

Come and take it!

Lean Pull Applied



Don.McGreal
@ImprovingEnterprises.com



Rod.Coffin

@ImprovingEnterprises.com



Lean?

“Most business processes are 90% waste and 10% value-added work”
- Jeffrey Liker, The Toyota Way

From a Time Magazine article i read:

In a revealing set of studies, a team led by Gloria Mark and Victor Gonzalez of the University of California at Irvine tracked 36 officeworkers—in this case information-technology workers at an investment firm—and recorded how they spent their time, minute by minute. The researchers found that the employees devoted an average of just 11 minutes to a project before the ping of an e-mail, the ring of the phone or a knock on the cubicle pulled them in another direction. Once they were interrupted, it took, on average, a stunning 25 minutes to return to the original task—if they managed to do so at all that day. The workers in the study were juggling an average of 12 projects apiece—a situation one subject described as "constant, multitasking craziness." The five biggest causes of interruption in descending order, according to Mark: a colleague stopping by, the worker being called away from the desk (or leaving voluntarily), the arrival of new e-mail, the worker switching to another task on the computer and a phone call.

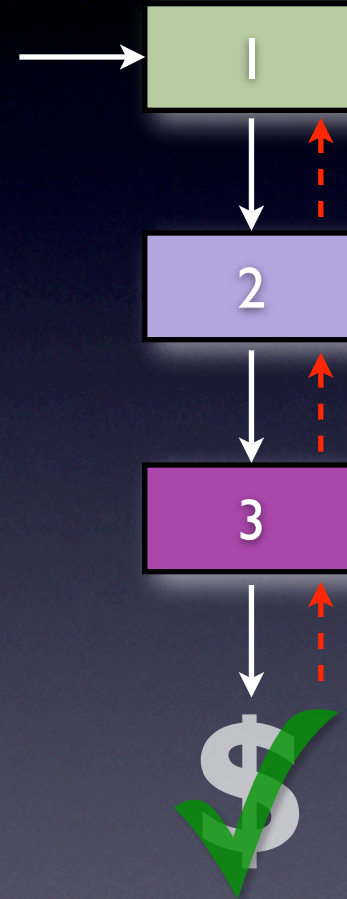
Lean?

“I believe that the average farmer puts to a really useful purpose only about 5% of the energy he expends.... Not only is everything done by hand, but seldom is a thought given to a logical arrangement. A farmer doing his chores will walk up and down a rickety ladder a dozen times. He will carry water for years instead of putting in a few lengths of pipe. His whole idea, when there is extra work to do, is to hire extra men. He thinks of putting money into improvements as an expense.... It is waste motion— waste effort— that makes farm prices high and profits low.”

- Henry Ford, My Life and Work

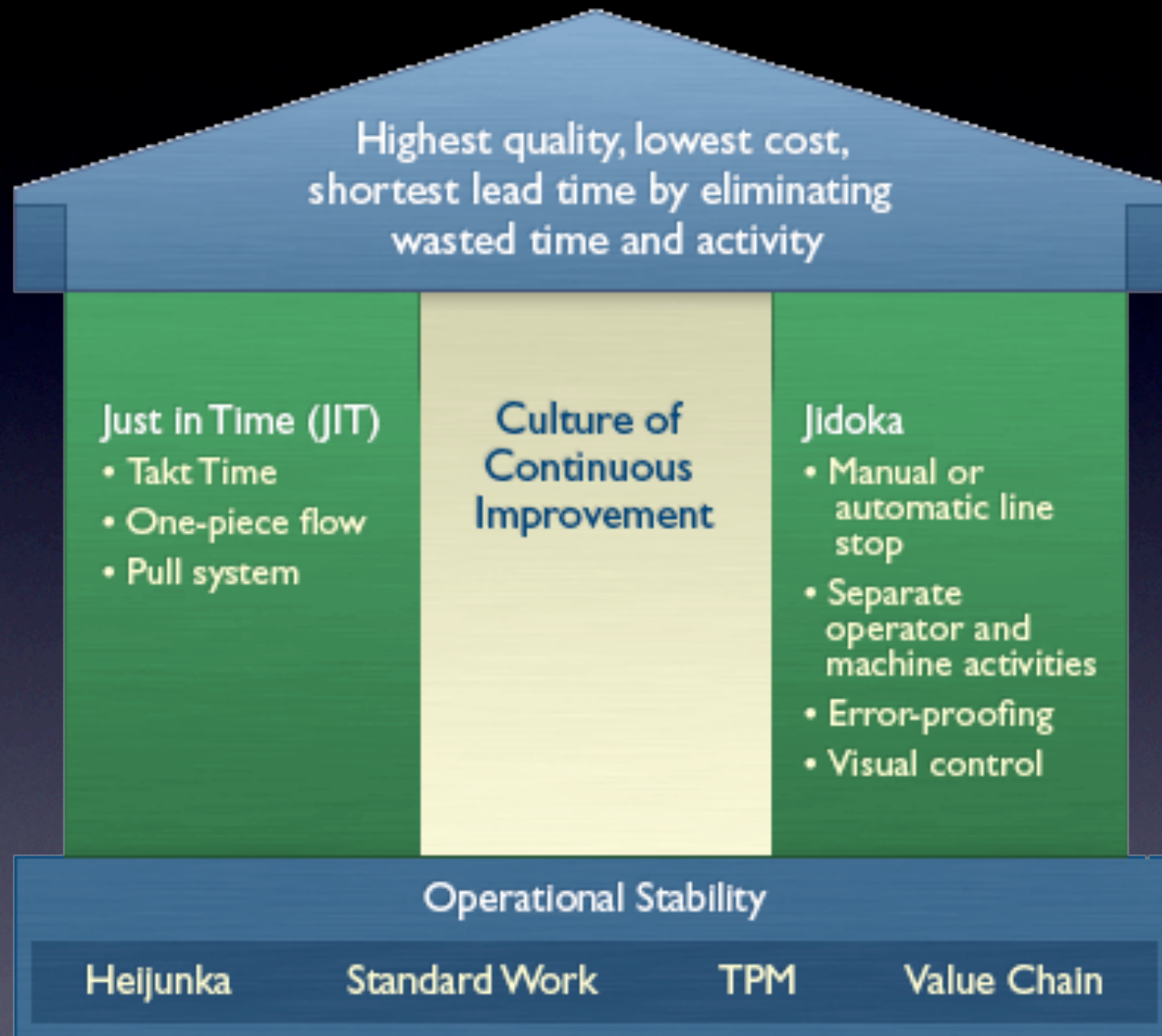
Lean Principles

- Value
- Value Stream
- Flow
- Pull
- Perfection



Lean implementation is therefore focused on getting the right things, to the right place, at the right time, in the right quantity to achieve perfect work flow while minimizing waste and being flexible and able to change.

Toyota Production System



http://www.tbmcg.com/images/about_roots/house_toyota.gif

Jidoka : "the decision to stop and fix problems as they occur rather than pushing them down the line to be resolved later"

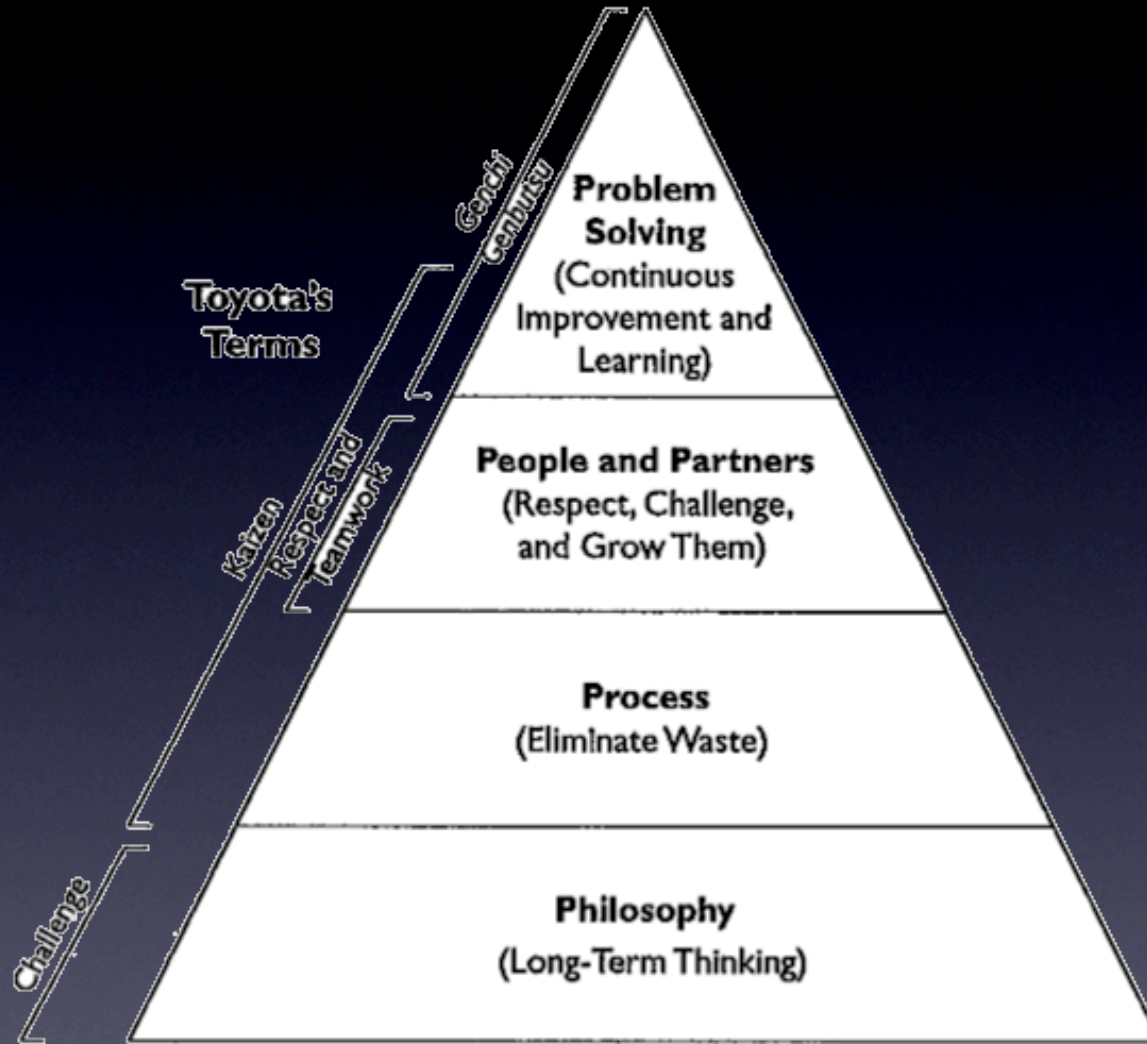
Heijunka: Production Leveling

TPM: Total Product (or Process) Management - Assign experts to stations - they should know how to fix their own problems.

The "House of Toyota"

To help explain the Toyota Production System to employees and suppliers, the "House of Toyota" graphic was created by Taiichi Ohno and Eiji Toyoda. They chose the house shape because it was a familiar one – and also conveyed stability. The roof contains the primary goals of TPS: superior quality, cost and delivery through waste elimination.

The Toyota Way



From The Toyota Way, by Jeffrey K. Liker

Combine to create an effective problem solving organization continuously improving to outperform the competition

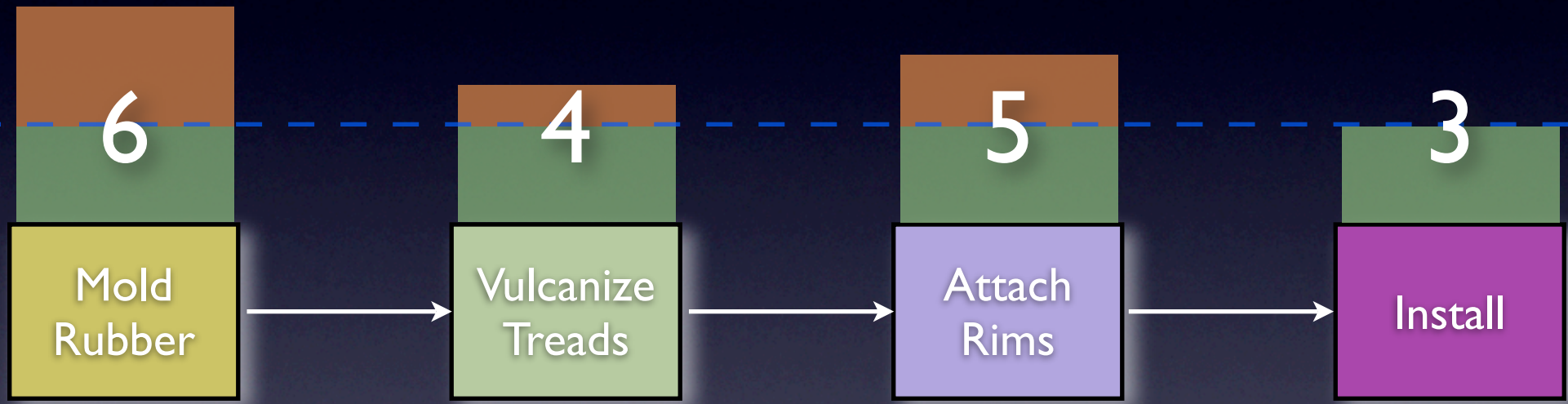
Types of Waste

Manufacturing	Software Development
In-Process Inventory	Partially Done Work
Over-Production	Extra Features
Extra Processing	Relearning
Transportation	Handoffs
Motion	Task Switching
Waiting	Delays
Defects	Defects

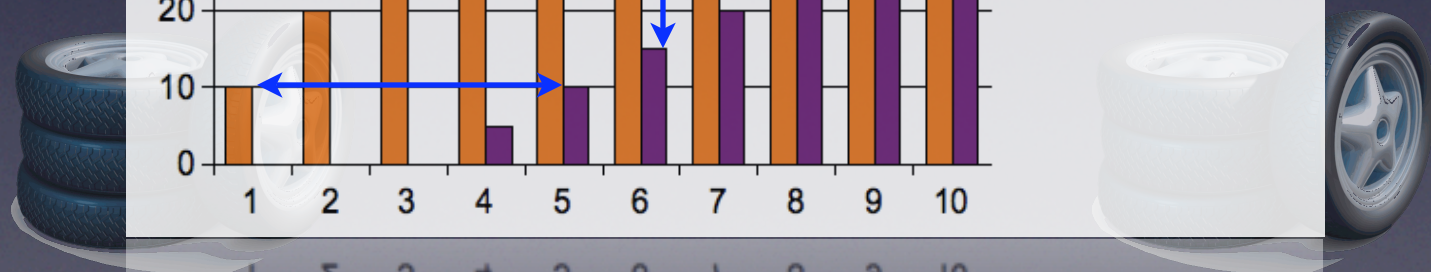
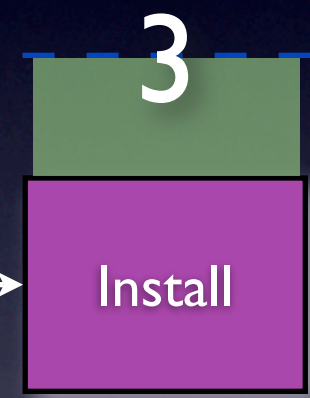
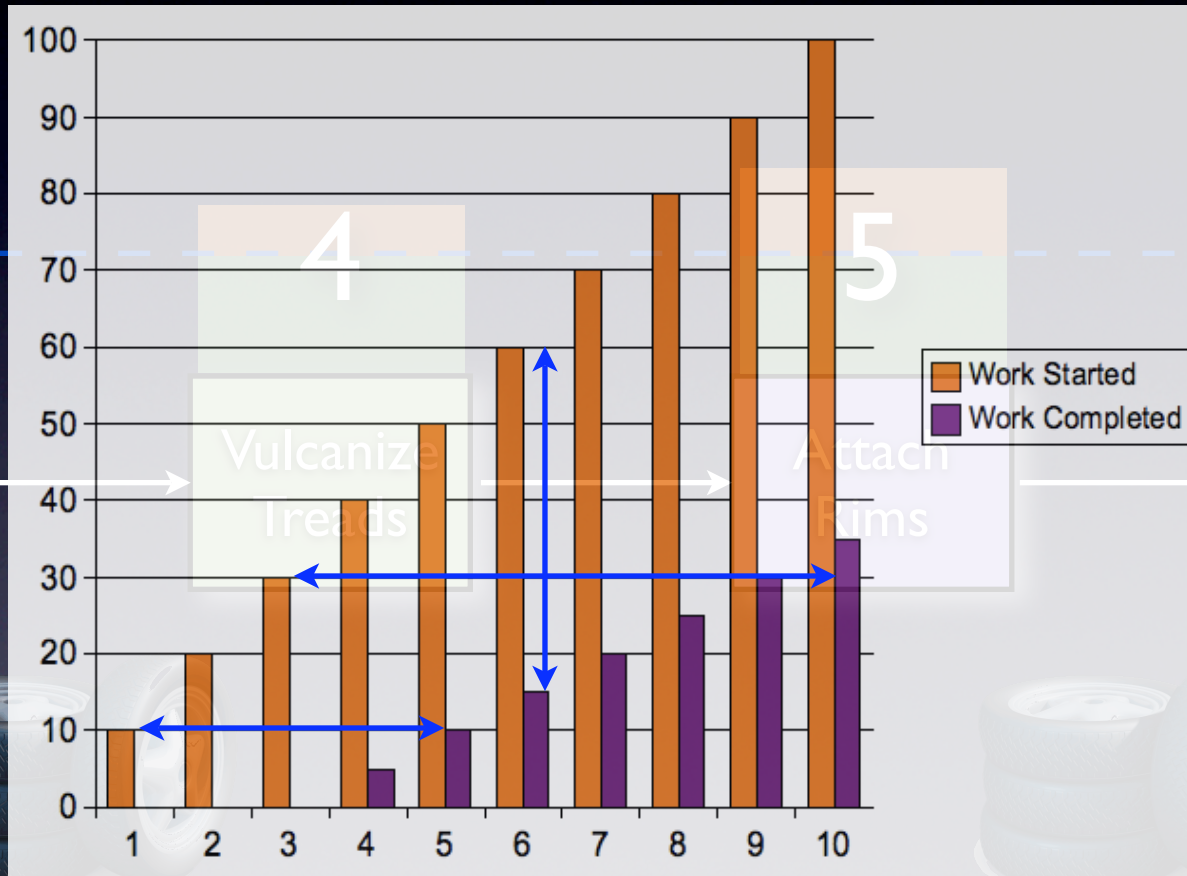
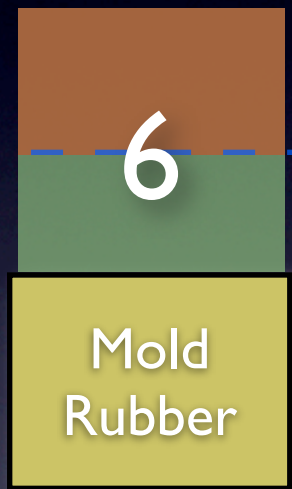
From Implementing Lean Software Development by Mary and Tom Poppendieck

Wastes highlighted in red are most directly addressed by pull

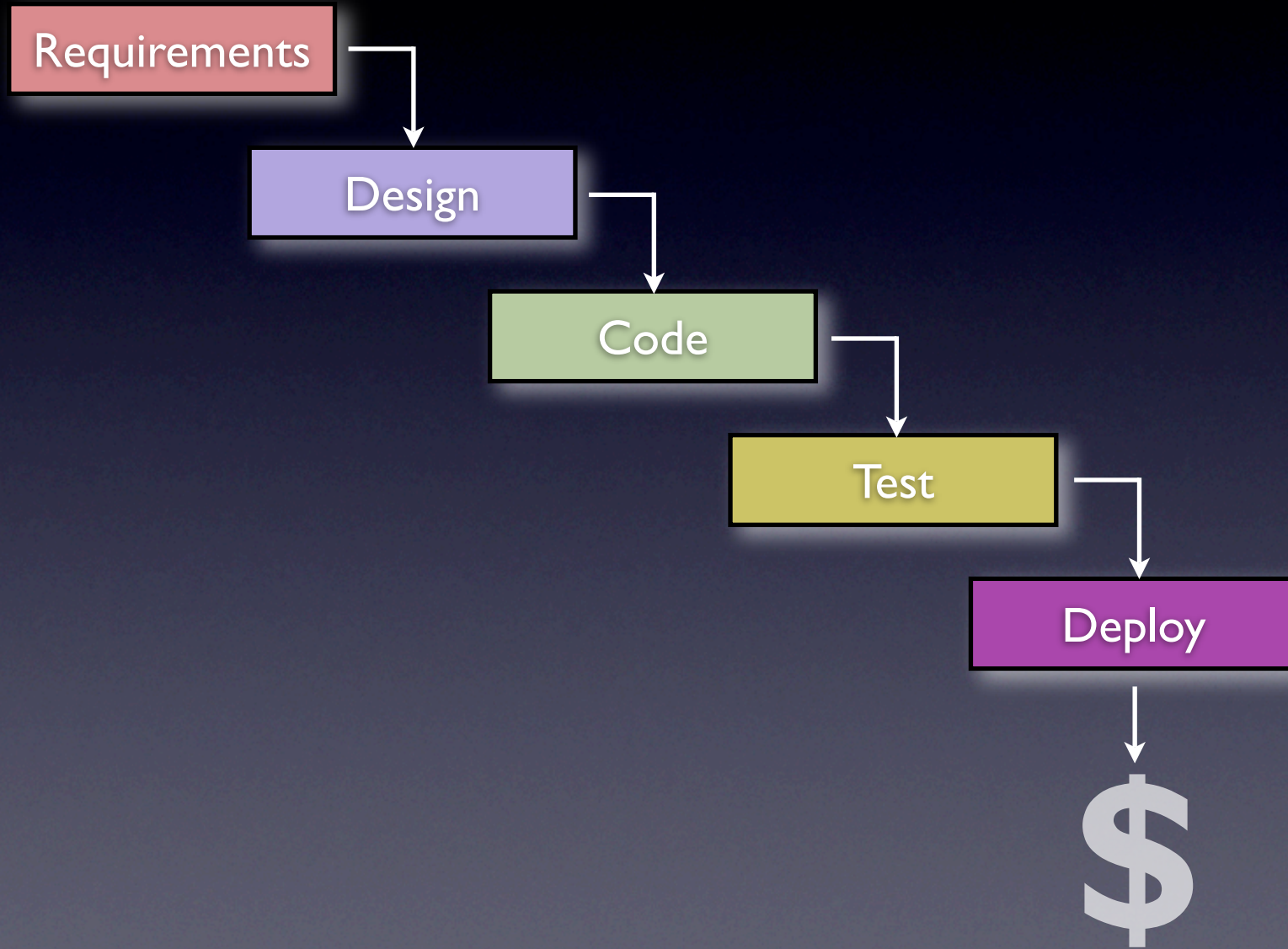
Push Model Manufacturing



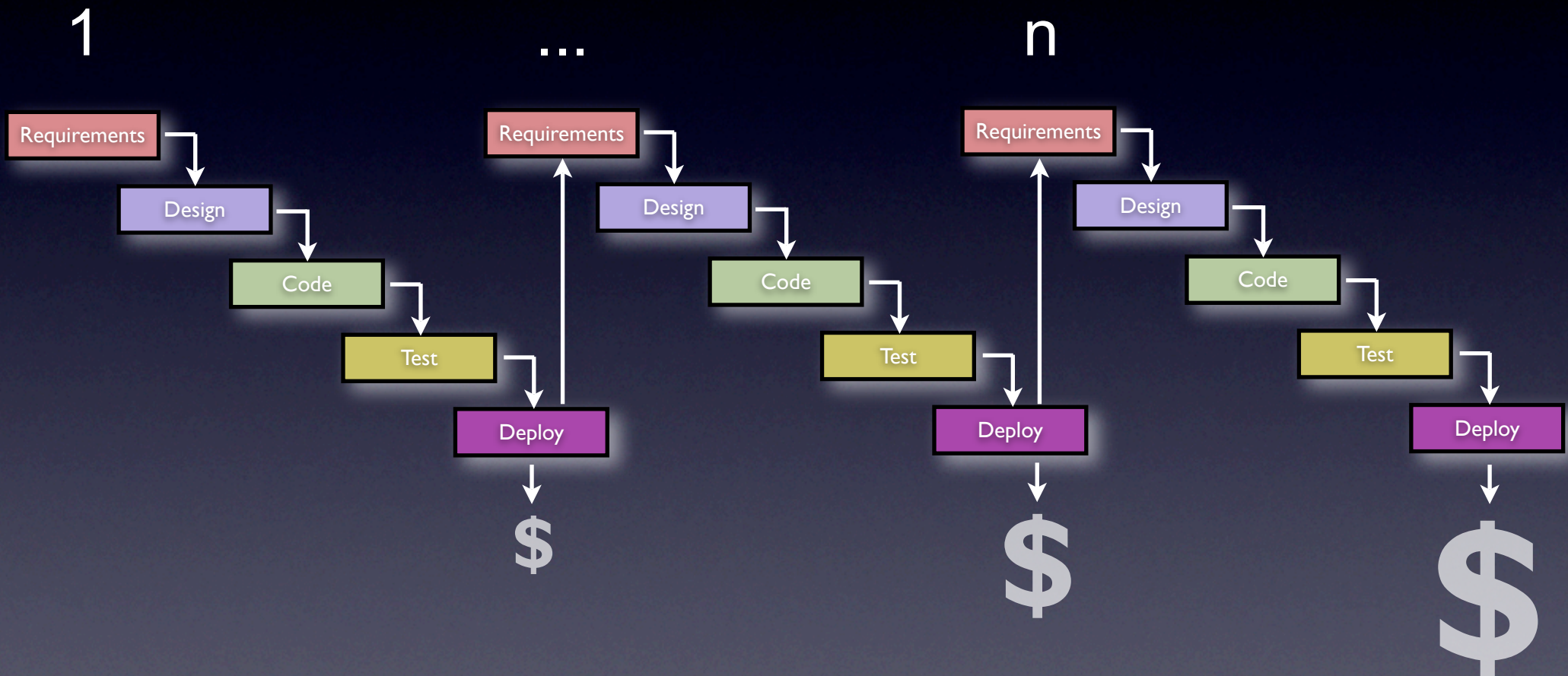
Push Model Manufacturing



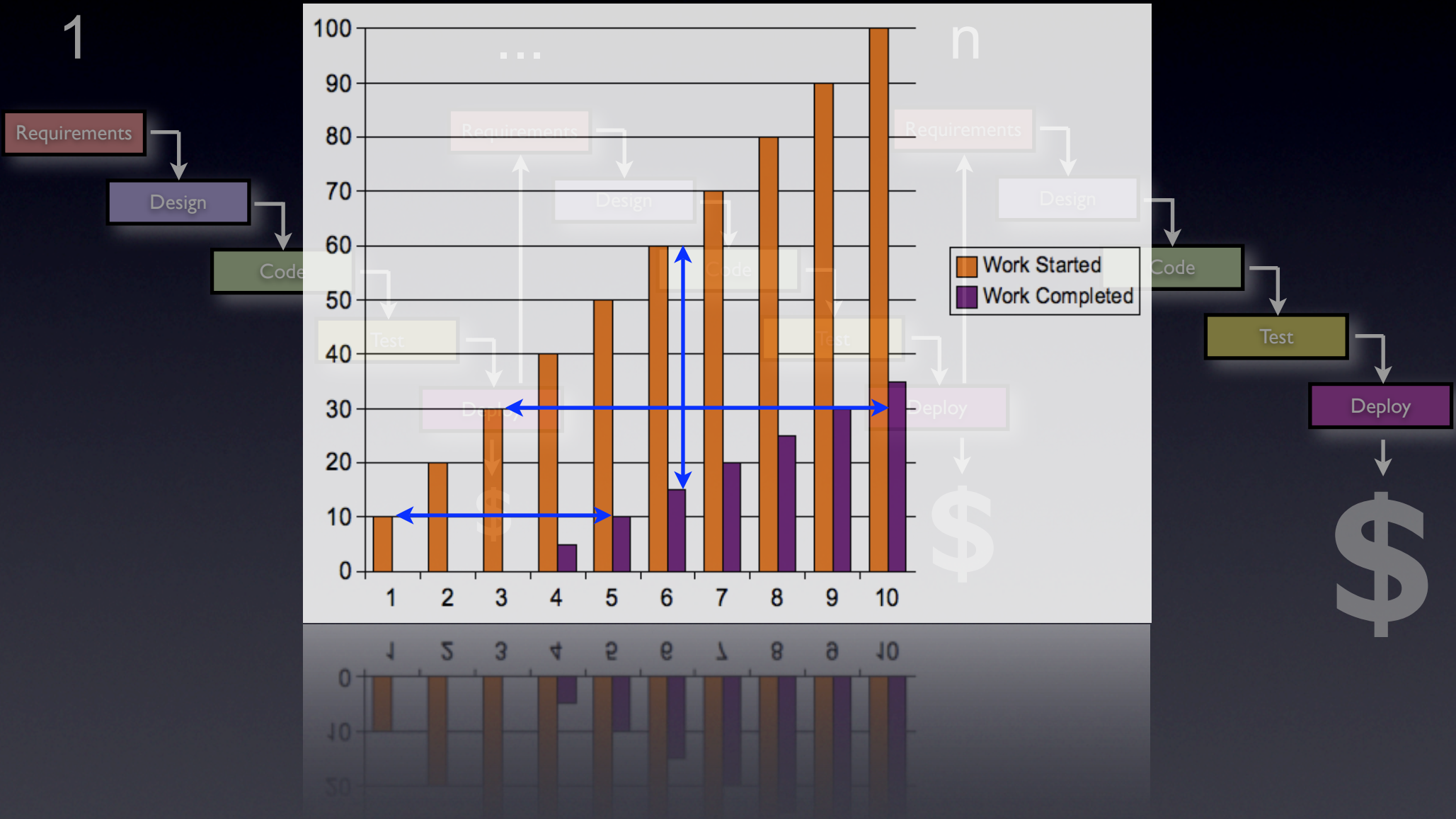
Waterfall



Iterative

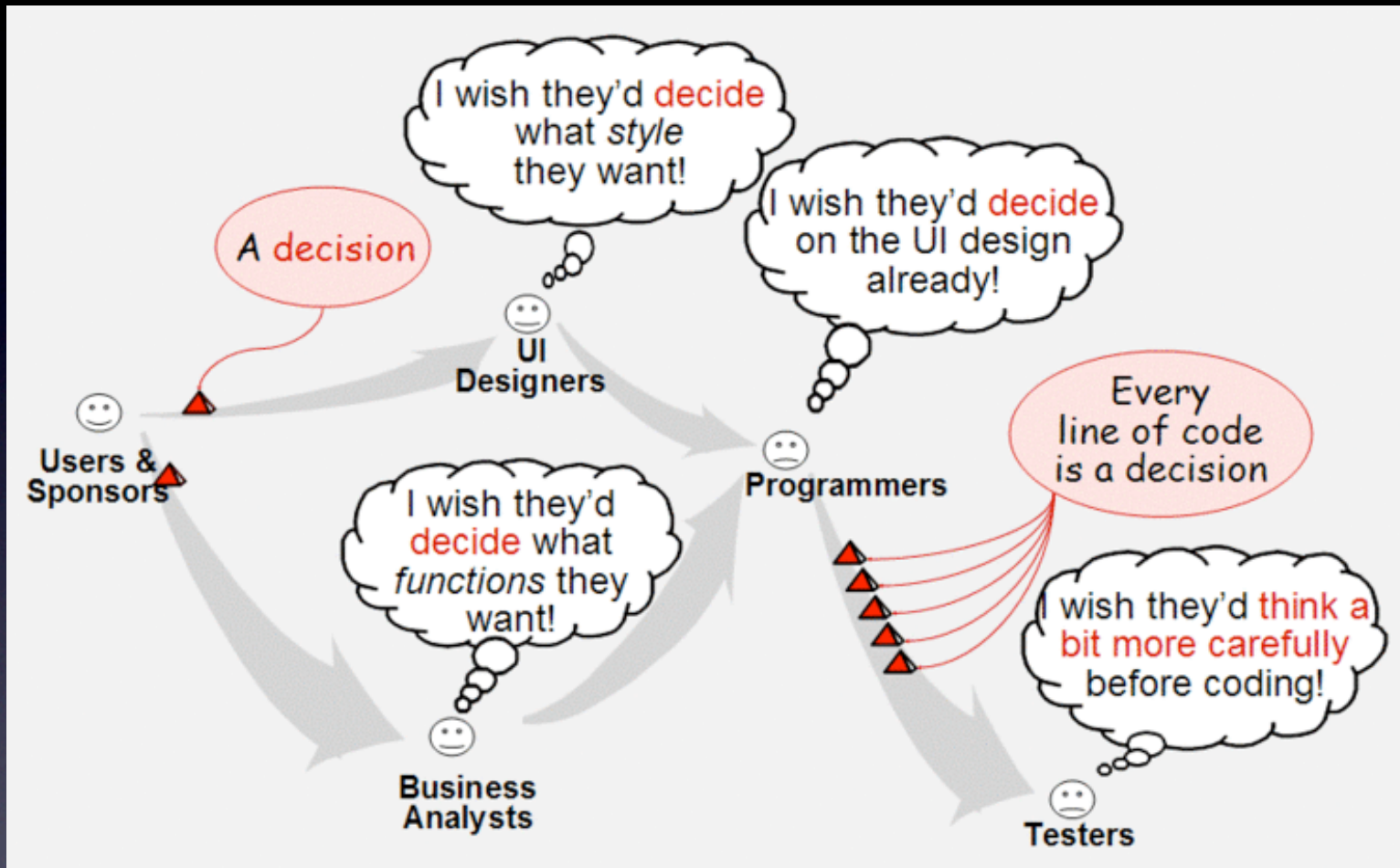


Iterative



Deferral of initial revenue production
Increasing work in progress
Increasing cycle times

Lean Software



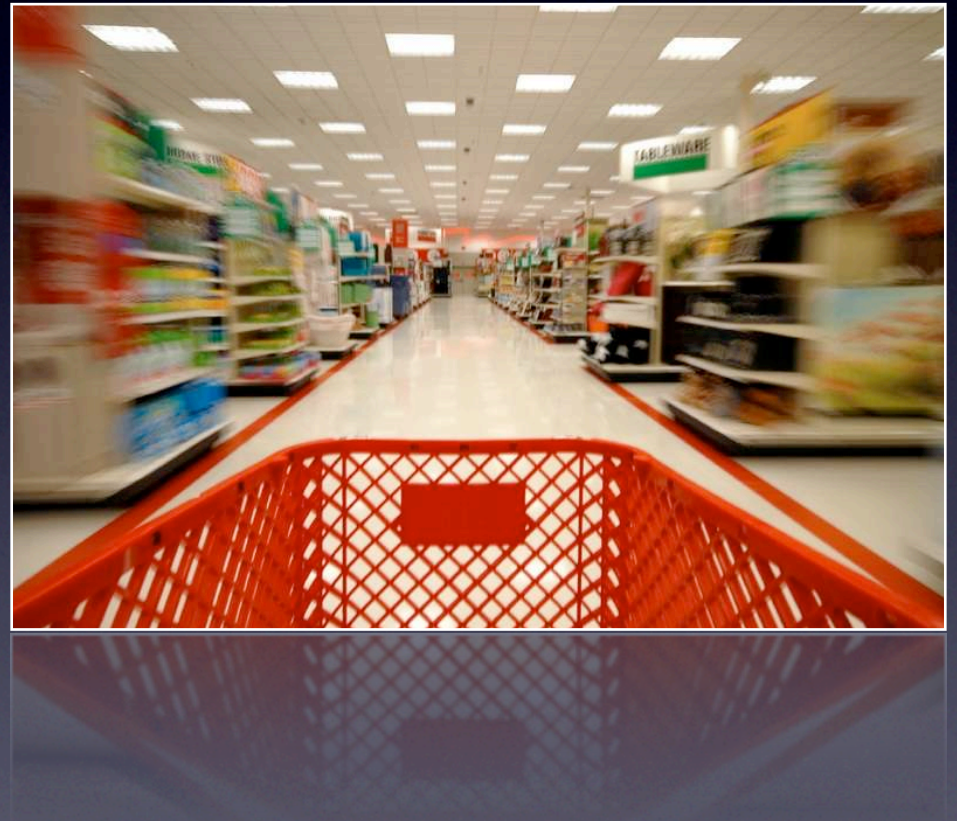
<http://alistair.cockburn.us/index.php/>

What_engineering_has_in_common_with_manufacturing_and_why_it_matters

Pull

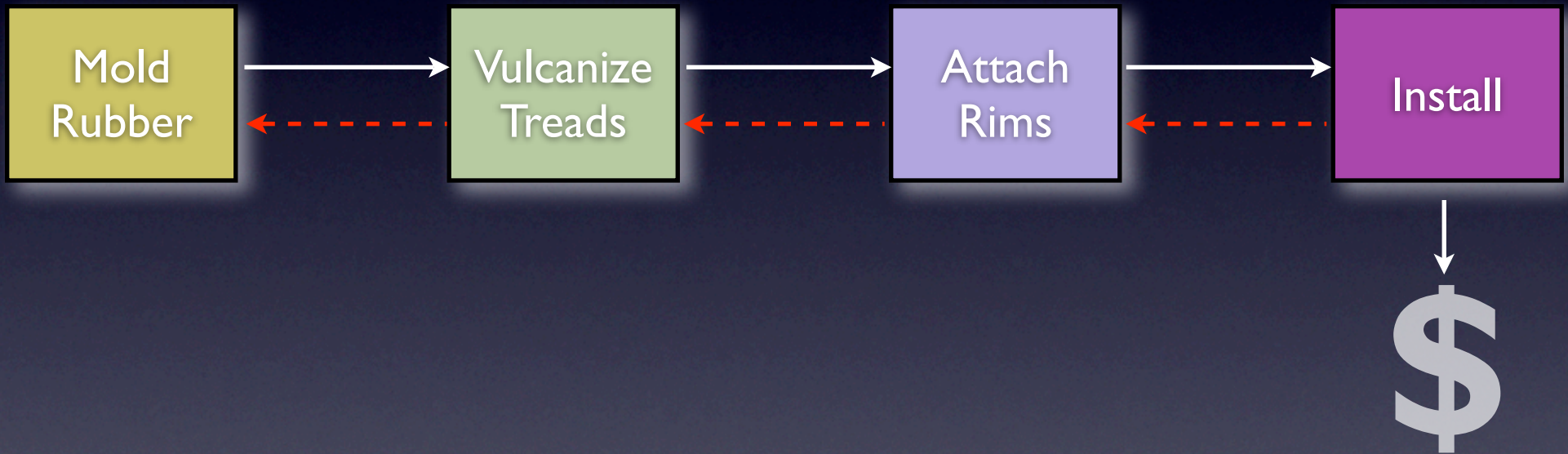
Invert the process to:

- Build just enough
- Build the right thing
- Build it at the best time

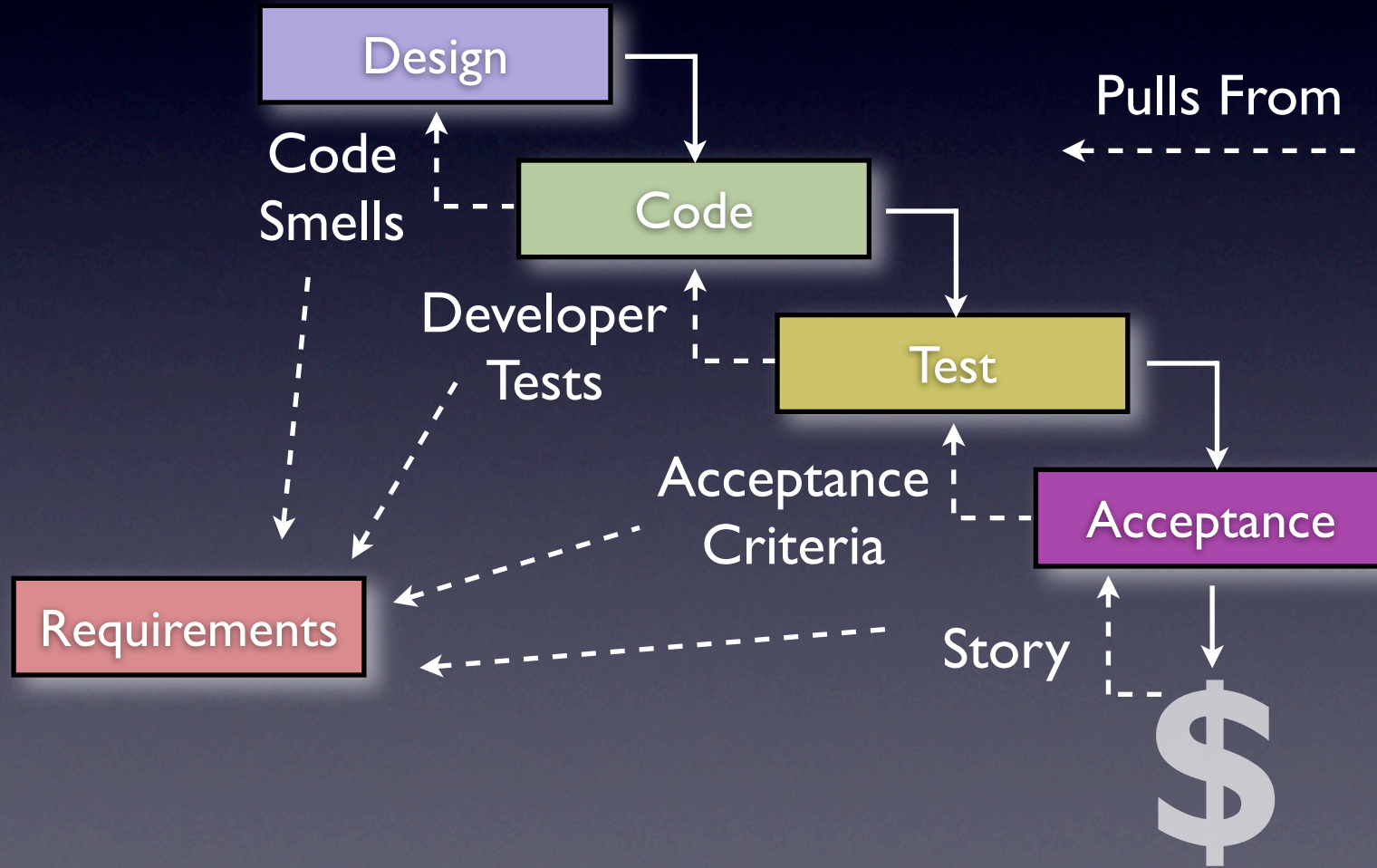


Supermarkets work by pulling requisite parts from upstream processes
How to coordinate the interactions of stations with their upstream counterparts?
Kanban cards are an example mechanism for creating negative pressure

Pull Model Manufacturing



Pull Model



STORY

CRITERIA

DEV SPEC

DESIGN

To frame the presentation , i'd like to stick to the 4 steps that we talked about.
First, we can go over each and talk about them. Then we go over each and do the demo.

It all starts with a promise...



S
T
O
R
Y

A story is a promise for a future conversation...

As a **<user>**, I want to
<do something> so that I
can **<accomplish some
goal>**.

STORY

CRITERIA

DEV SPEC

DESIGN

Given **<a condition>**
When **<event occurs>**
Then the **<system
should ...>**

C
R
I
T
E
R
I
A

flip the story card to reveal the acceptance criteria

Multiplication

First Number	Second Number	Product?
10	4	40
12.3	5	61.5

C R I T E R I A

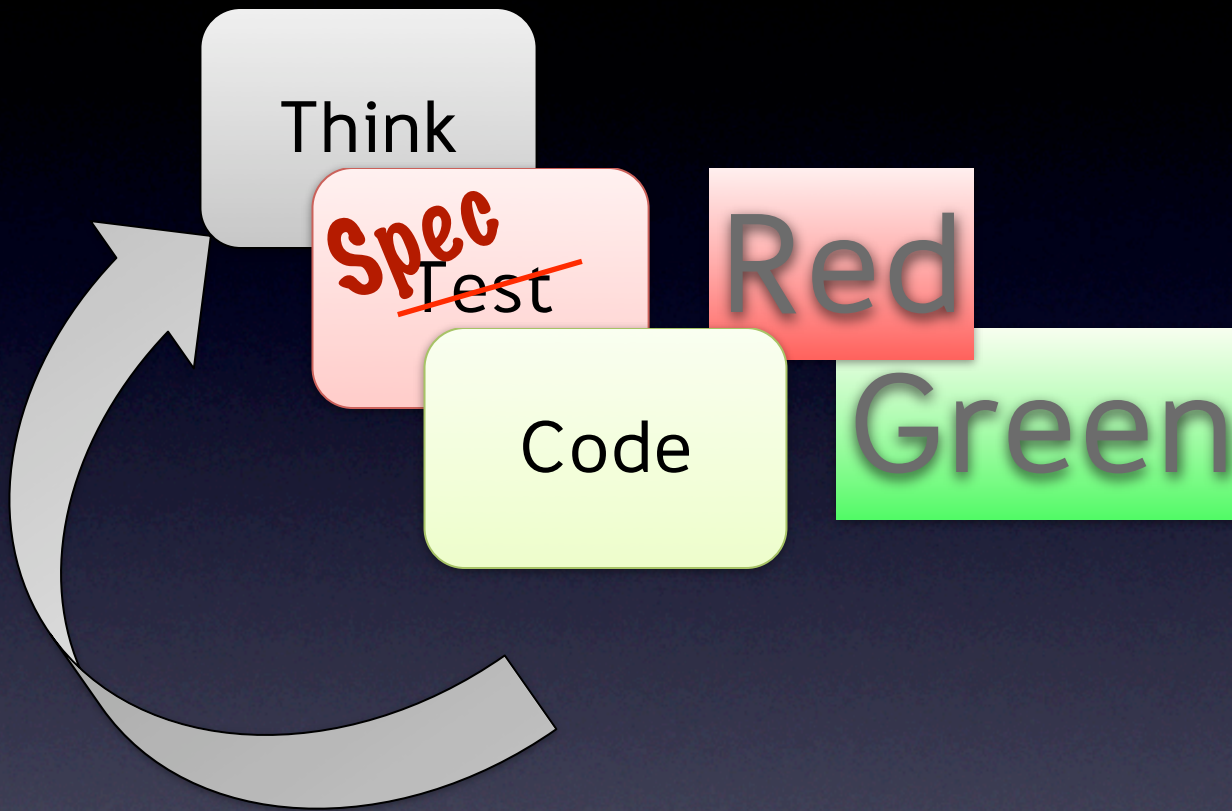
Show a table (as in executable requirements) and then tell them about it.

P R O M I S E

C R I T E R I A

D E V S P E C

D E S I G N



DEV SPEC

Let's talk about coding in the context of TDD/BDD. Do the simplest thing that works, then refactor.

Step 0: Think
(take as long as you need*)

* Jim Shore

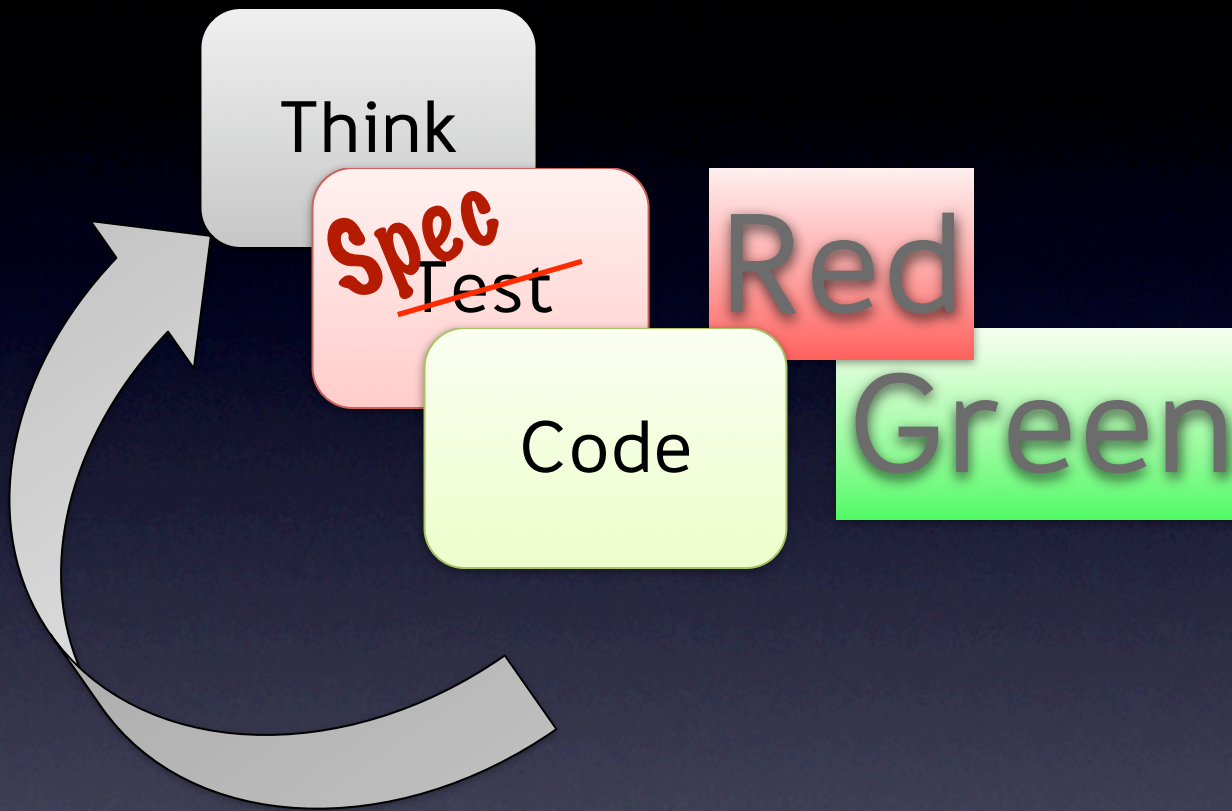
Step Zero, which Jim Shore writes about in his blog, is to think. Think about what the code should look like, what behavior it should exhibit, and how you would like to program against it. As Jim says, this is the hardest step. Take as long as you want. After a while, it will become more natural.

Step 1: Write a spec.

Step 1 is to write a test that expresses your thoughts about how the interfaces should look, about how the software should behave. By virtue of the fact that you are making choices about Class and Method names, dependencies, and so on, you are actually designing the software at this point. This is a key point that many people like me missed at first glance.

Step 2: Make the spec pass. (Just barely)

Step 2 is to make the test pass. This usually turns out to be far easier than writing the test, especially if you do the simplest thing that could possibly work.



DEV SPEC

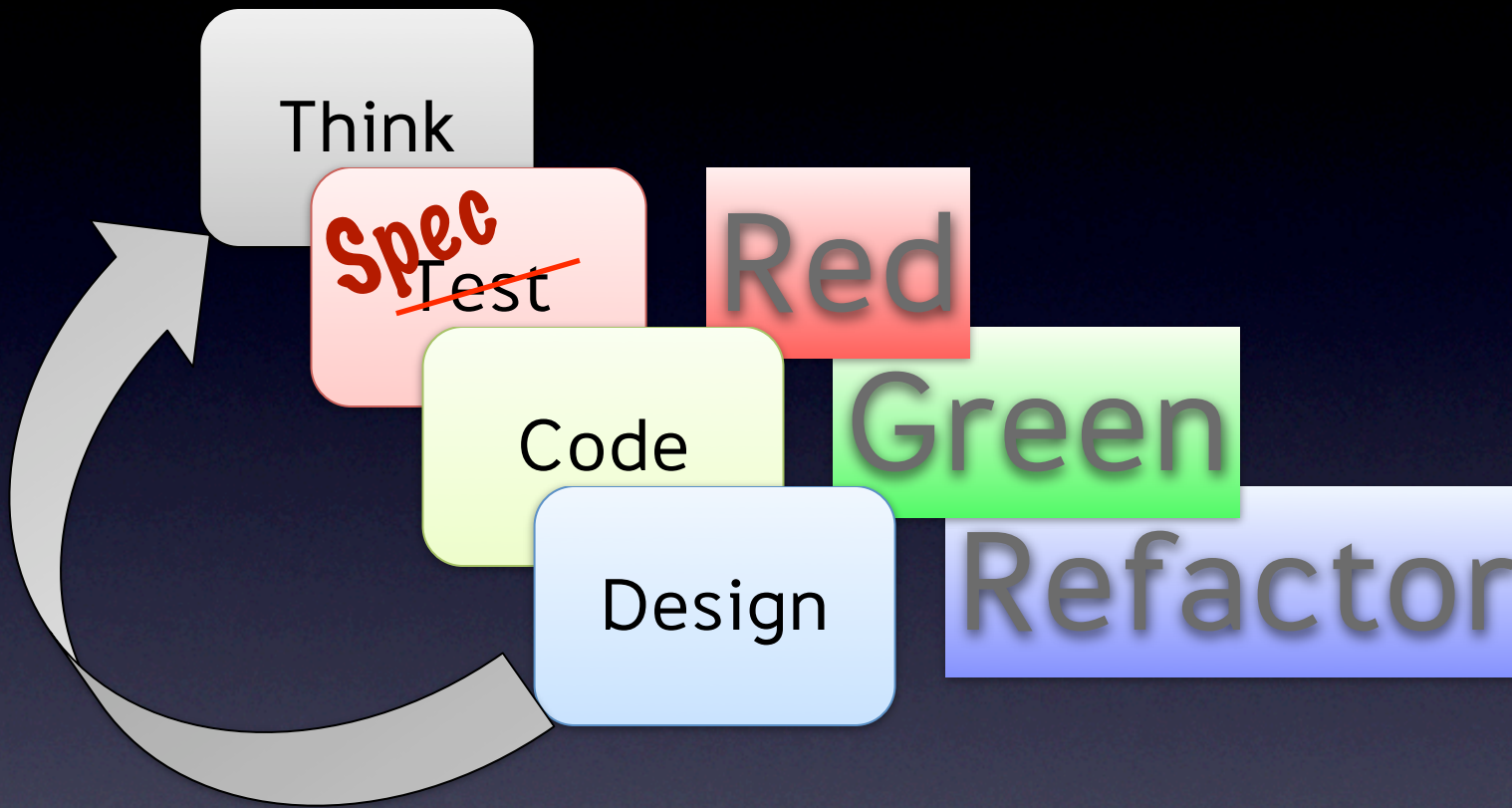
Let's talk about coding in the context of TDD/BDD. Do the simplest thing that works, then refactor.

P R O M I S E

C R I T E R I A

D E V S P E C

D E S I G N

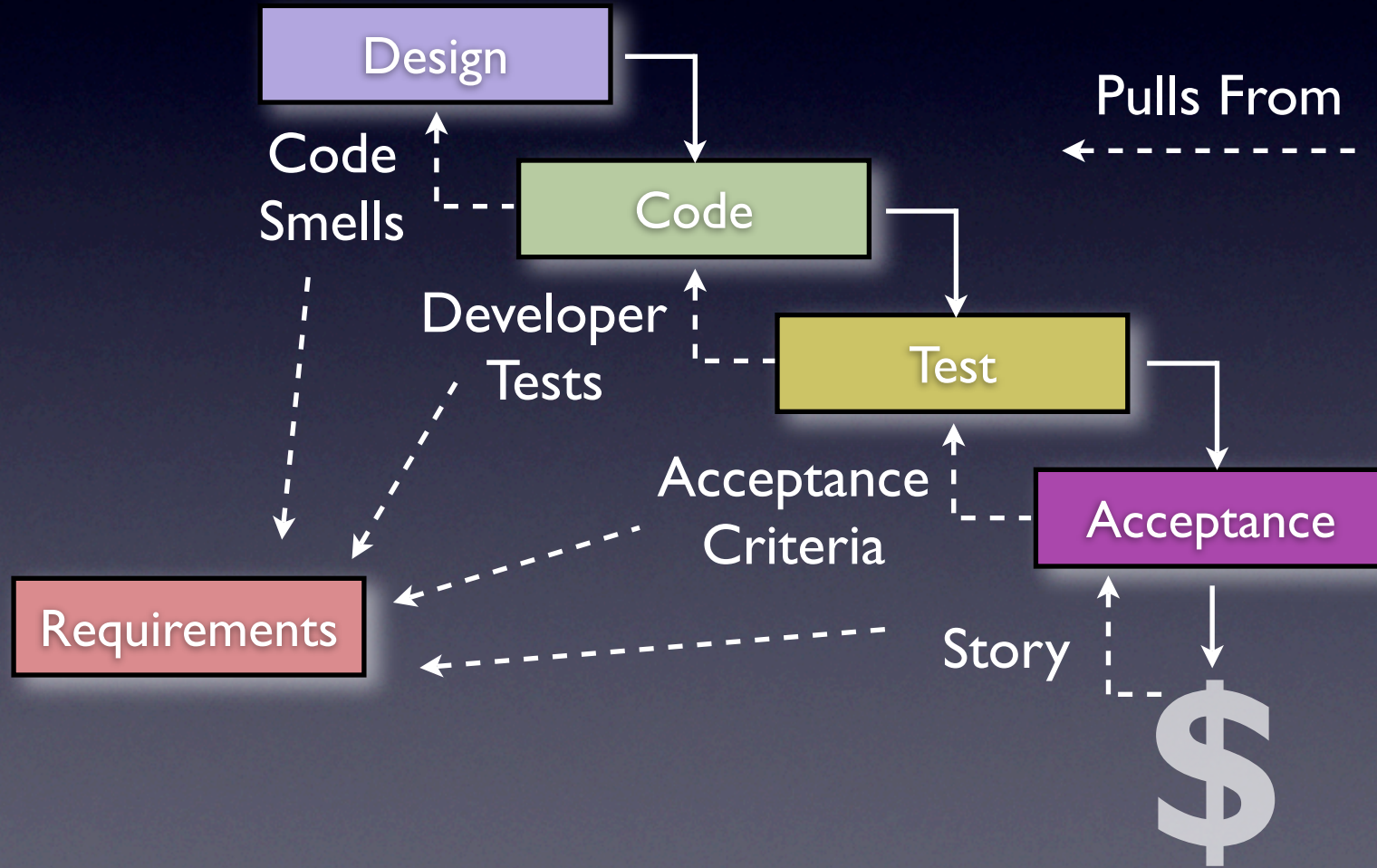


DESIGN

The final step is to re-factor. This is *really where you design your system to make it easier to understand and more maintainable. You aren't altering behavior here - you're tests should all still pass. You're just trying to simplify things. Because you have a Green suite of tests to back you up, you can make your changes with the confidence that you won't break other parts of the system. When I was a kid, my dad taught me that it was important to clean up as you went while working in our shop. This is kind of the same thing.

Demo

Pull Model



Questions?

Thank You!

Rod.Coffin@ImprovingEnterprises.com

Don.McGreal@ImprovingEnterprises.com

These are just some stories that i put together for our demo idea.