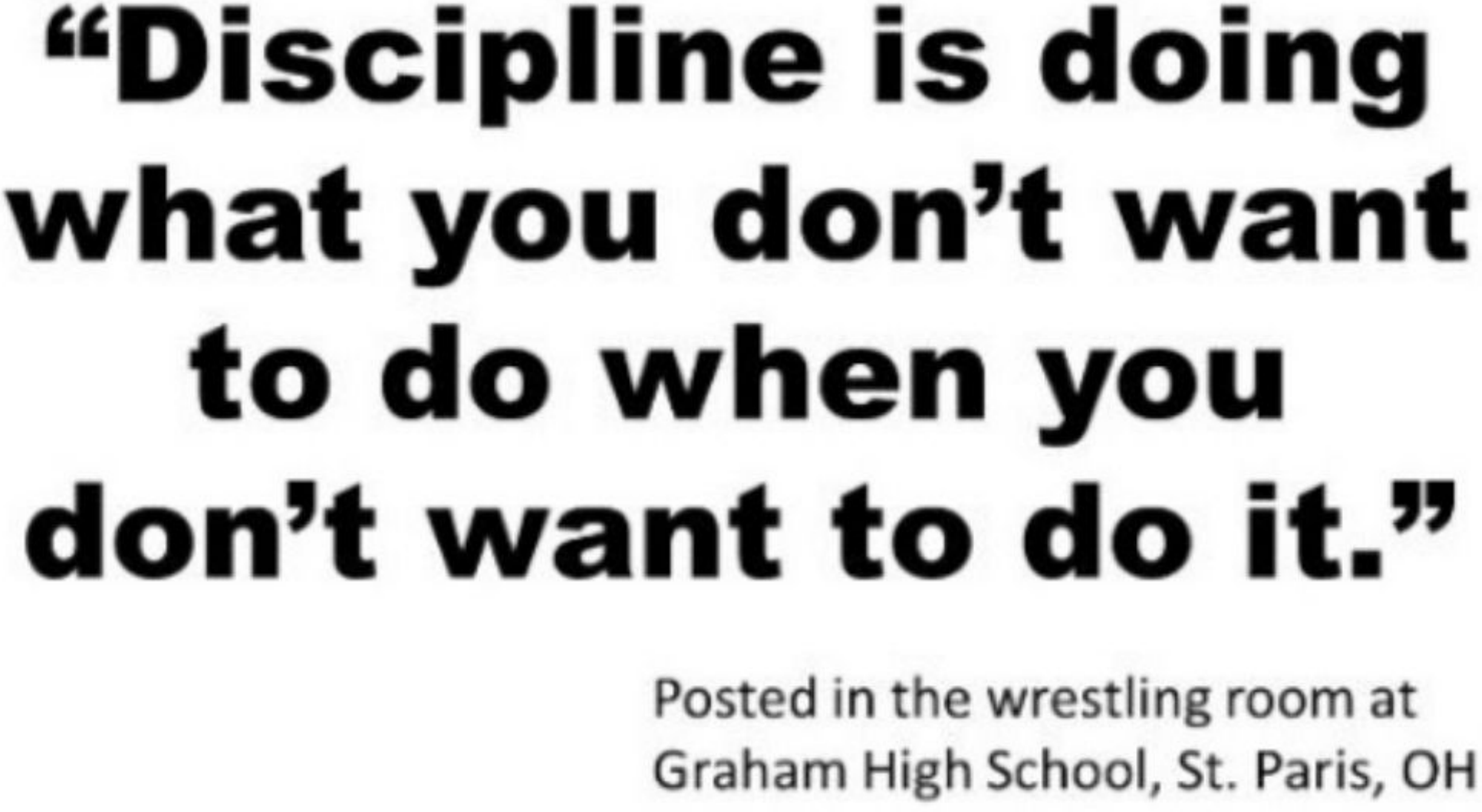


Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

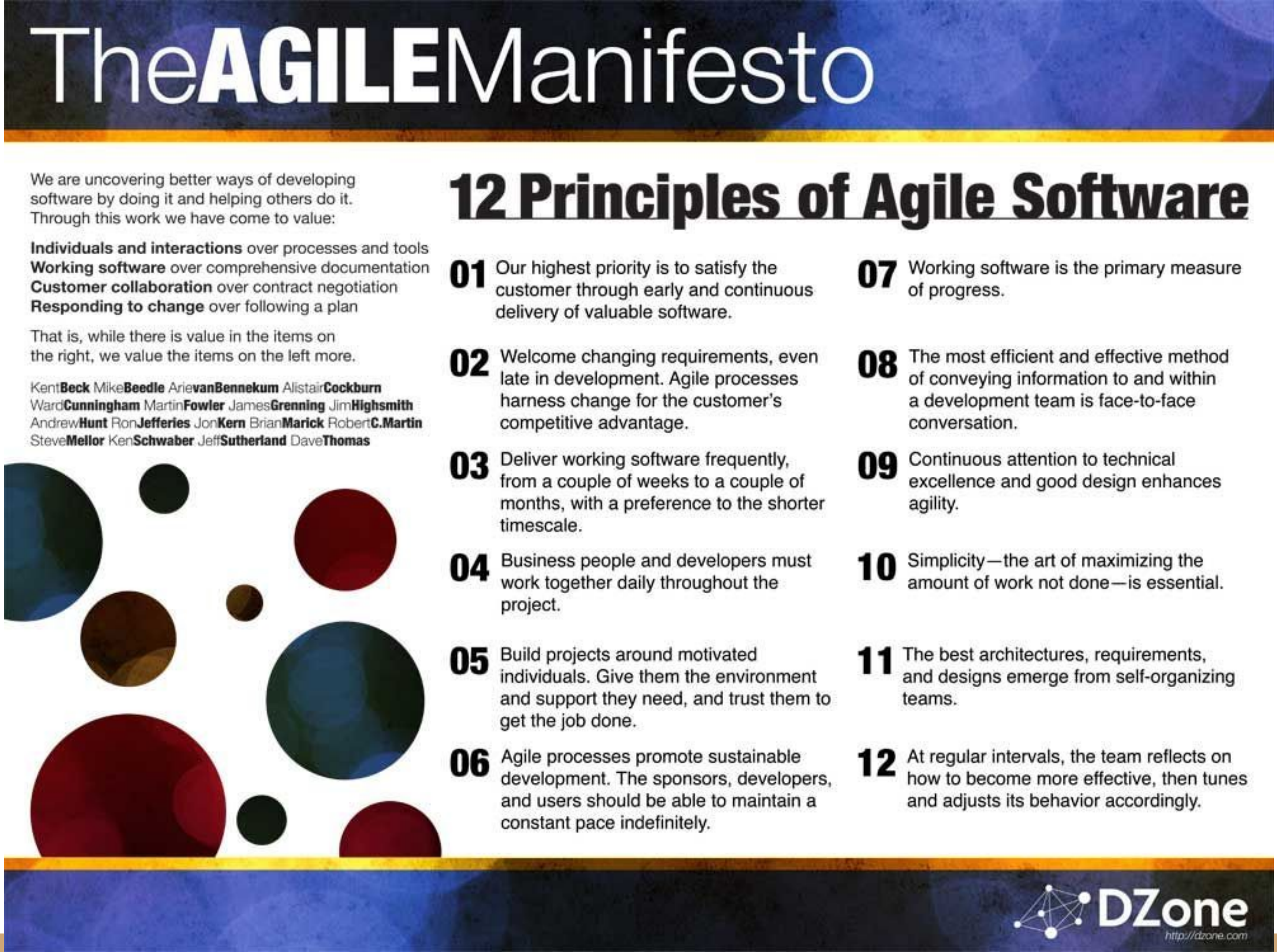
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.



“Discipline is doing what you don’t want to do when you don’t want to do it.”

Posted in the wrestling room at
Graham High School, St. Paris, OH



The**AGILE**Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent**Beck** Mike**Beedle** Arie**vanBennekum** Alistair**Cockburn** Ward**Cunningham** Martin**Fowler** James**Grenning** Jim**Highsmith** Andrew**Hunt** Ron**Jefferies** Jon**Kern** Brian**Marick** Robert**C.Martin** Steve**Mellor** Ken**Schwaber** Jeff**Sutherland** Dave**Thomas**

12 Principles of Agile Software

01 Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

02 Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

03 Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

04 Business people and developers must work together daily throughout the project.

05 Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

06 Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

07 Working software is the primary measure of progress.

08 The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

09 Continuous attention to technical excellence and good design enhances agility.

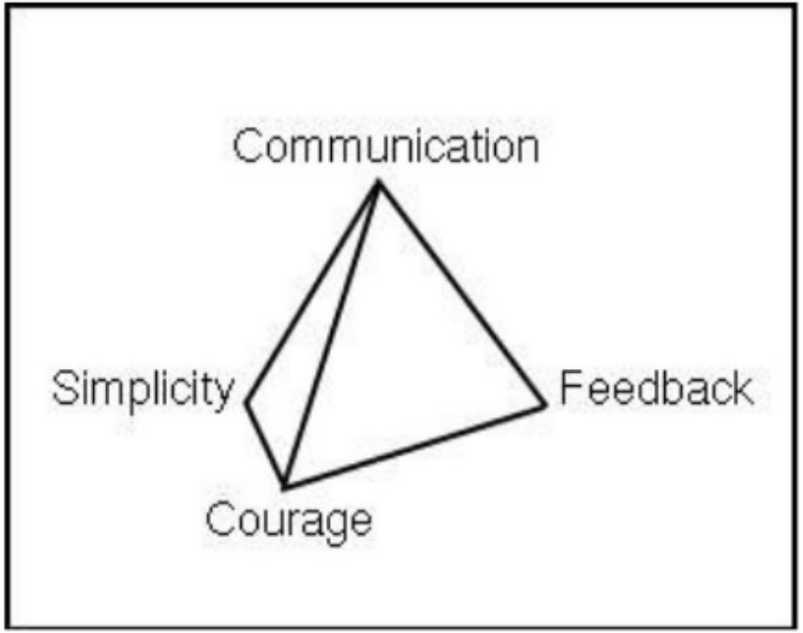
10 Simplicity—the art of maximizing the amount of work not done—is essential.

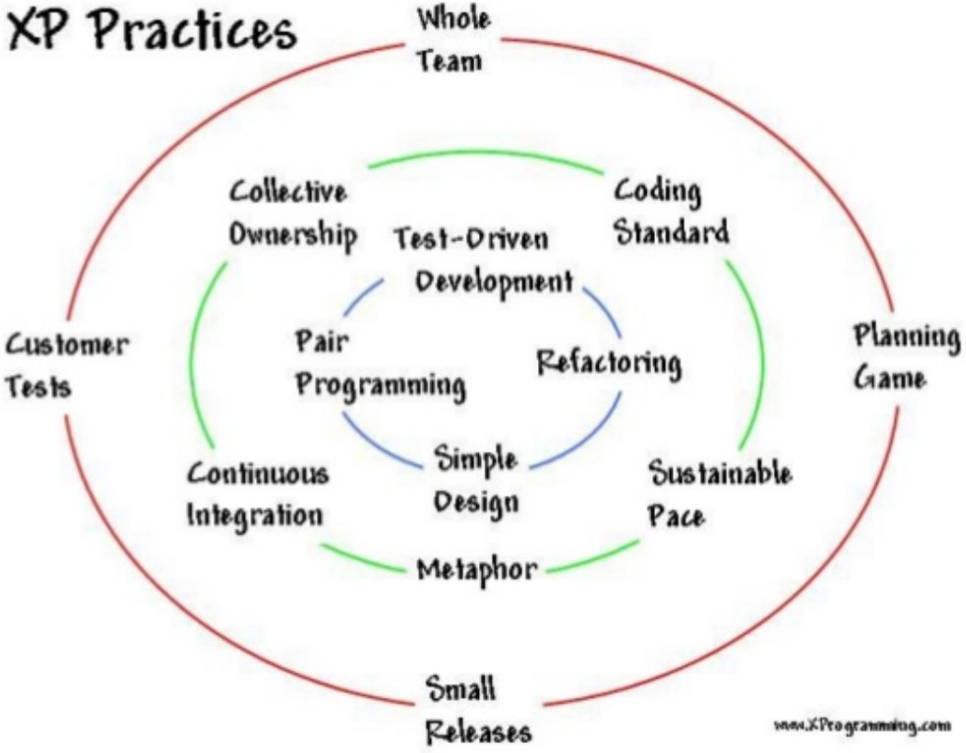
11 The best architectures, requirements, and designs emerge from self-organizing teams.

12 At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

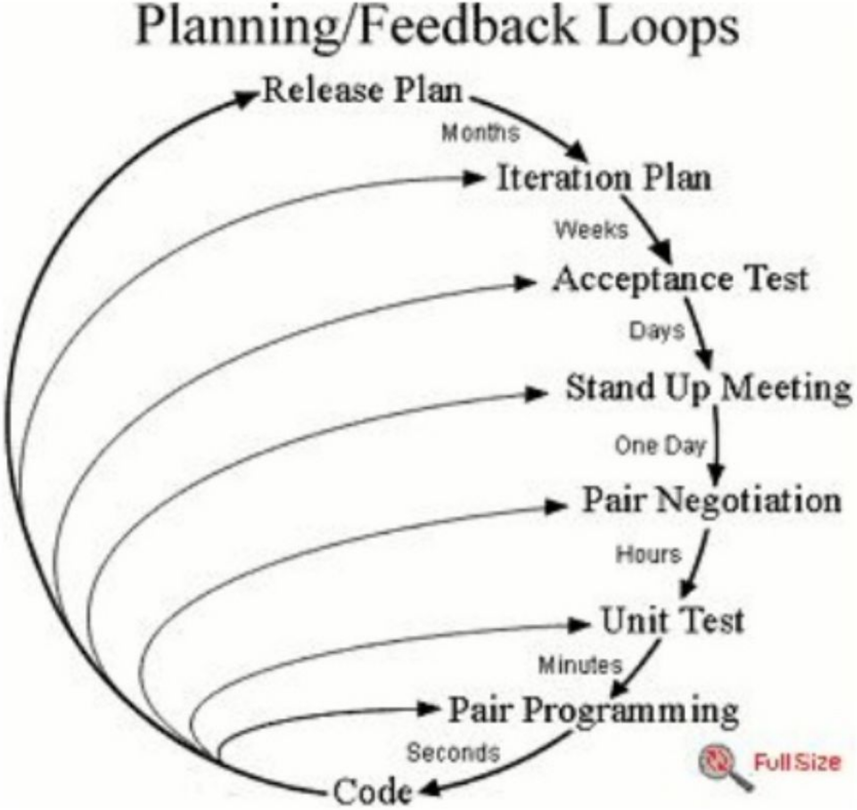
DZone
http://dzone.com

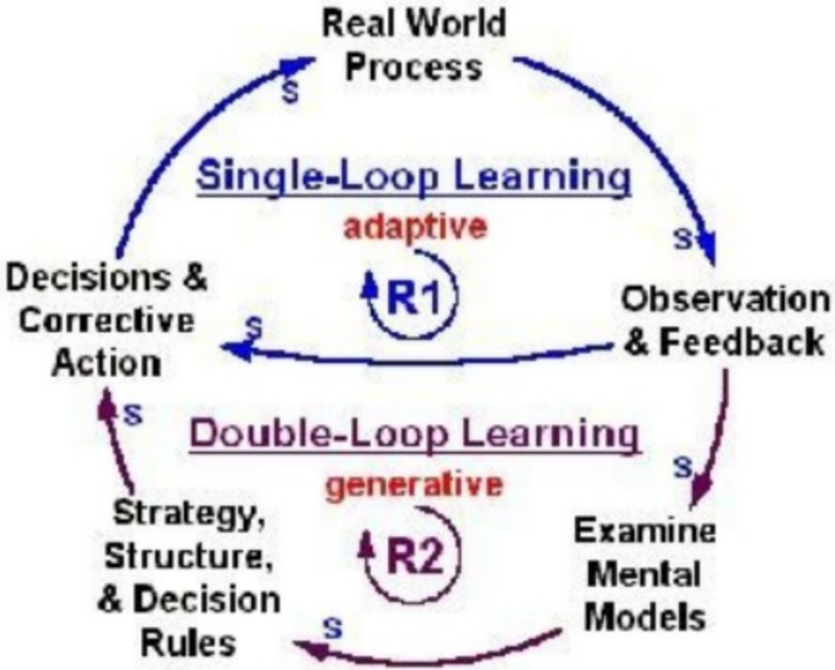
XP


Communication
Simplicity
Feedback
Courage


XP Practices
Whole Team
Collective Ownership
Coding Standard
Test-Driven Development
Customer Tests
Pair Programming
Refactoring
Planning Game
Simple Design
Continuous Integration
Sustainable Pace
Metaphor
Small Releases
www.XProgramming.com

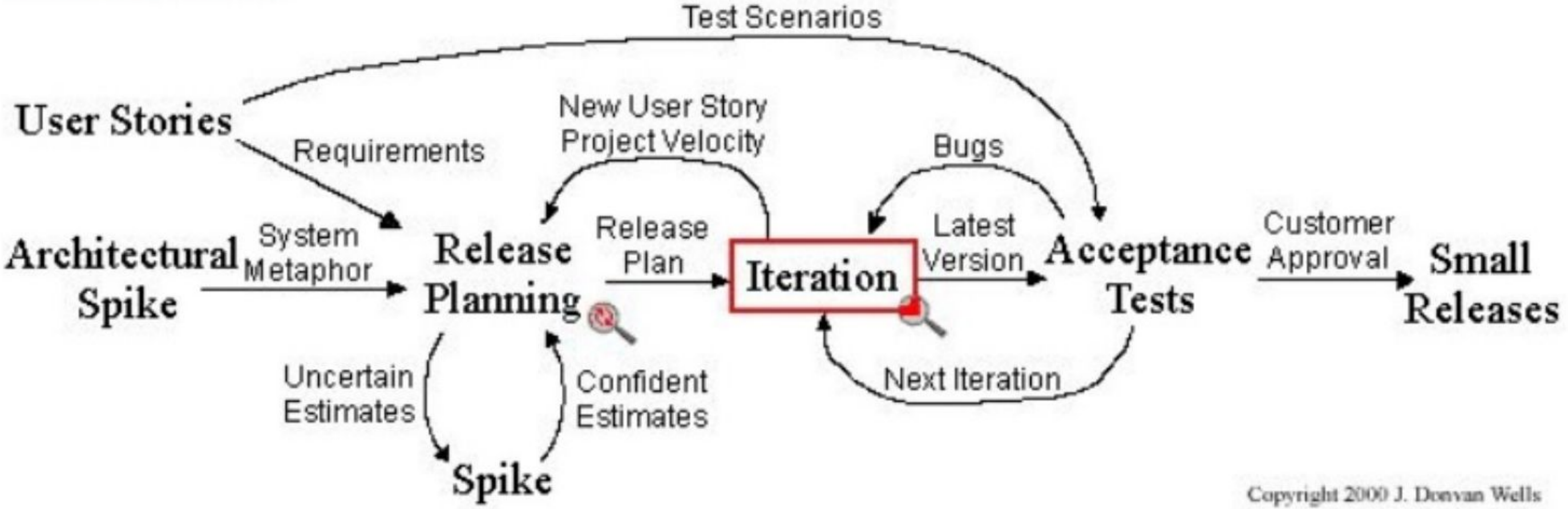
XP


Planning/Feedback Loops
Release Plan
Months
Iteration Plan
Weeks
Acceptance Test
Days
Stand Up Meeting
One Day
Pair Negotiation
Hours
Unit Test
Minutes
Pair Programming
Seconds
Code
Full Size


Real World Process
S
Single-Loop Learning
adaptive
R1
S
Decisions & Corrective Action
S
Observation & Feedback
S
Double-Loop Learning
generative
R2
S
Strategy, Structure, & Decision Rules
S
Examine Mental Models
S

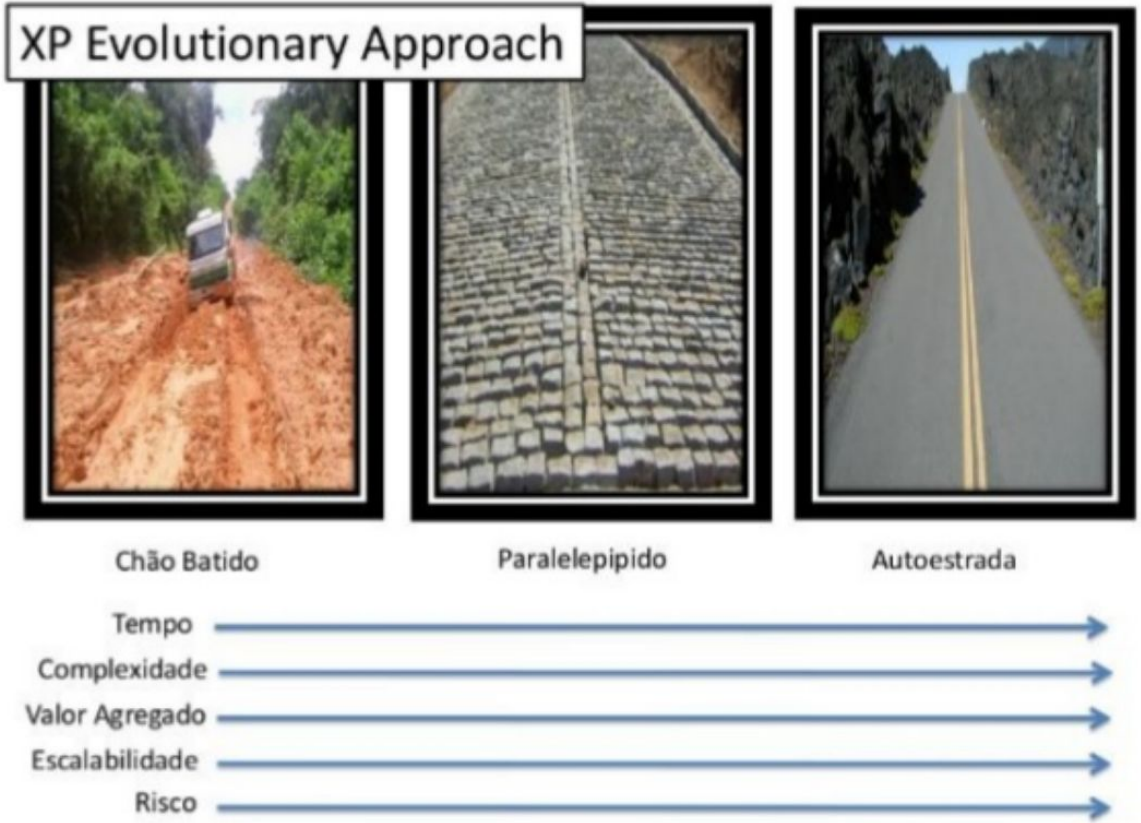
Extreme Programming

Extreme Programming Project



Copyright 2000 J. Donvan Wells

XP Dimensional Quality



XP Evolutionary Approach

Chão Batido

Paralelepipido

Autoestrada

Tempo

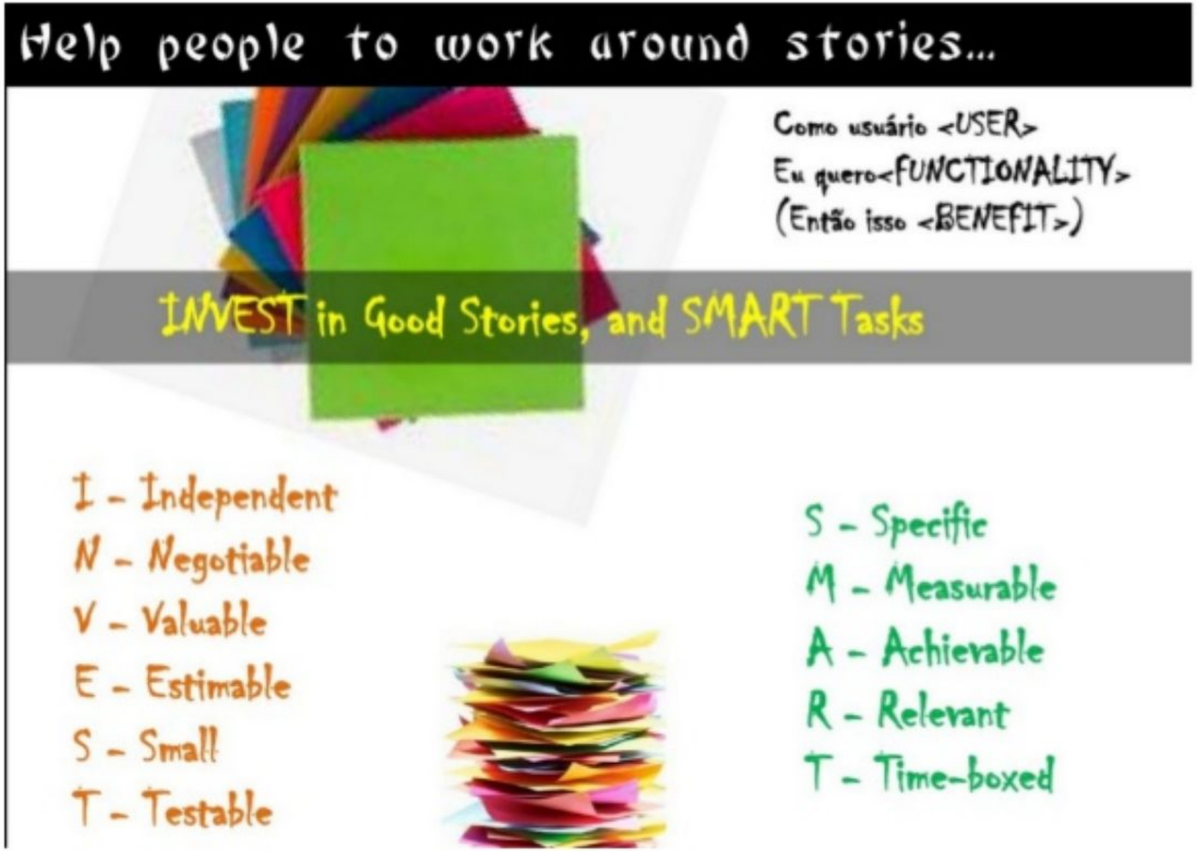
Complexidade

Valor Agregado

Escalabilidade

Risco

XP SMART / INVEST



Help people to work around stories...

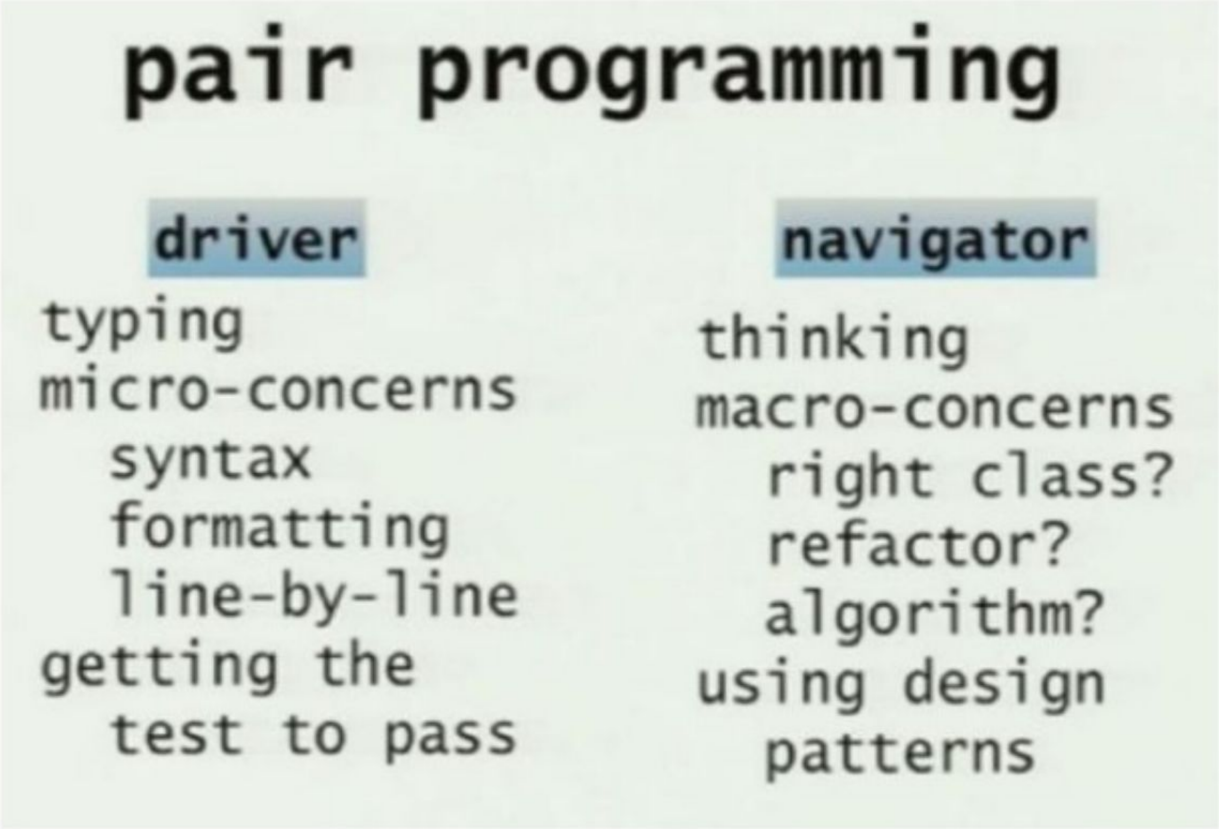
Como usuário <USER>
Eu quero<FUNCTIONALITY>
(Então isso <BENEFIT>)

INVEST in Good Stories, and SMART Tasks

I - Independent
N - Negotiable
V - Valuable
E - Estimable
S - Small
T - Testable

S - Specific
M - Measurable
A - Achievable
R - Relevant
T - Time-boxed

XP Paring



pair programming

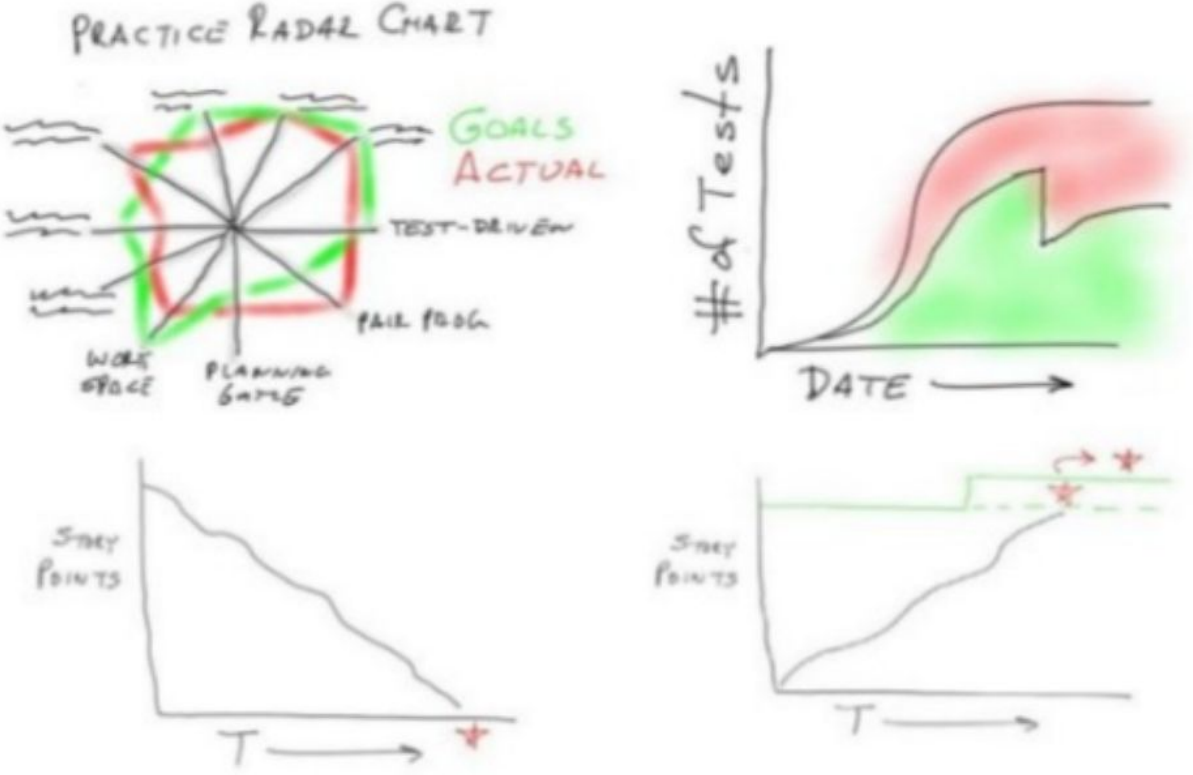
driver

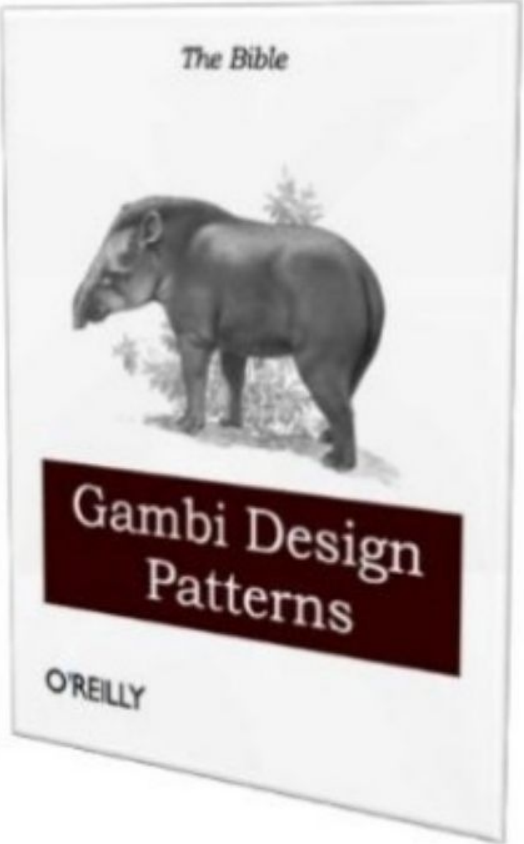
- typing
- micro-concerns
  - syntax
  - formatting
  - line-by-line
- getting the test to pass

navigator

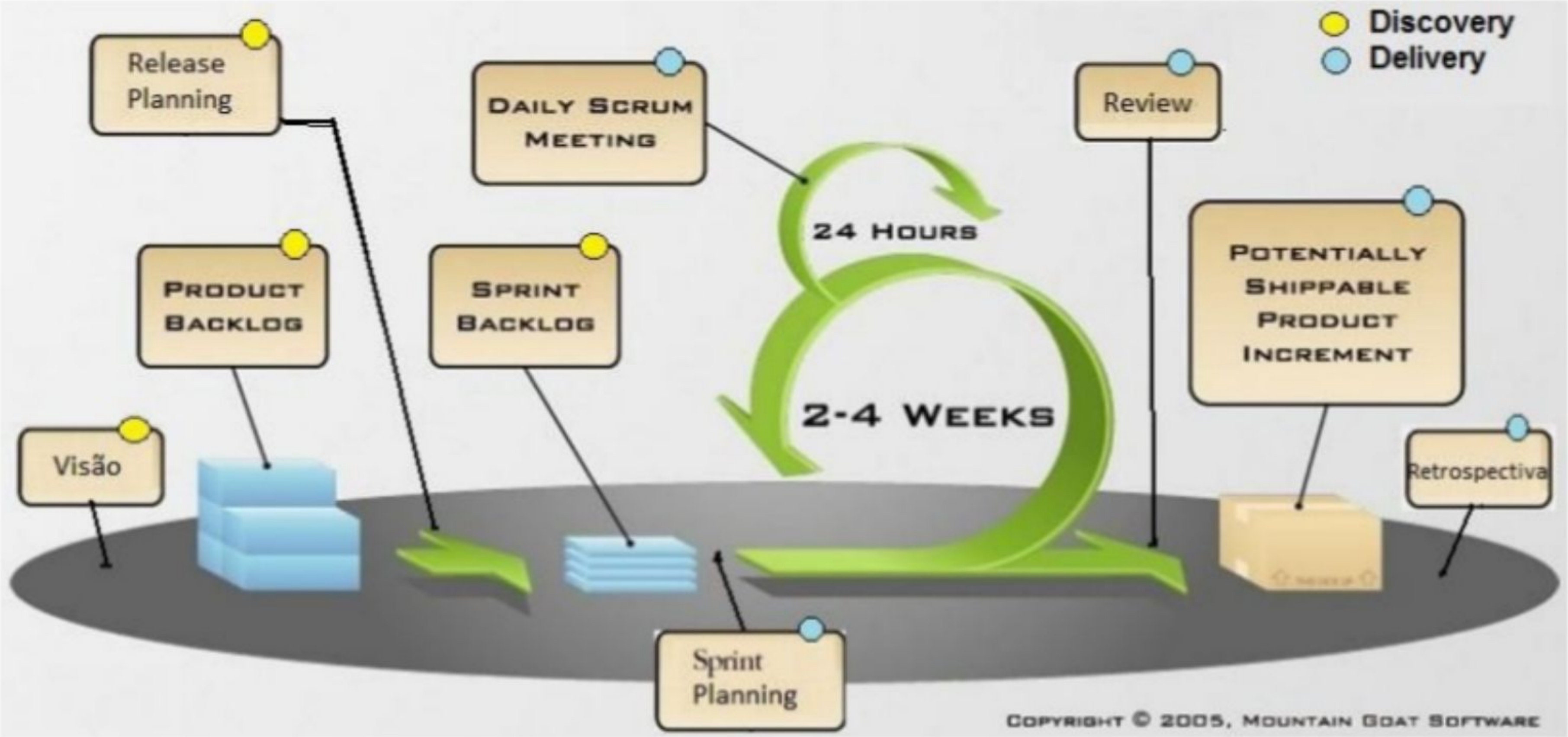
- thinking
- macro-concerns
  - right class?
  - refactor?
  - algorithm?
- using design patterns

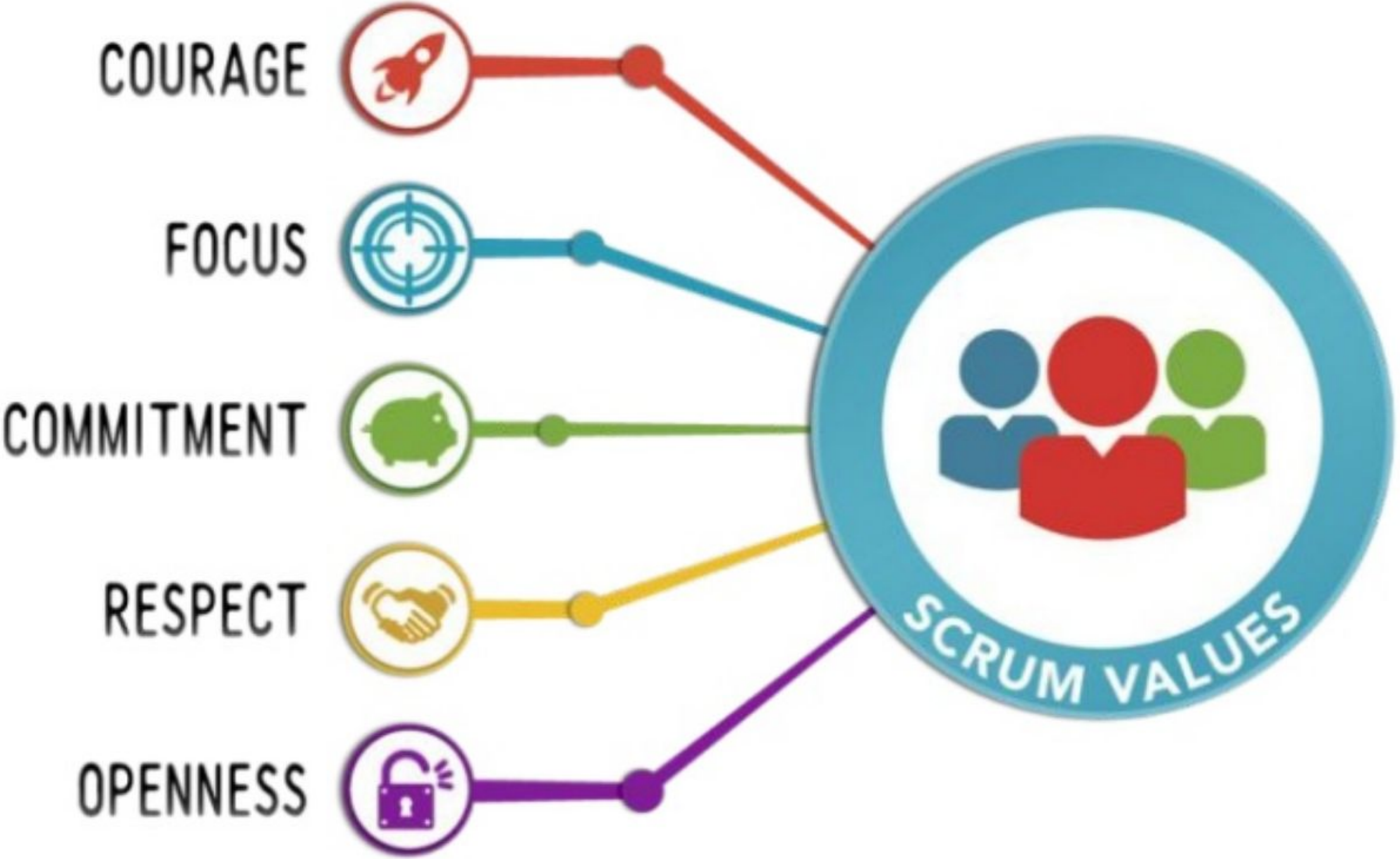
Big Visible Charts

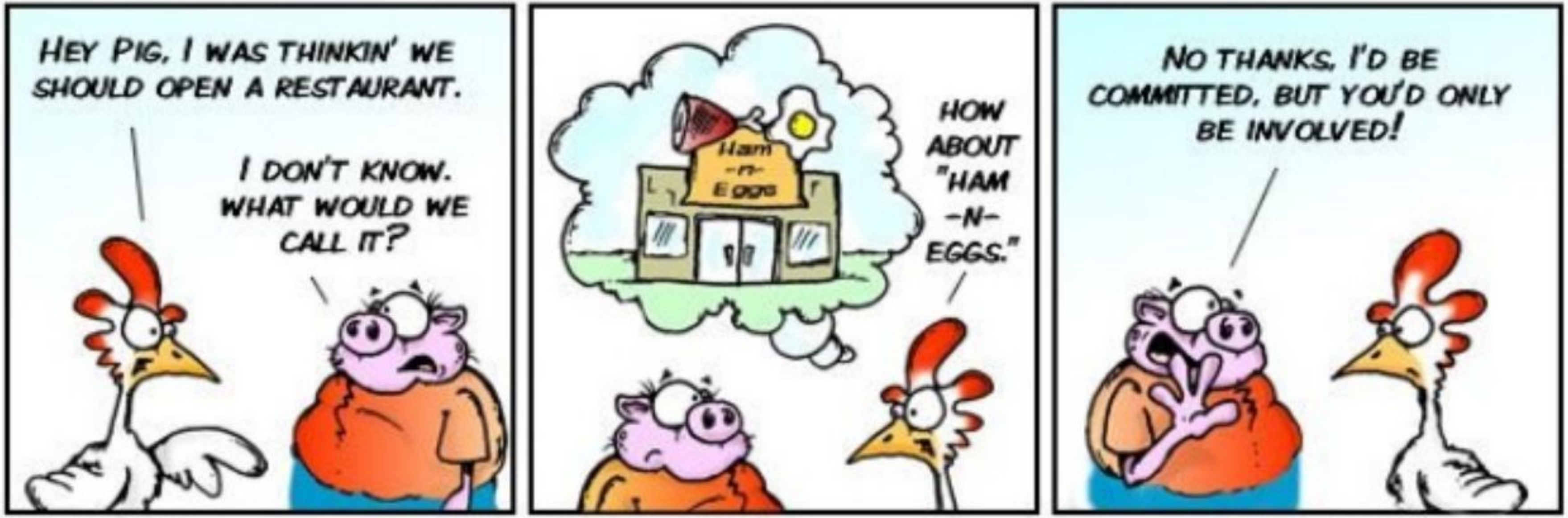




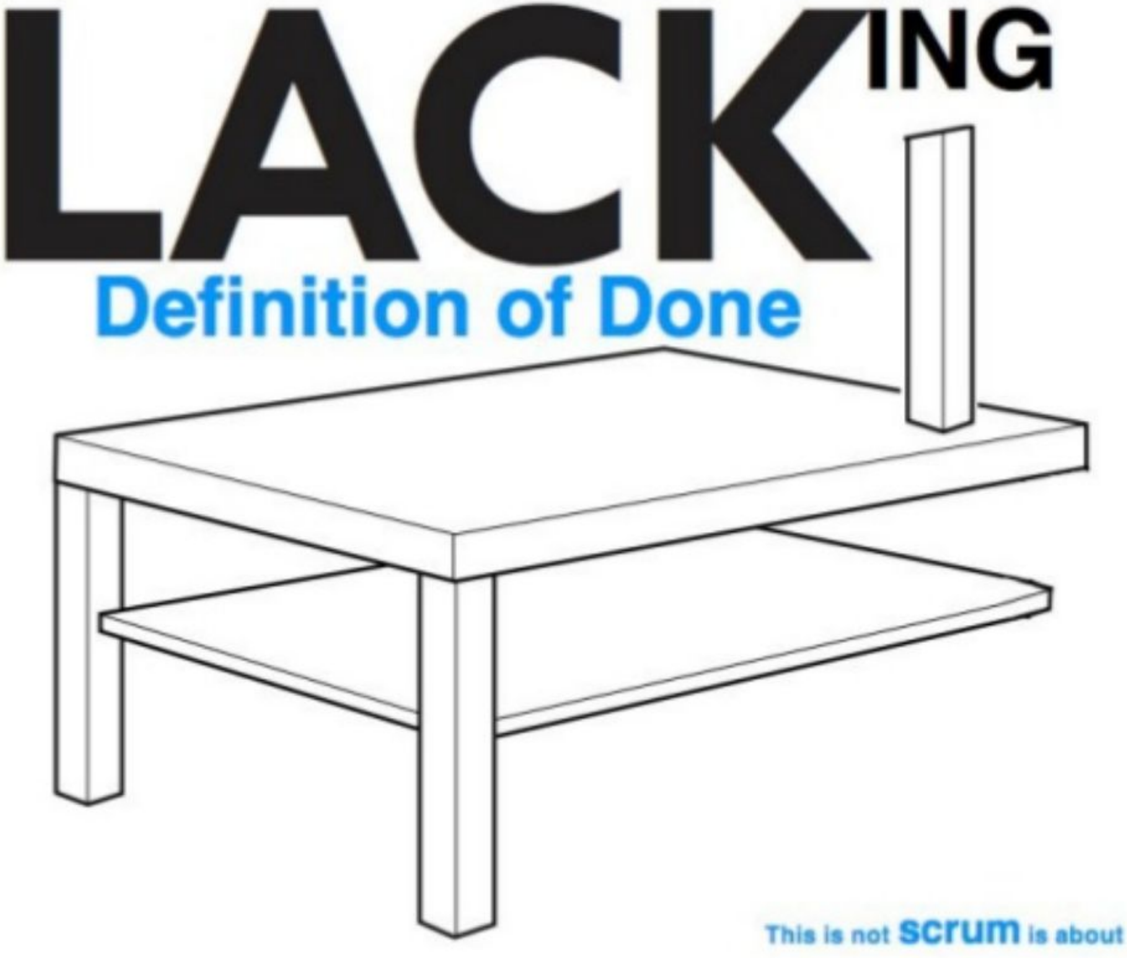
Technical Debt

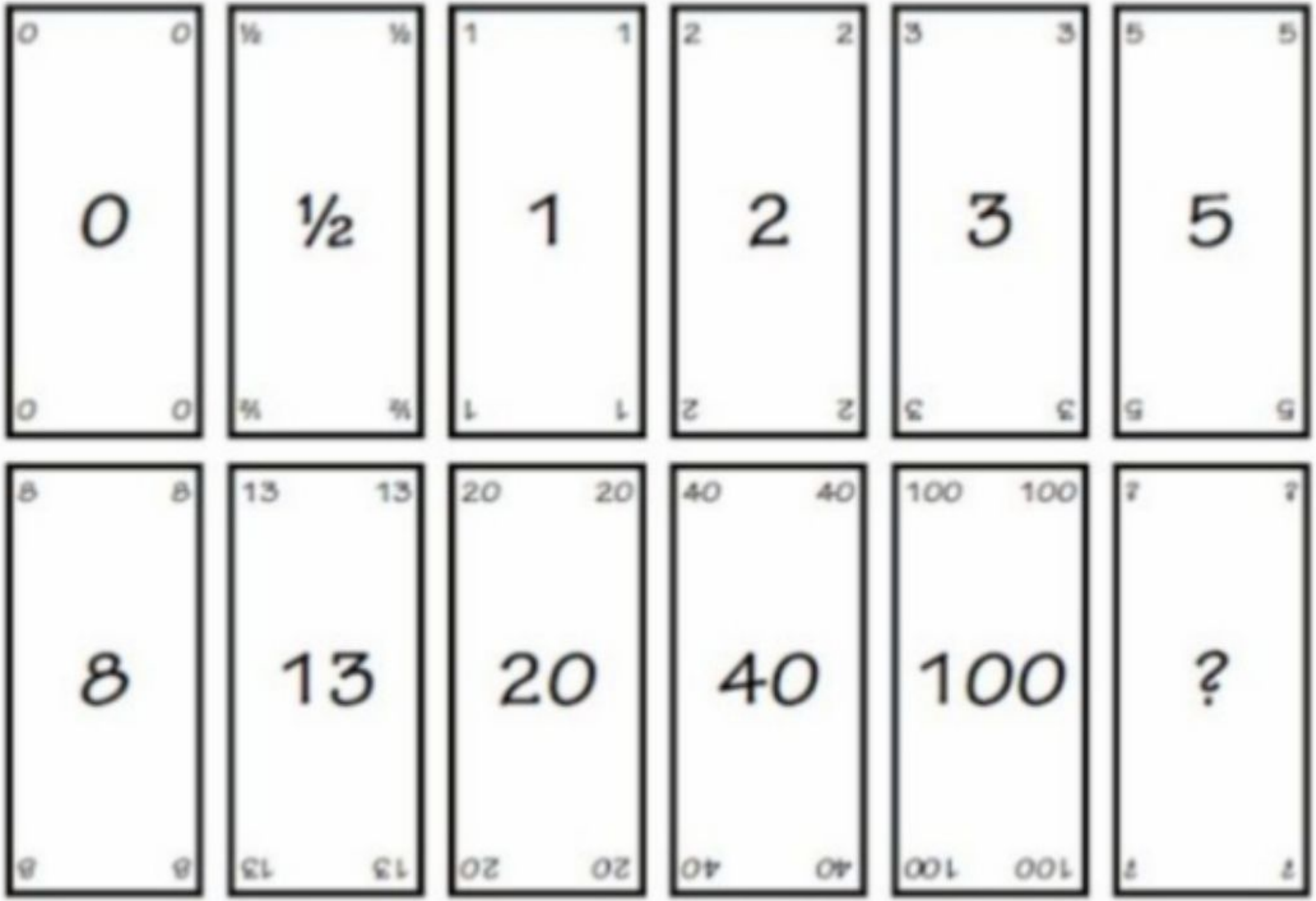

● Discovery
○ Delivery
Release Planning
Daily Scrum Meeting
24 Hours
Review
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment
Visão
Retrospectiva
Sprint Planning
Copyright © 2005, Mountain Goat Software


COURAGE
FOCUS
COMMITMENT
RESPECT
OPENNESS
SCRUM VALUES


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


LACKING
Definition of Done
This is not scrum is about


0
½
1
2
3
5
8
13
20
40
100
?

Lean & Agile

Diego Pacheco