Lean IT Summit

“Training Within Software” using Dojo and Mob Programming

Bernard Notarianni

Paris, France
16 & 17 October, 2014

Bernard Notarianni

Agile Coach / XP
http://umolelo.com

Software creation

- Web/Mobile
- Microsoft .Net
- Erlang

Agile Coaching

- Managers, product owners, devs
- XP practices trainer
- Team dynamics facilitation

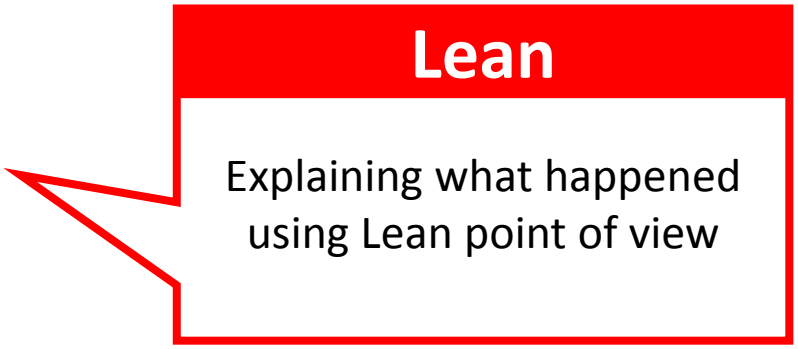
Story

Those people found what works for them

We are sharing with those who want to try

We do not try to convince anybody

Story



Those people found what works for them

We are sharing with those who want to try

We do not try to convince anybody

The team

4 functional experts

- Within 10 to 20 years of experience
- Distributed in France, Belgium, Italy and Spain

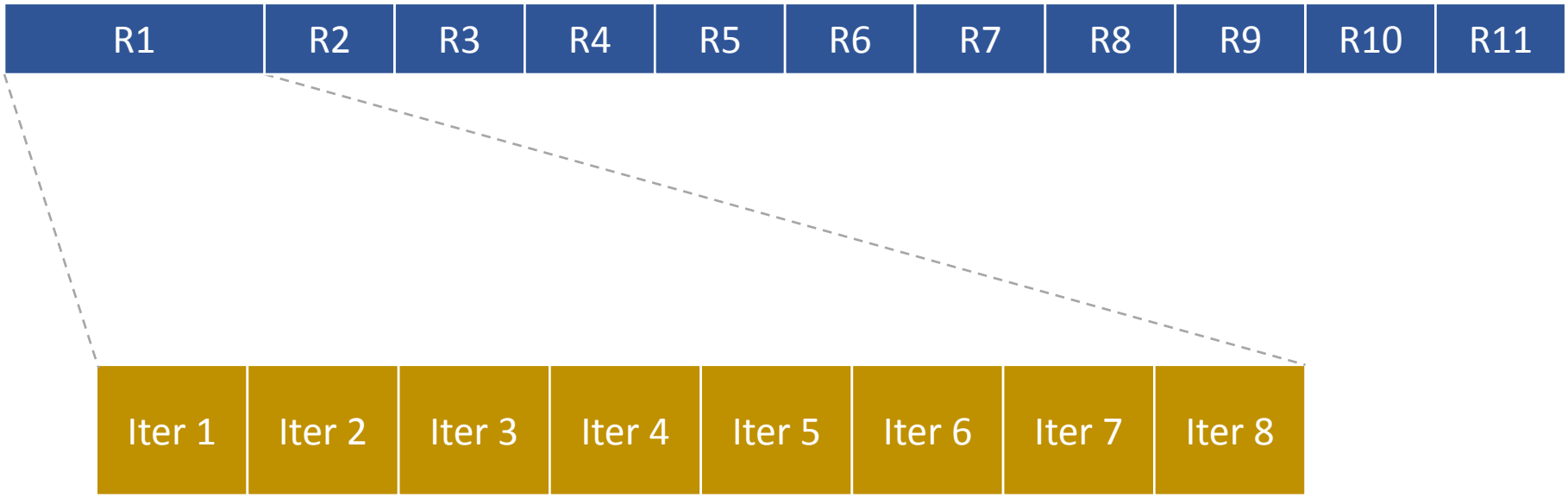
5 developers colocated in Paris

- 1 expert .Net /MVC – technical lead
- 1 senior dev on .Net/MVC
- 3 internal developers

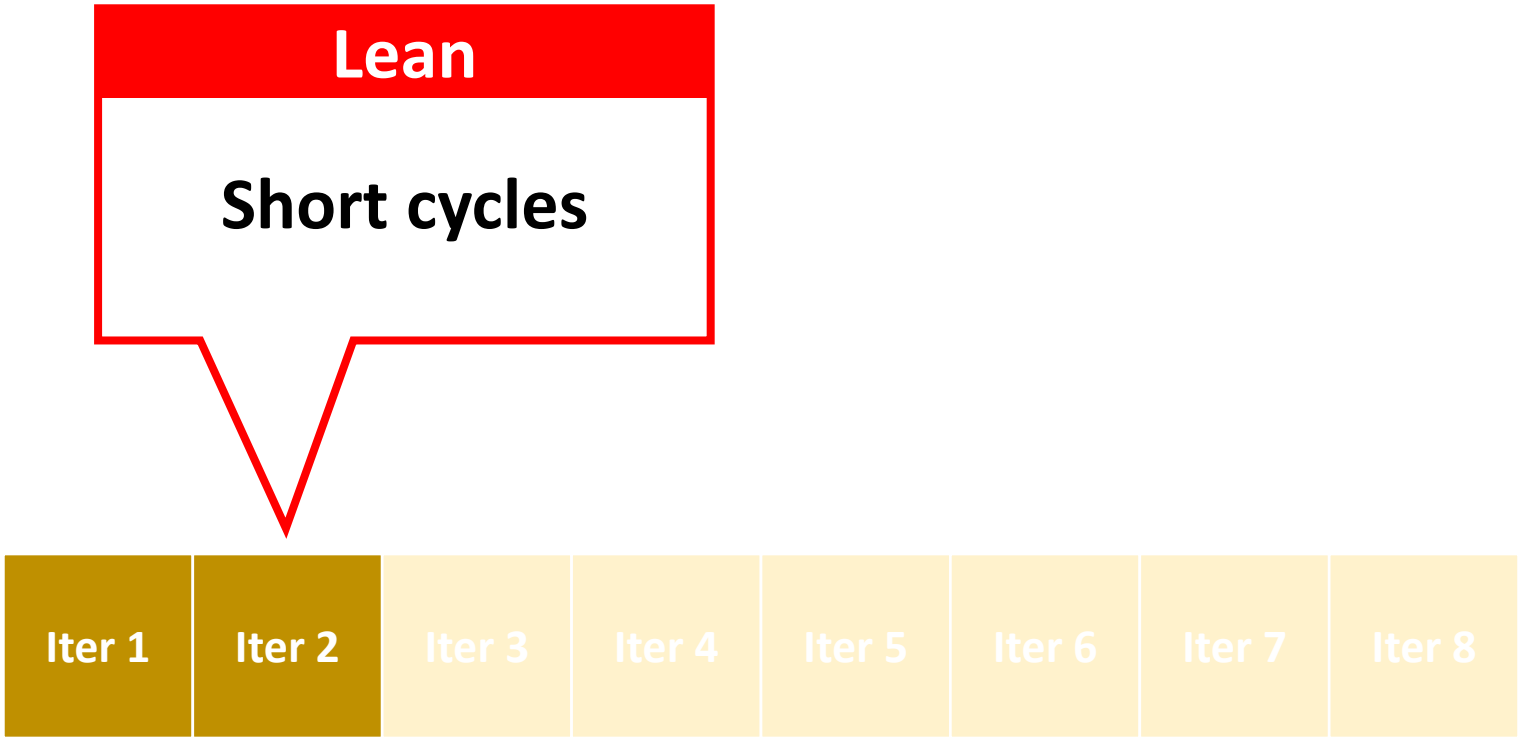
1 agile coach (XP / Craftmanship)

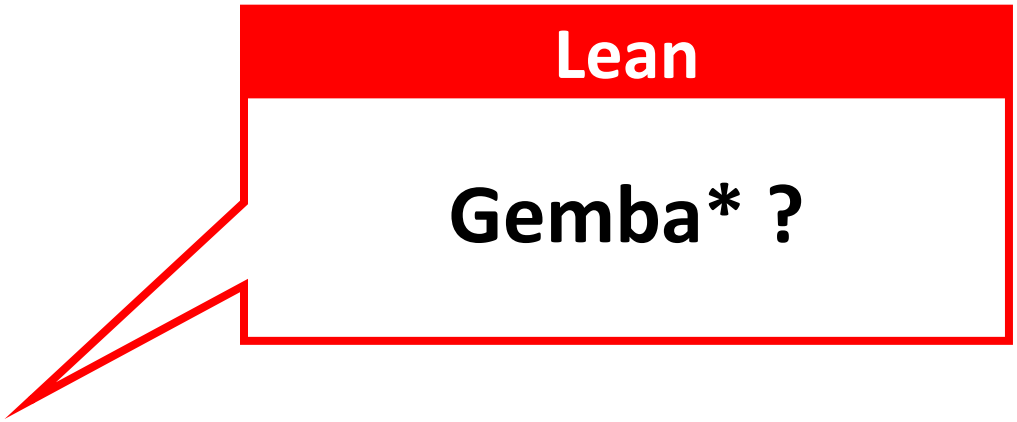
One year project

About 4000 to 8000 Man.Days

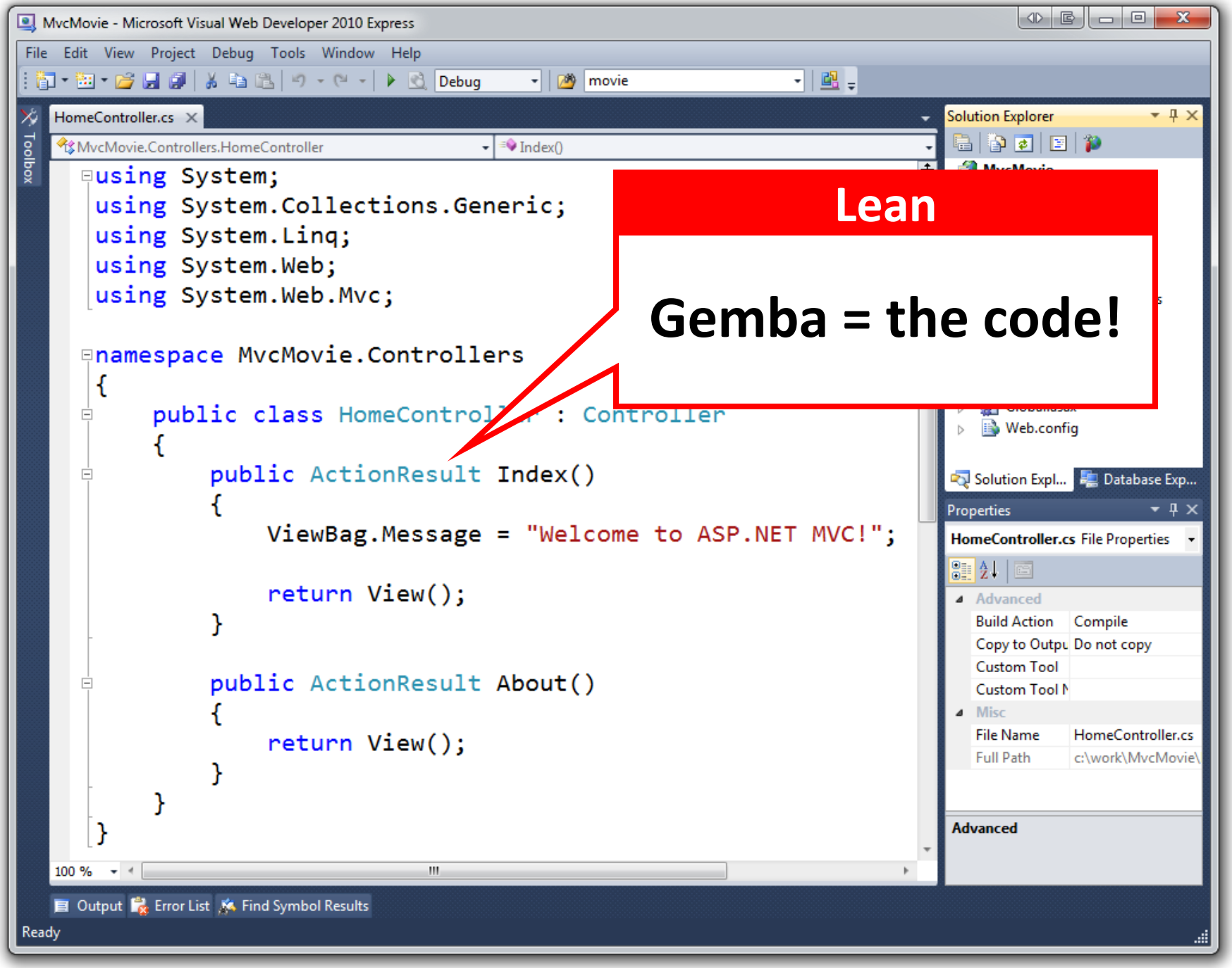


Release 1 = 8 one-week iterations


Lean
Short cycles
Iter 1
Iter 2
Iter 3
Iter 4
Iter 5
Iter 6
Iter 7
Iter 8



*Gemba = the « real place » where the work is done and the product is created.


MvcMovie - Microsoft Visual Web Developer 2010 Express
File Edit View Project Debug Tools Window Help
Debug
movie
HomeController.cs
MvcMovie.Controllers.HomeController
Index()
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace MvcMovie.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
ViewBag.Message = "Welcome to ASP.NET MVC!";
return View();
}
public ActionResult About()
{
return View();
}
}
}
Lean
Gemba = the code!
Solution Explorer
Web.config
Solution Expl...
Database Exp...
Properties
HomeController.cs File Properties
Advanced
Build Action
Compile
Copy to Outpu
Do not copy
Custom Tool
Misc
File Name
HomeController.cs
Full Path
c:\work\MvcMovie\
Advanced
100 %
Output
Error List
Find Symbol Results
Ready

Observation	Hypothesis	Experiment	Result
Cycle 1			
Bad quality of code	We dont know the technology	Training with cycle of Dojo	

Training program

Lean

Topics are fully customised to the context of the team

1.	CRUD Asp.Net MVC & Code Fluent Entities	**Tech**
2.	Partial View, Child Action with Asp.Net MVC	**Tech**
3.	TDD	**Method**
4.	Unit Test & Mock	**Tech**
5.	FitNesse	**Tech**
6.	Working methodology	**Method**
7.	MVC ++ (Ajax etc.)	**Tech**
8.	TFS + Azure	**Method**
9.	Code Fluent Entities ++	**Tech**
10.	Dependency injection with Unity	**Tech**
11.	Watin	**Tech**

1. Création de la couche Business + Base de données

CodeFluent Entities est un outil intégré dans Visual Studio permettant notamment aux développeurs de créer le modèle de base de données d'une application, et de générer en continu la base de données. Une mise à jour du modèle met automatiquement à jour la base de données sans perte de données.

Dans la surface « Default » :

- Création d'une entité « Client » (dans le Namespace)
  - Création d'une propriété Id de type Guid
  - Création d'une propriété LegalName de type string
  - Création d'une propriété de type string
- Création d'une entité « Contact » (dans le Namespace)
  - Création d'une propriété Id de type Guid
  - Création d'une propriété FirstName de type string
  - Création d'une propriété LastName de type string
- Création de la relation entre Client et Contact : un client a zé lusieurs contact, un contact à un seul client.
  - Création d'une propriété Client dans l'entité C
  - Création d'une propriété Contacts dans l'entit
  - Création d'une relation de la propriété Client
    - Garder la touche SHIFT enfoncée
    - Cliquer gauche sur la propriété Client jusqu'à la propriété Contact (une flèch

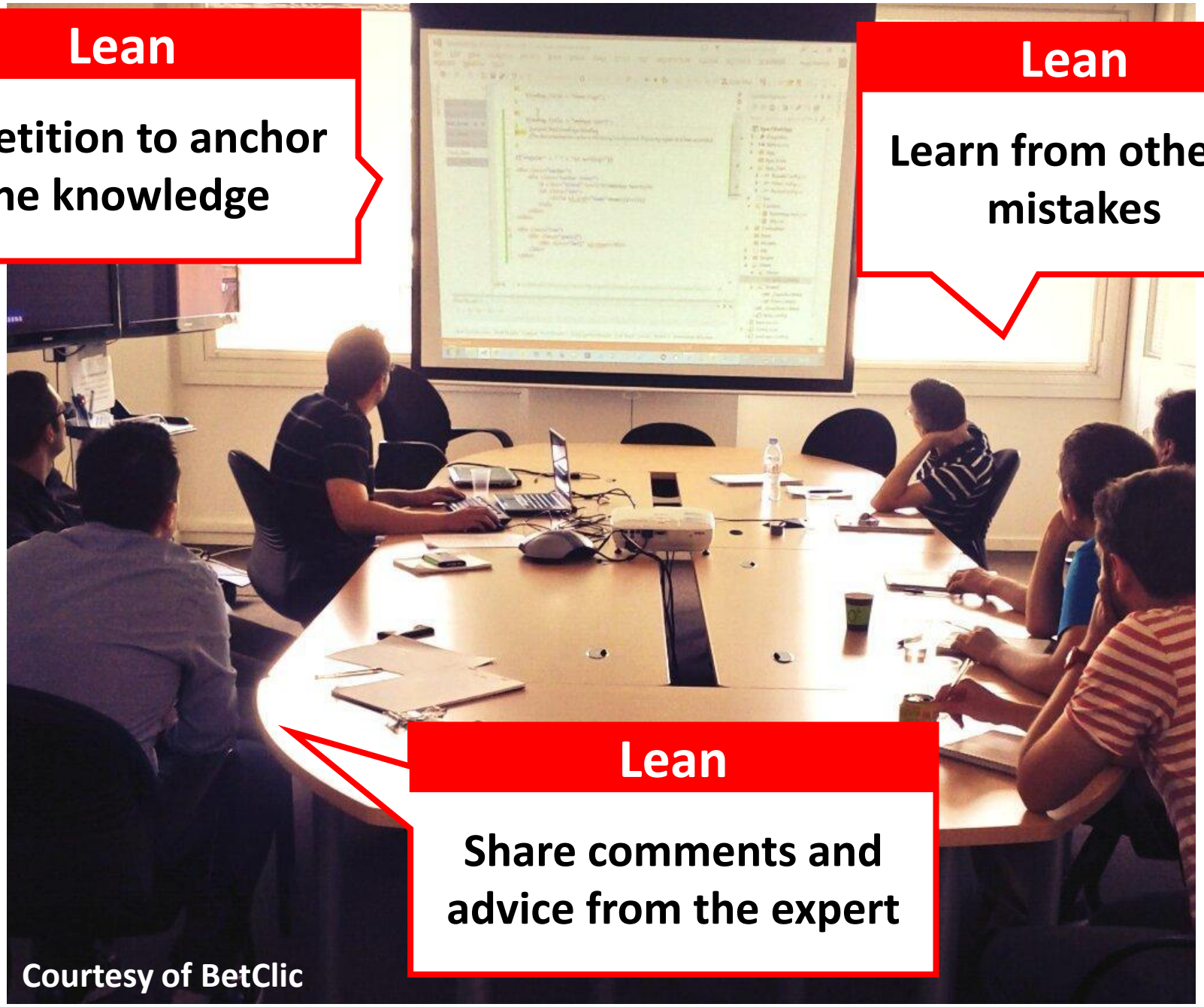

Lean
Repetition to anchor the knowledge
Lean
Learn from others' mistakes
Lean
Share comments and advice from the expert
Courtesy of BetClic


Photo Zach Dischner - CC BY 2.0

Observation	Hypothesis	Experiment	Result
Cycle 1			
Bad quality of code	We dont know the technology	Training with cycle of Dojo	Code compliant with .net/MVC standards


Iter 1
Iter 2
Iter 3
Iter 4
Iter 5
Iter 6
Iter 7
Iter 8

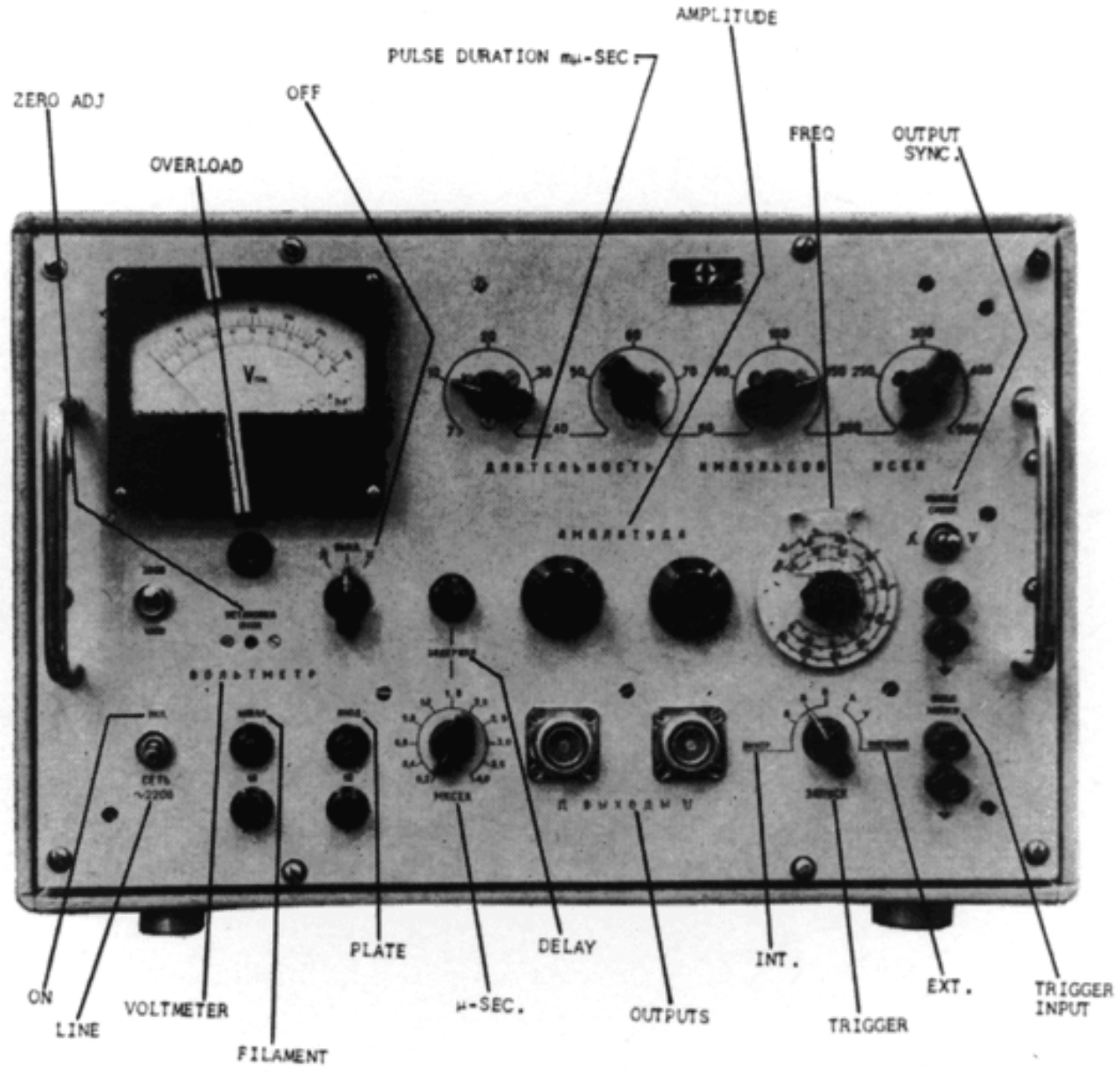
x 2

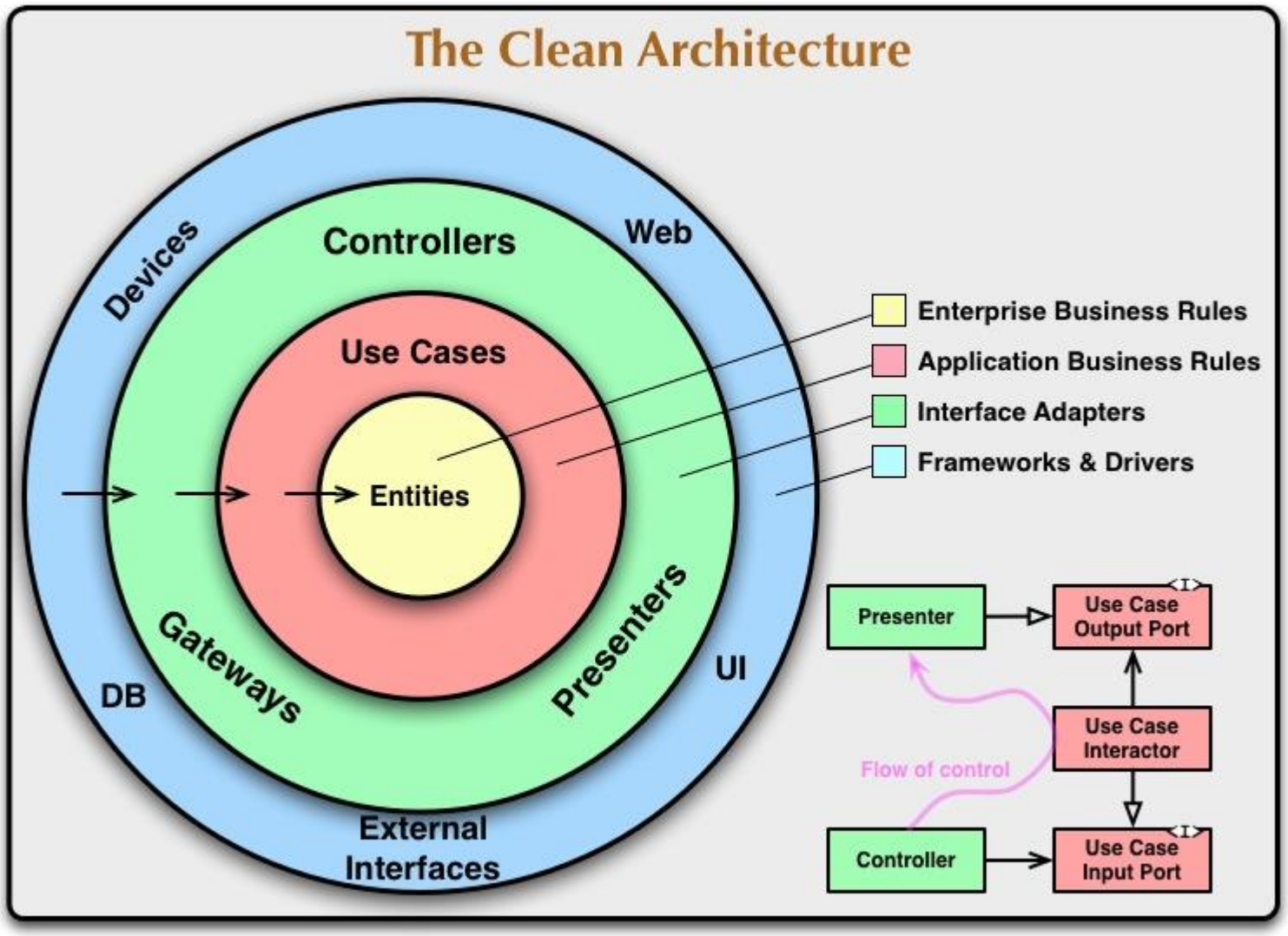
Lean

The voice of the client

x 2

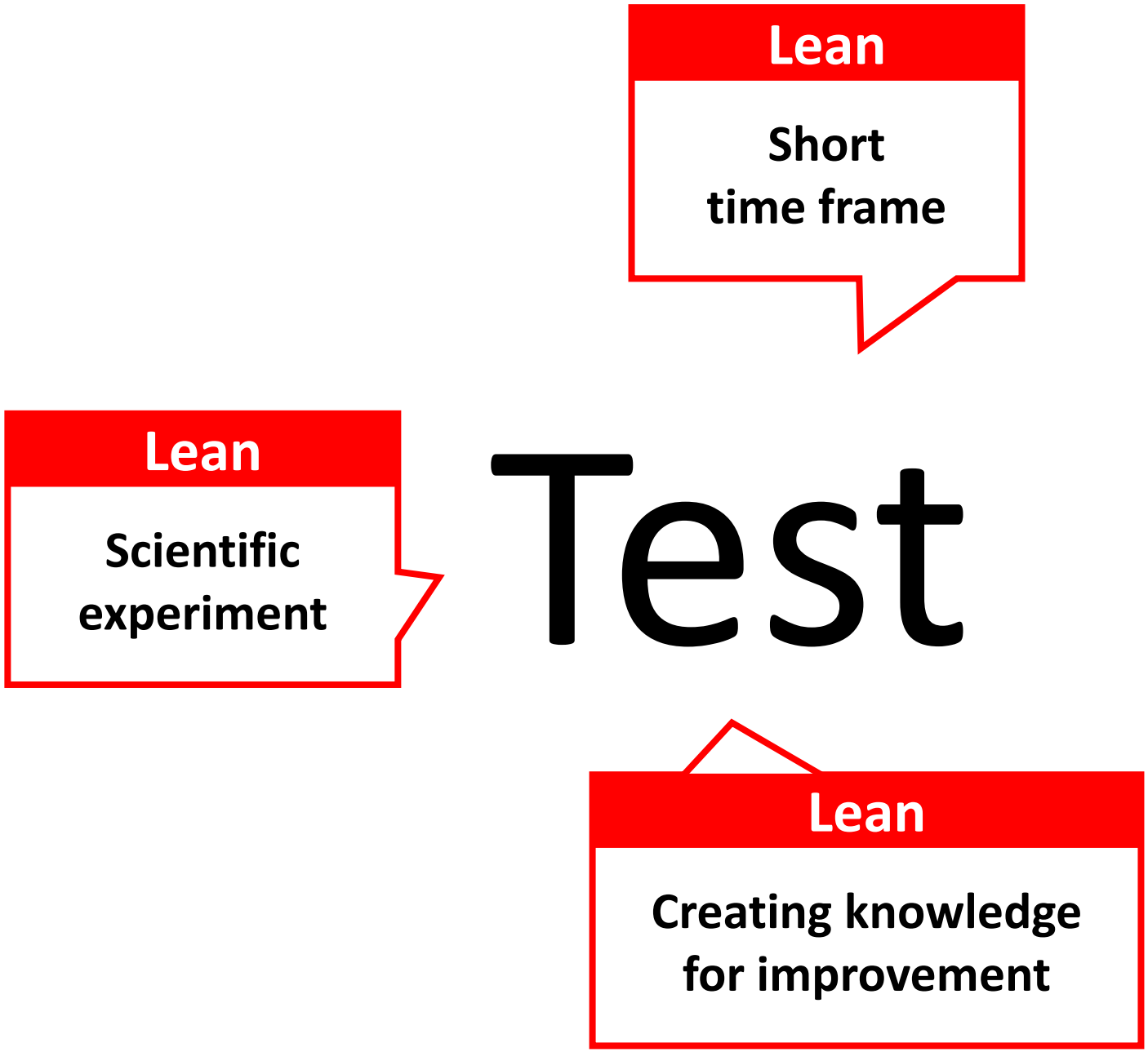
Observation	Hypothesis	Experiment	Result
Cycle 1			
Bad quality of code	We dont know the technology	Training with cycle of Dojo	Code compliant with .net/MVC standards
Cycle 2			
Low velocity	UI complexity is slowing us down		


AMPLITUDE
PULSE DURATION mμ-SEC.
ZERO ADJ
OFF
FREQ
OUTPUT SYNC.
OVERLOAD
ON
LINE
VOLTMETER
FILAMENT
PLATE
DELAY
μ-SEC.
OUTPUTS
INT.
TRIGGER
EXT.
TRIGGER INPUT



From 8th light blog, Robert Martin (Uncle Bob)

Test

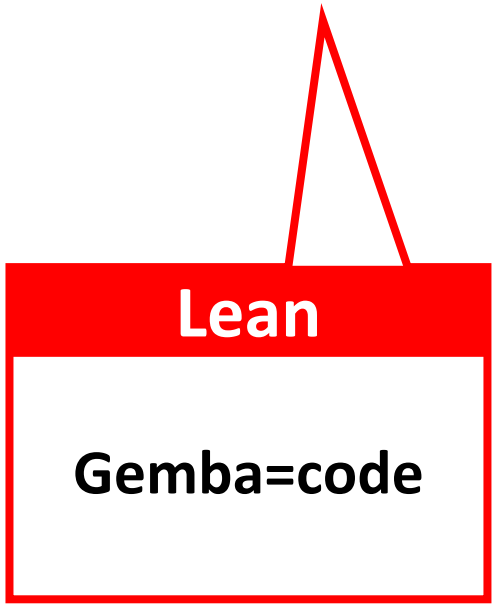

Lean
Short
time frame
Lean
Scientific
experiment
Test
Lean
Creating knowledge
for improvement

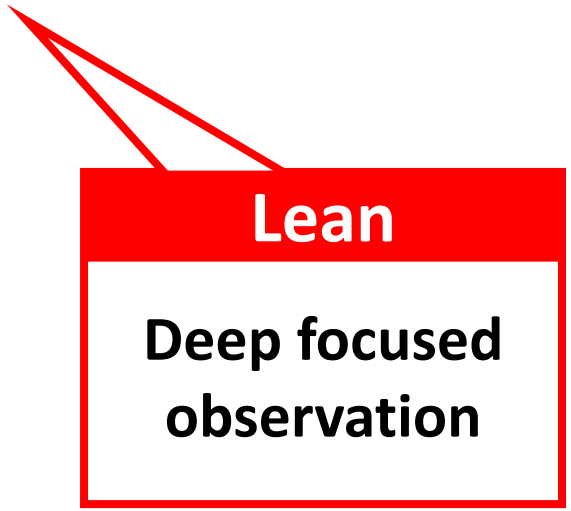
Observation	Hypothesis	Experiment	Result
Cycle 1			
Bad quality of code	We dont know the technology	Training with cycle of Dojo	Code compliant with .net/MVC standards
Cycle 2			
Low velocity	UI complexity is slowing us down	Implement all but UI in one day	

Black Friday

I was wrong

Observation	Hypothesis	Experiment	Result
Cycle 1			
Bad quality of code	We dont know the technology	Training with cycle of Dojo	Code compliant with .net/MVC standards
Cycle 2			
Low velocity	UI complexity is slowing us down	Implement all but UI in one day	FAIL: everything is slowing us down.





Observation	Hypothesis	Experiment	Result
Cycle 1			
Bad quality of code	We dont know the technology	Training with cycle of Dojo	Code compliant with .net/MVC standards
Cycle 2			
Low velocity	UI complexity is slowing us down	Implement all but UI in one day	FAIL: everything is slowing us down.
Cycle 3			
Low velocity	We don't understand the code from team mates	Mob Programming complex refactoring	


www.MobProgramming.org
9:00 AM
Driver
Product owner
15 minute rotations

Experiment

Refactoring of customer table

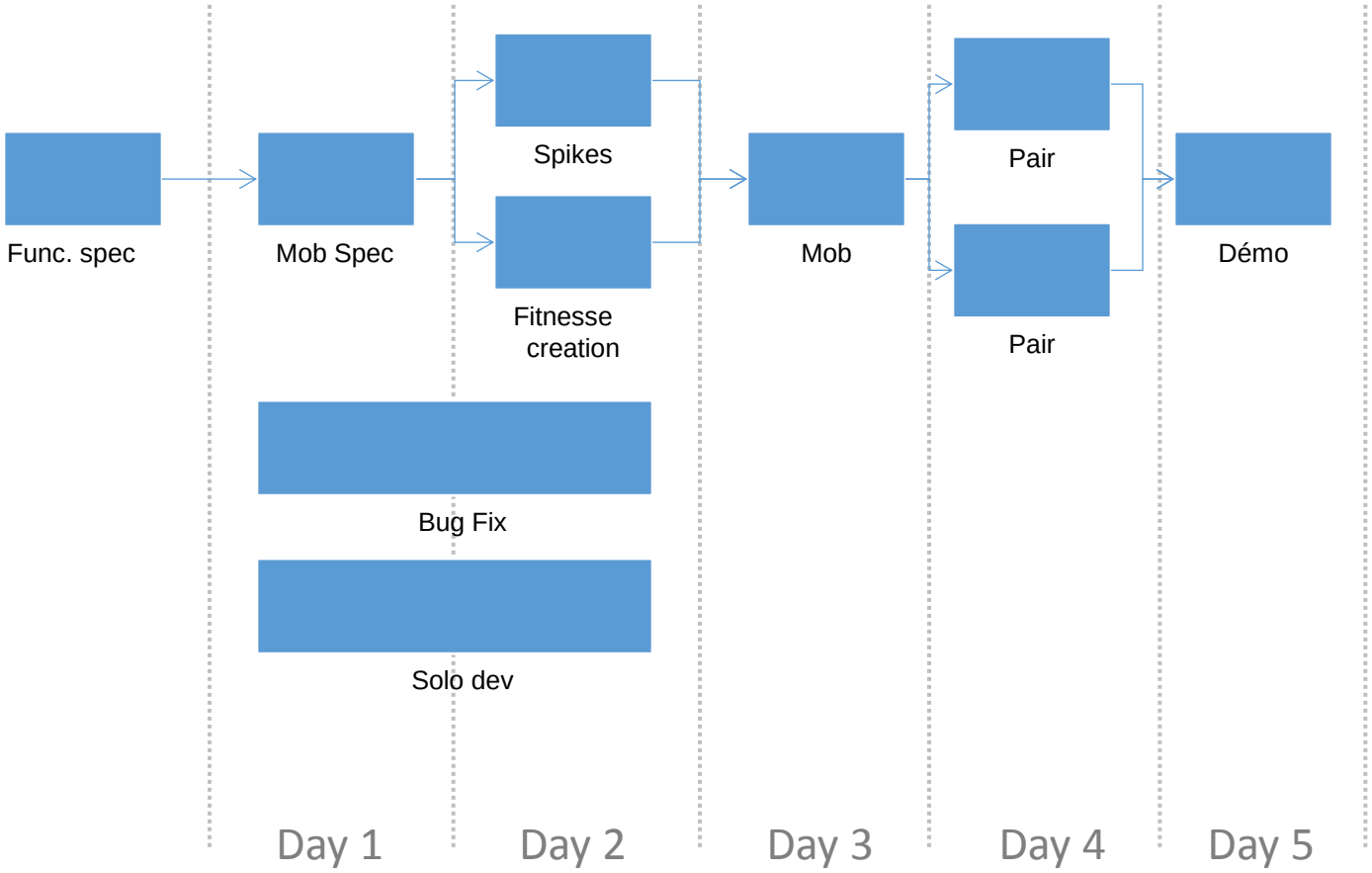
Impact all over the application

4 developers in a meeting room. No experts.


Photo Zach Dischner - CC BY 2.0

Observation	Hypothesis	Experiment	Result
Cycle 1			
Bad quality of code	We dont know the technology	Training with cycle of Dojo	Code compliant with .net/MVC standards
Cycle 2			
Low velocity	UI complexity is slowing us down	Implement all but UI in one day	FAIL: everything is slowing us down.
Cycle 3			
Low velocity	We don't understand the code from team mates	Mob Programming complex refactoring	Faster creation of quality code

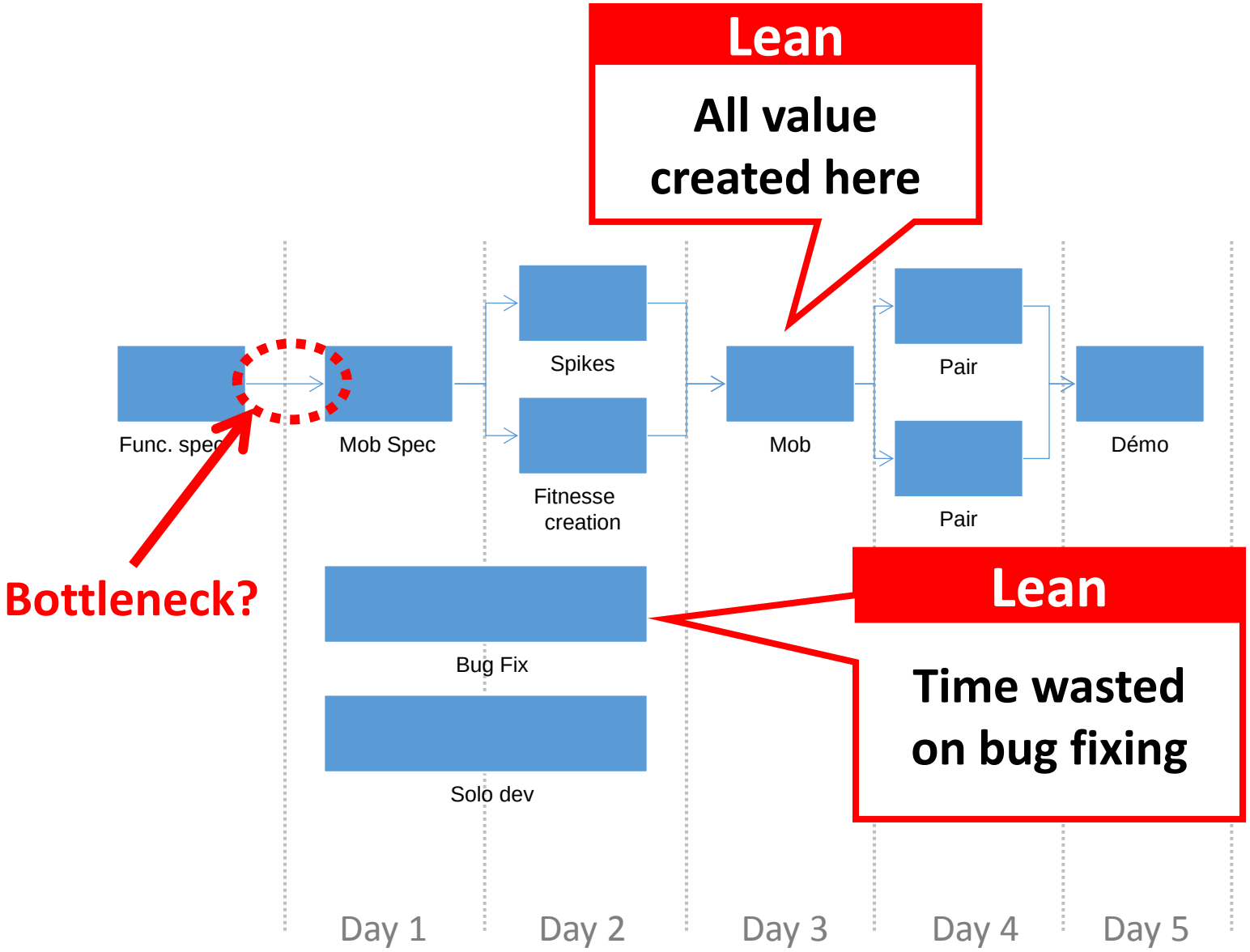
Observation	Hypothesis	Experiment	Result
Cycle 1			
Bad quality of code	We dont know the technology	Training with cycle of Dojo	Code compliant with .net/MVC standards
Cycle 2			
Low velocity	UI complexity is slowing us down	Implement all but UI in one day	FAIL: everything is slowing us down.
Cycle 3			
Low velocity	We don't understand the code from team mates	Mob Programming complex refactoring	Faster creation of quality code
Cycle 4			
Time wasted understanding specs	Specs are not detailed enough with users' input	Mob Specs	Less time wasted during mob prog.


Func. spec
Mob Spec
Spikes
Fitnesse creation
Bug Fix
Solo dev
Mob
Pair
Pair
Démo
Day 1
Day 2
Day 3
Day 4
Day 5

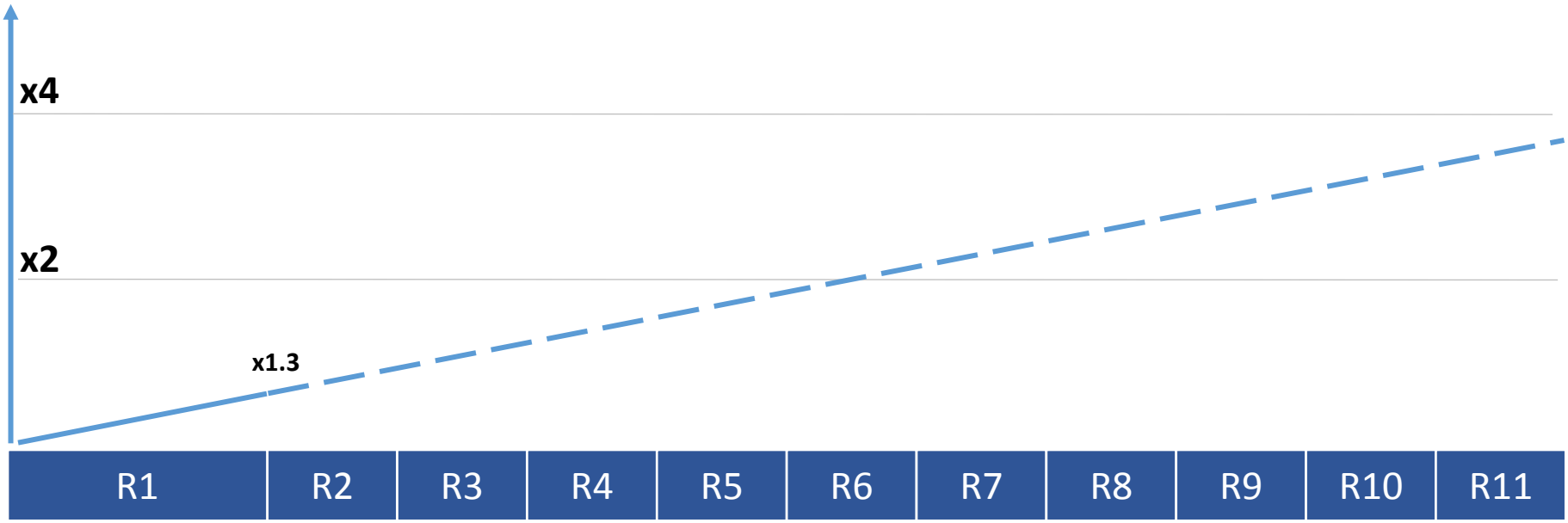
Productivity?


FAIL

X 1.3


Lean
All value created here
Spikes
Pair
Func. spec
Mob Spec
Mob
Démo
Fitnesse creation
Pair
Bottleneck?
Lean
Bug Fix
Time wasted on bug fixing
Solo dev
Day 1
Day 2
Day 3
Day 4
Day 5

Area for more improvement



Conclusion

Gemba = code

Continuous improvement using one week iterations

Bottleneck = sharing knowledge within team

Solution = learn as a team (Dojo, Mob Programming…)

We do need to learn everywhere

Design process as a learning acquisition activity

Questions?

« I want to try! »