

SOFT620020.01
Advanced Software
Engineering

Bihuan Chen, Pre-Tenure Assoc. Prof.

bhchen@fudan.edu.cn

<https://chenbihuan.github.io>

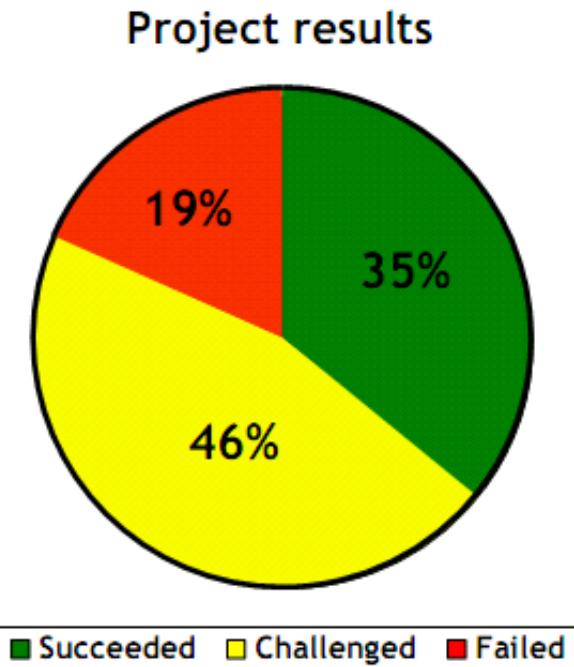
Course Outline

Date	Topic	Date	Topic
Sep. 09	Introduction	Nov. 04	Mobile Testing
Sep. 16	Testing Overview	Nov. 11	Delta Debugging
Sep. 23	Guided Random Testing	Nov. 18	Bug Localization
Sep. 30	Search-Based Testing	Nov. 25	Presentation 2
Oct. 12	Performance Analysis	Dec. 02	Automatic Repair
Oct. 14	Presentation 1	Dec. 09	Symbolic Execution
Oct. 21	Security Testing	Dec. 16	Scrum
Oct. 28	Compiler Testing	Dec. 23	Presentation 3

1

Why Agile?

Facts



From 2006 Standish Report: Based on analysis of more than 40,000 Projects over 10 years

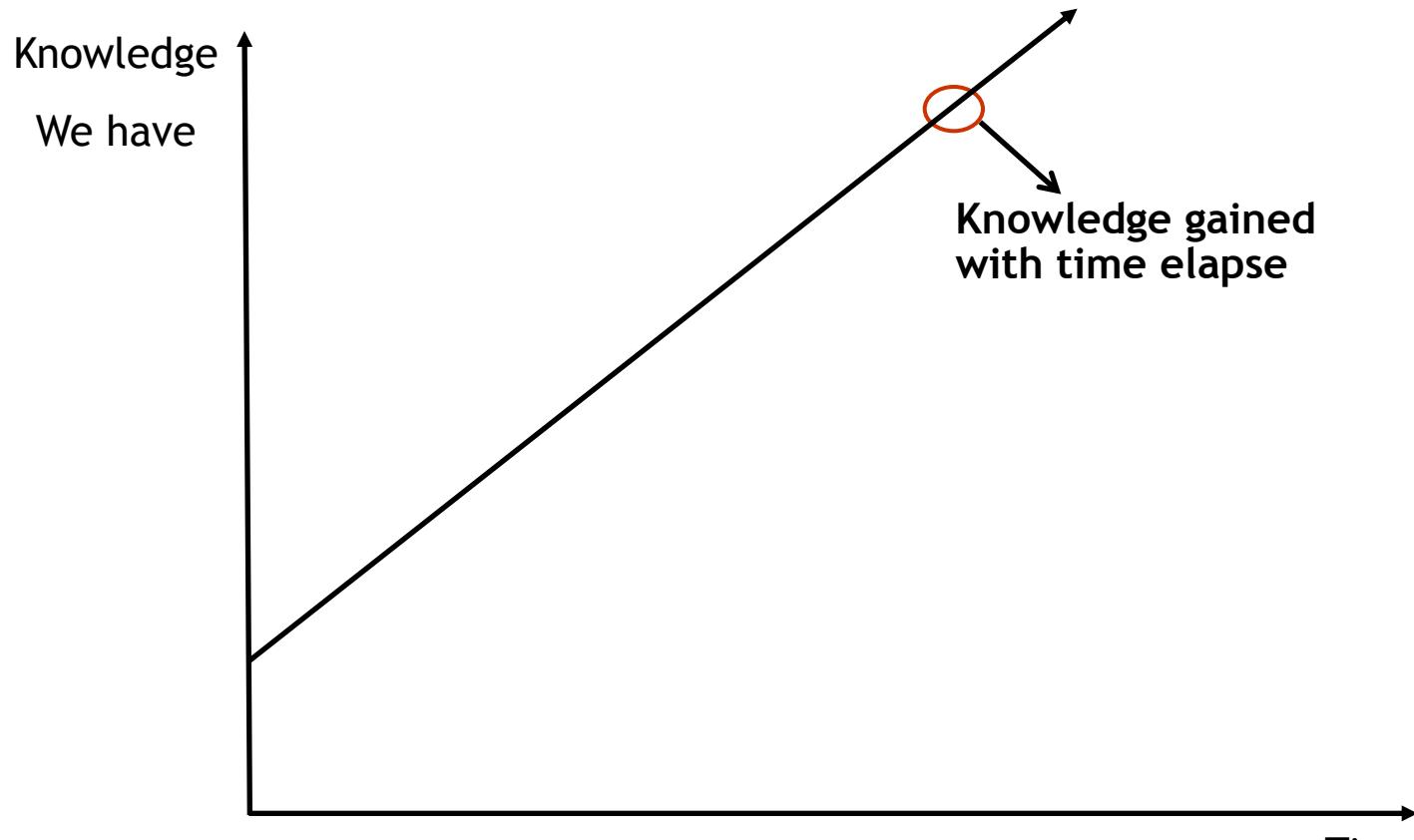
Facts

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

Discussion – Why software development is so difficult?

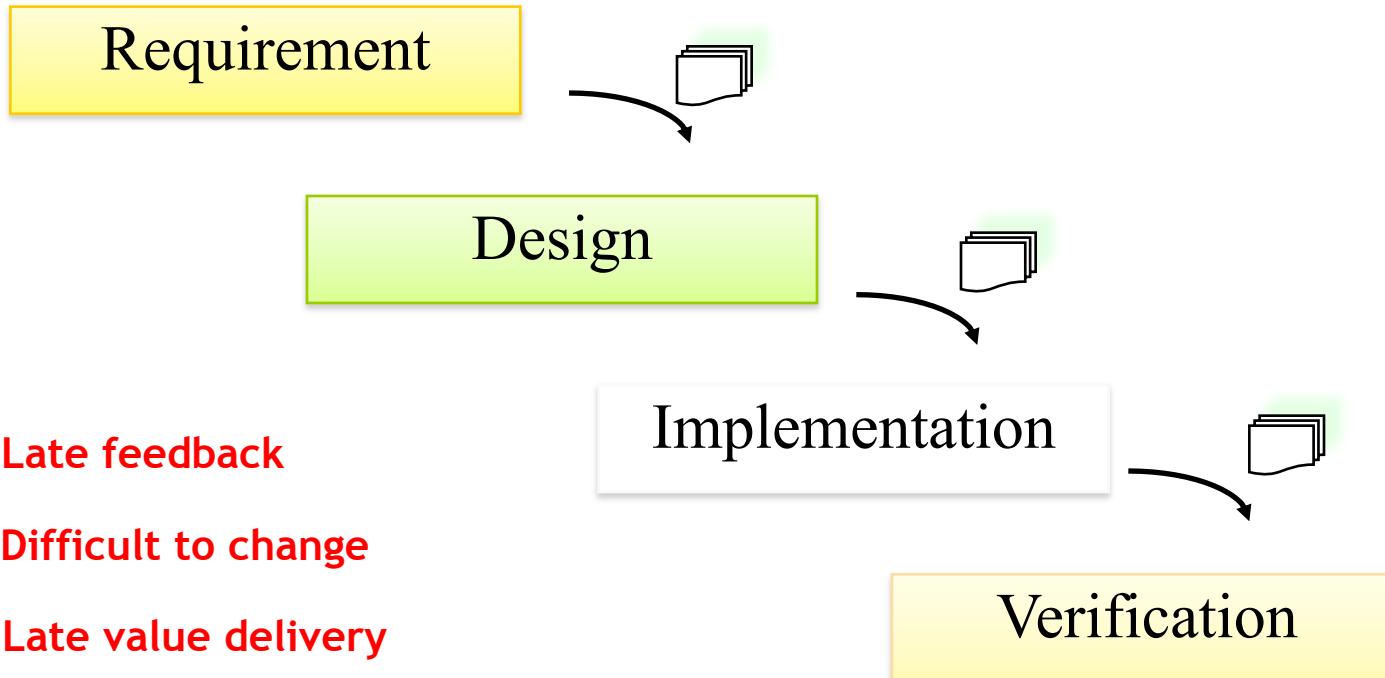


Why software development is so difficult



Software development is discovery

Waterfall – what we following before...



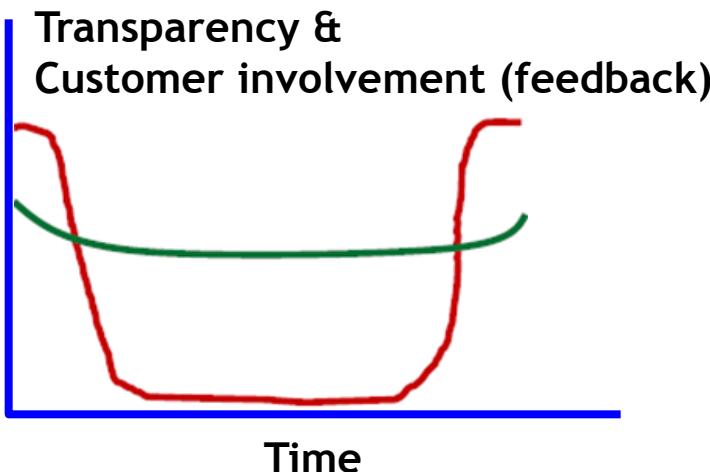
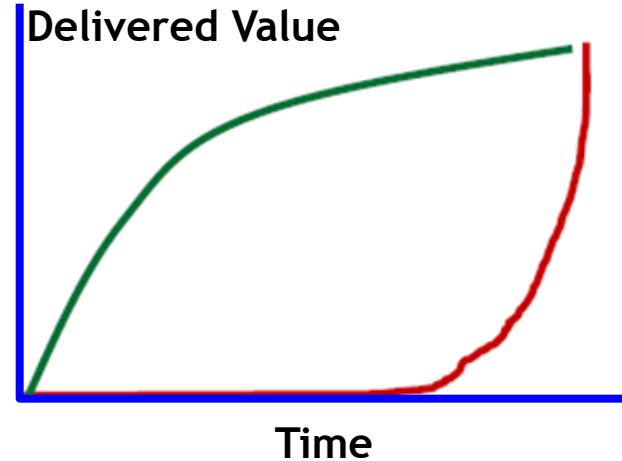
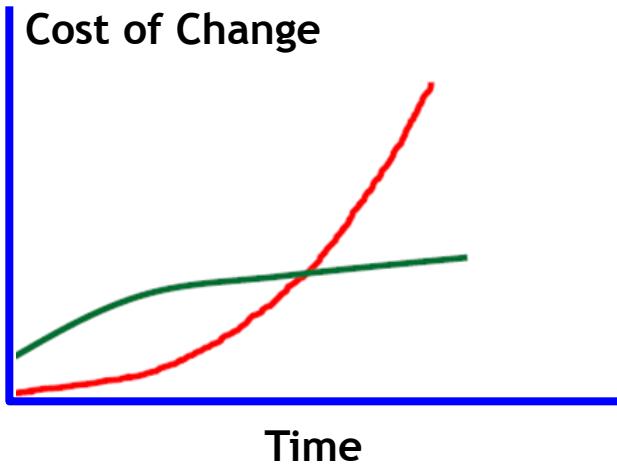
Iterative development



Iterative development



Iterative development vs. waterfall



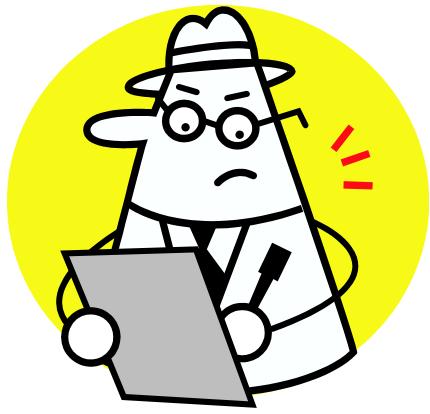
Software development

“Software is the most complex task ever undertaken by humanity...”

- Dr. Frederick Brooks
(recipient of the ACM Turing award)

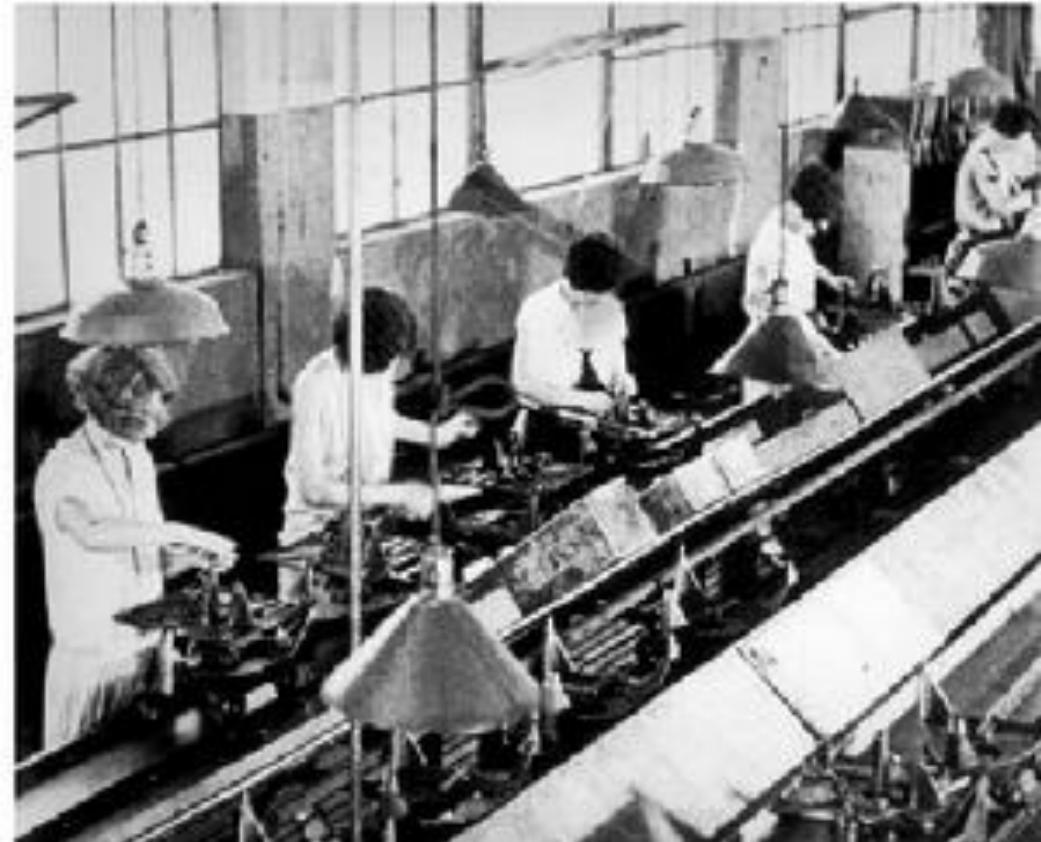
Software development is an R&D activity with high levels of inherent variability, creativity, and knowledge creation. It is not predictable, repeatable manufacturing.

Challenge of Defined Process



Plan the work

Work the plan



Empirical Process Thrives



Control vs. Self-Organize



It's All about People

- Top three factors in software development



People



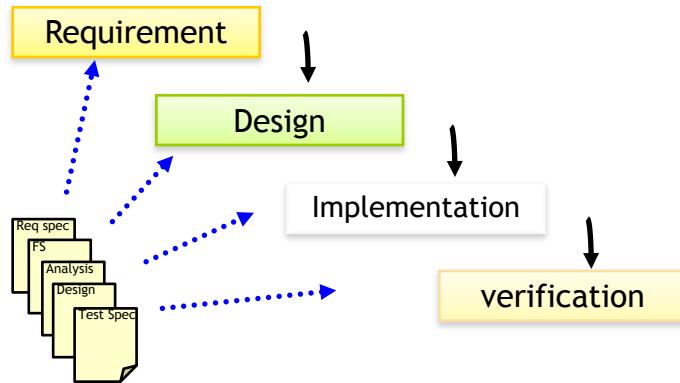
People



People

- Tom DeMarco on Peopleware

“Why agile?”

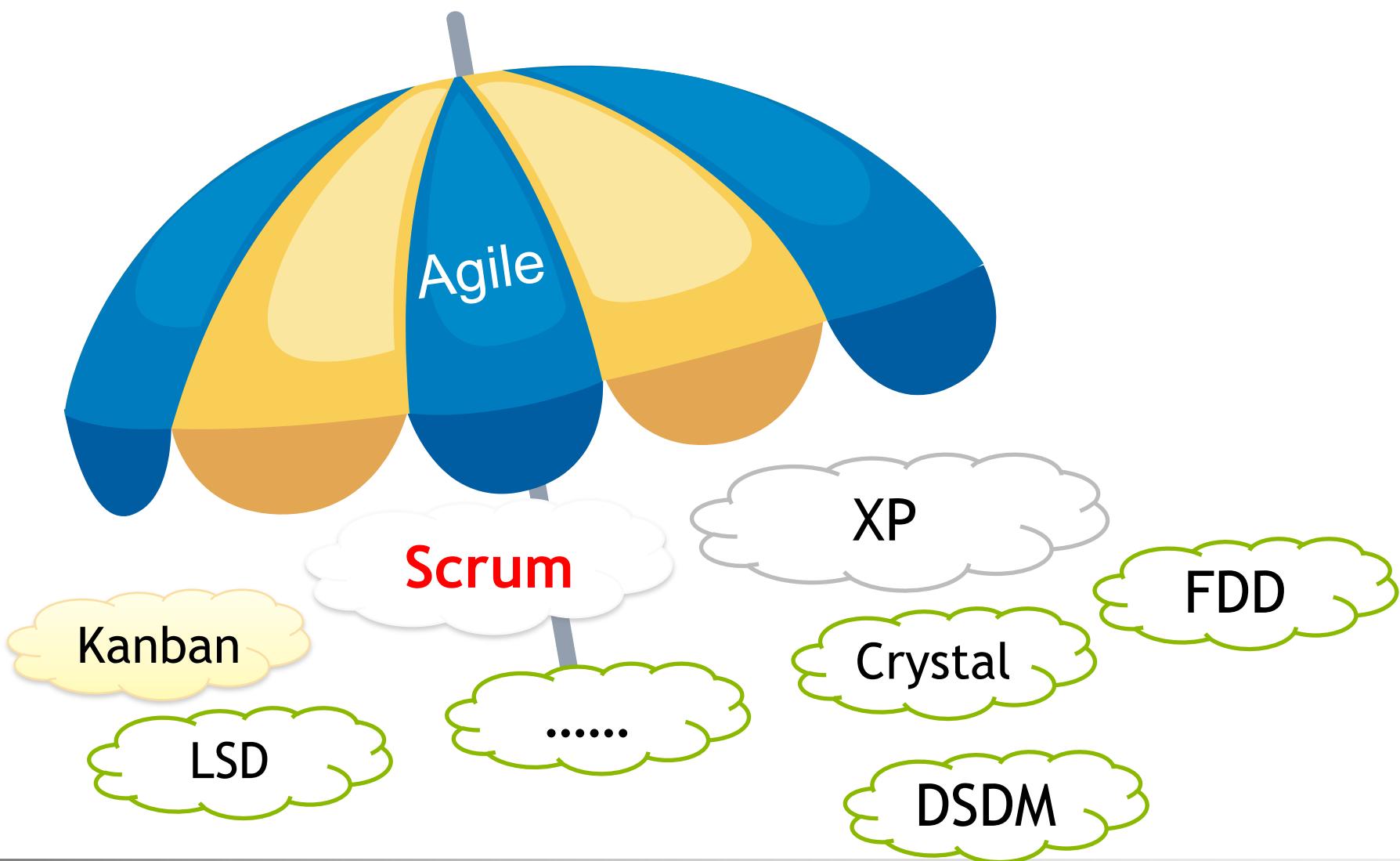


Challenge of waterfall



Difficulty of knowledge worker motivation

Agile is a Umbrella Term



Agile Manifesto

www.agilemanifesto.org

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.

Discussion – Does agile development mean disorganization?



2 Scrum Basic

Scrum

[http://en.wikipedia.org/wiki/Scrum_\(rugby\)](http://en.wikipedia.org/wiki/Scrum_(rugby))

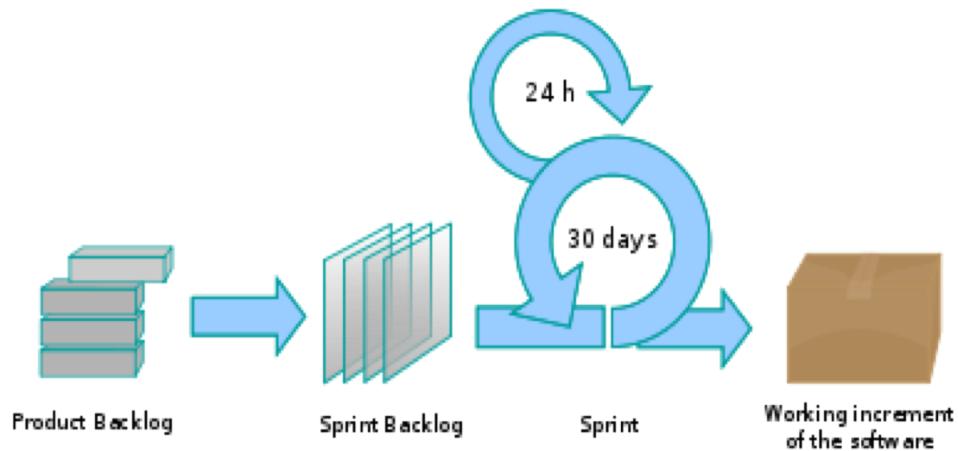
Scrum, in the sports of [rugby union](#) and [rugby league](#), is a way of restarting the game, either after an accidental infringement or when the ball has gone out of play



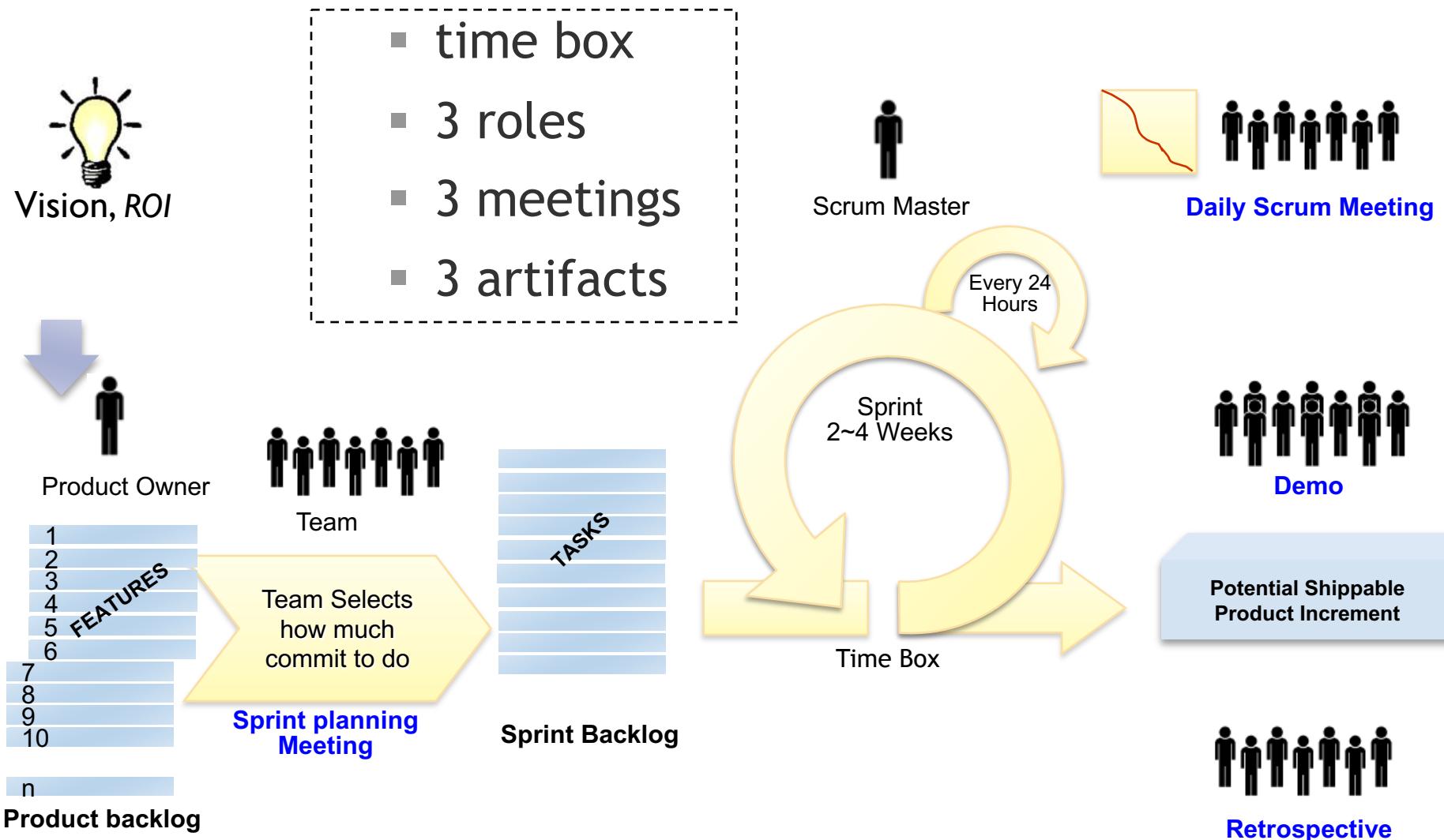
Scrum

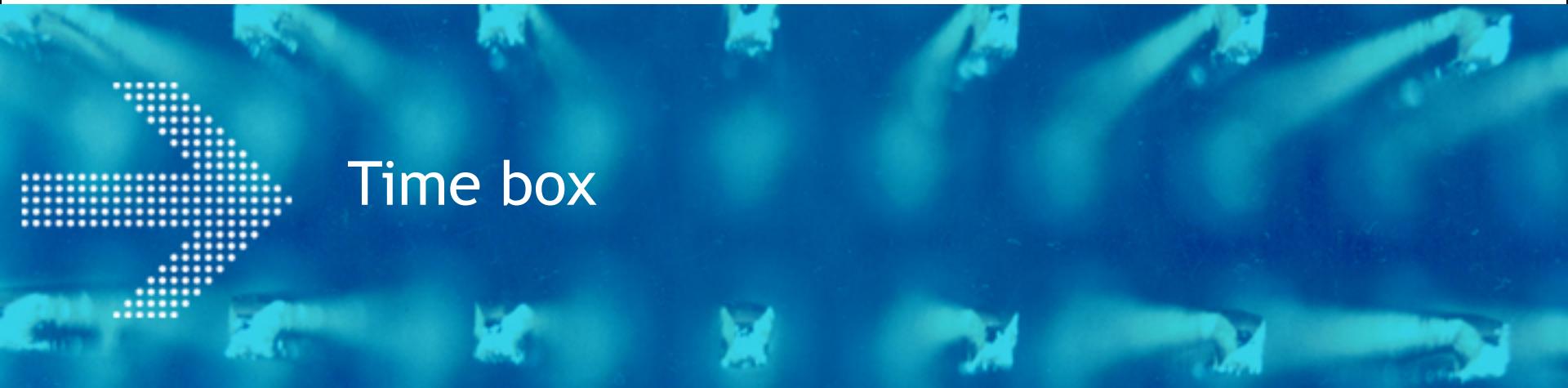
[http://en.wikipedia.org/wiki/Scrum_\(development\)](http://en.wikipedia.org/wiki/Scrum_(development))

Scrum is an iterative incremental framework for managing complex work (such as new product development) commonly used with agile software development.



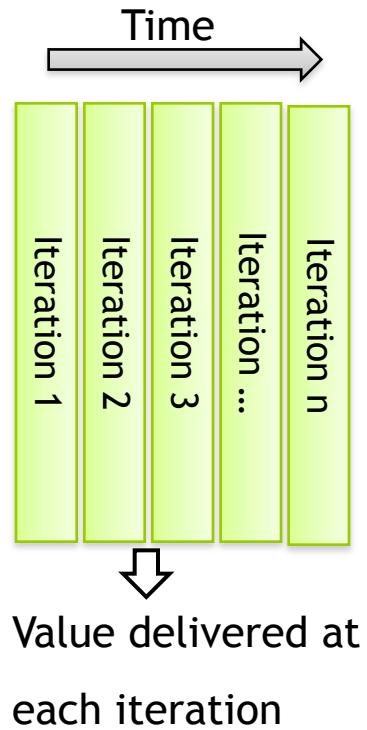
Scrum overview



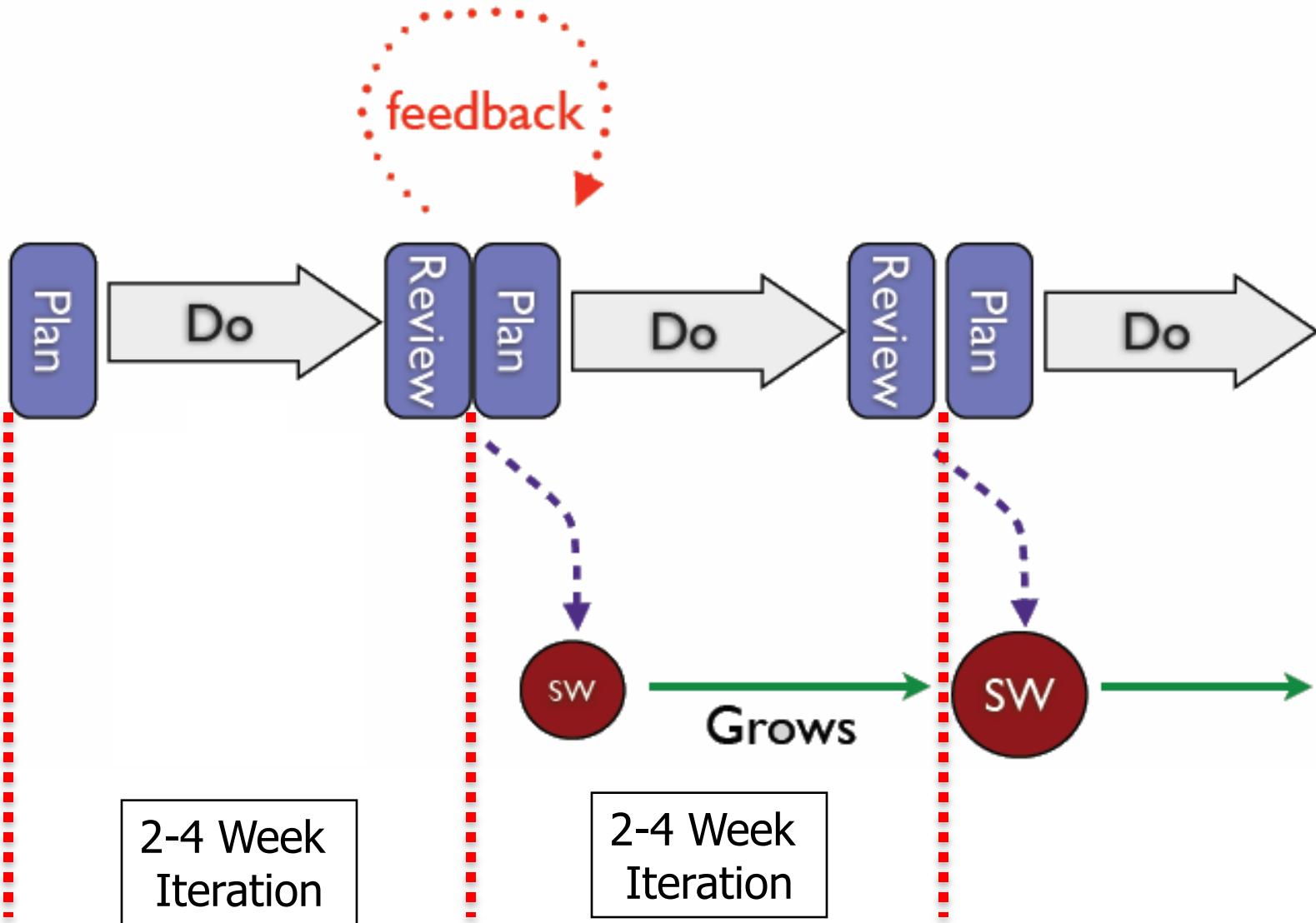


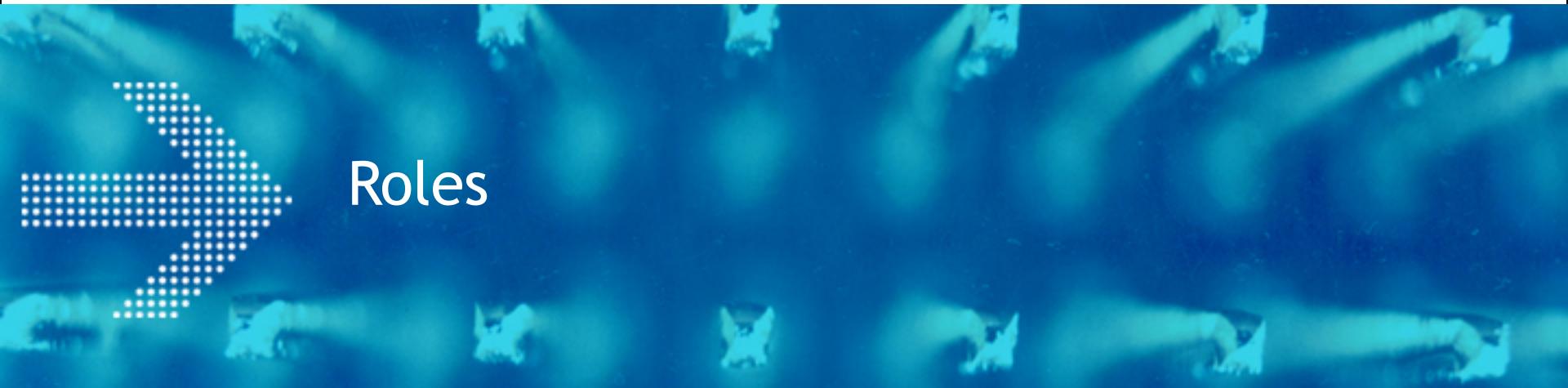
Time box

Time box



Time box

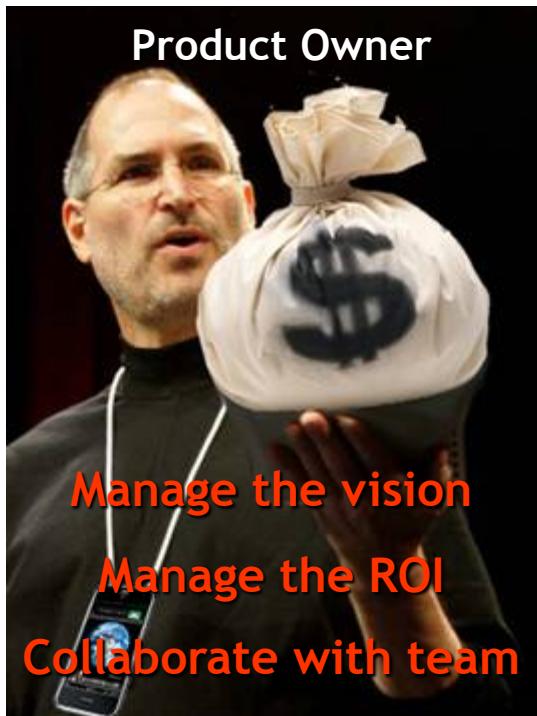




Roles

3 Roles

Product Owner



Manage the vision

Manage the ROI

Collaborate with team

Scrum Master



Manage the process

Shield the team

Remove obstacles

Team



Self organized

Cross functional

Commit for the result

3 Roles

Define the features of the product, decides on release date and content

Is responsible for the profitability of the product (ROI)

Prioritize features according to the market value

Can change features and priority every sprint

Accept or reject work result



Product Owner



Team

Cross-functional, 7+/-2 members

Has the right to do everything within the boundaries to reach goal

Organize and manage itself and its work

Review work results with the product owner

Ensure that the team is fully functional and productive

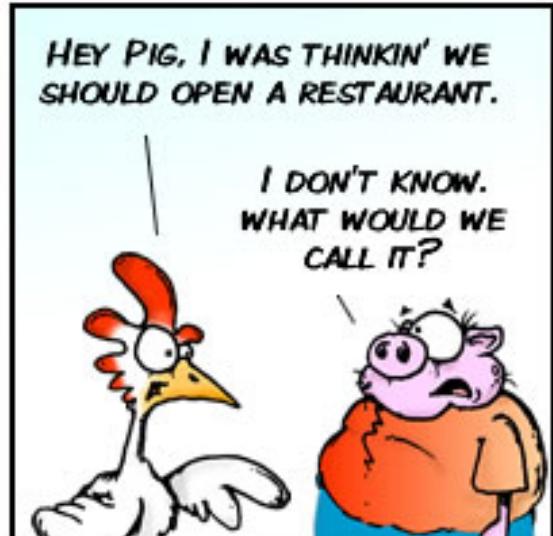
Enable close cooperation across all roles and functions, and remove barriers

Shield the team from external interferences

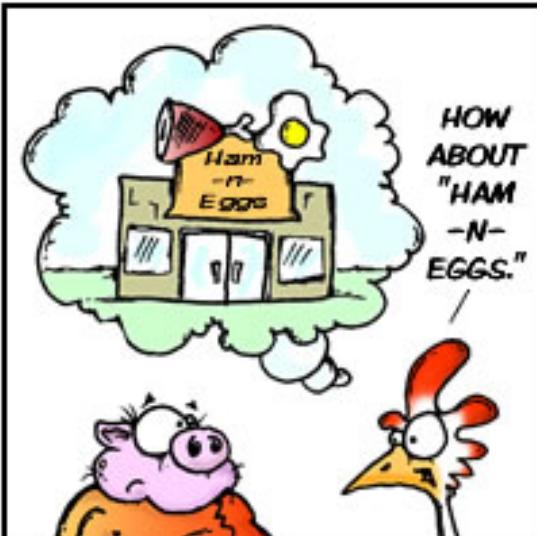


Scrum Master

Roles in Scrum



By Clark & Vizzini



© 2006 implementingscrum.com

The Scrum world is divided into Pigs and Chickens

Definition of Done

Team together with product owner defines what “done” means

Examples of “done”

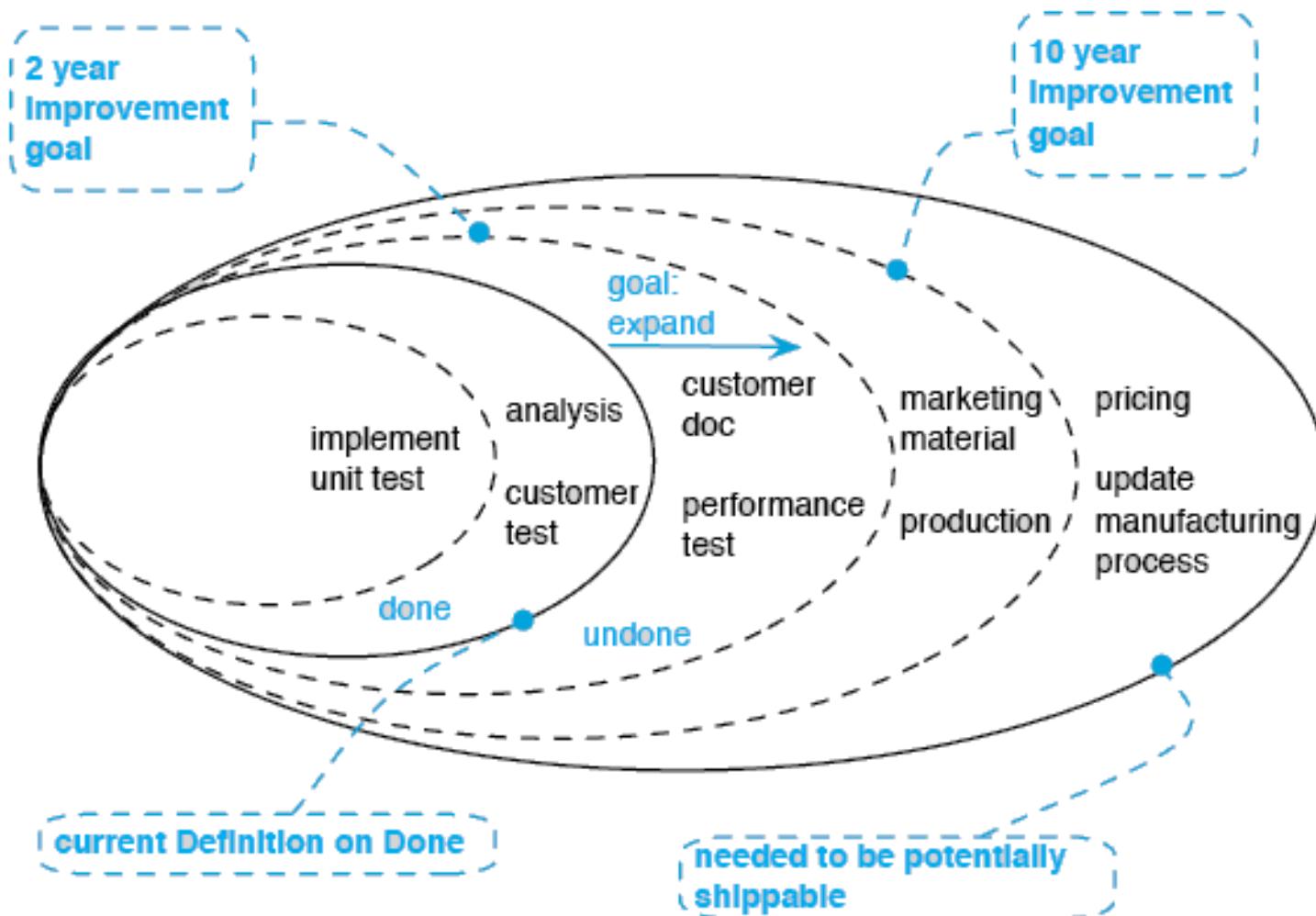
- Design, coding, unit testing, integrated
- Static analysis, refactored
- Acceptance tested, deployable

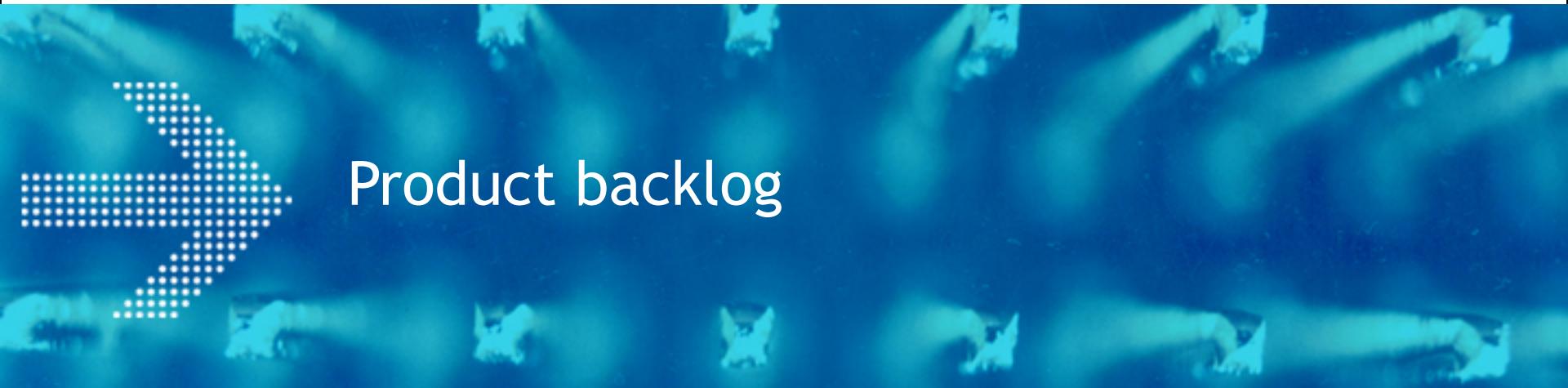
Done defines the current technical capability of the team

- Over time Done should include everything needed before deployment

Not done backlog items should not be reviewed

Extending DoD





Product backlog

Release planning

Sprint Planning

Daily Run

Sprint Review



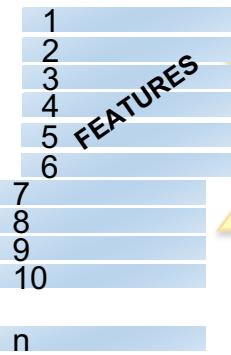
Vision, ROI



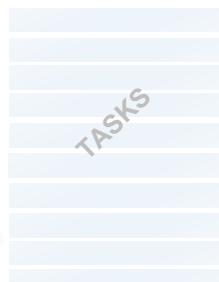
Product Owner



Team



Team Selects how much commit to do
Sprint planning Meeting



Sprint Backlog

Scrum Master

Every 24 Hours

Sprint
2~4 Weeks

Time Box

Daily Scrum Meeting



Demo

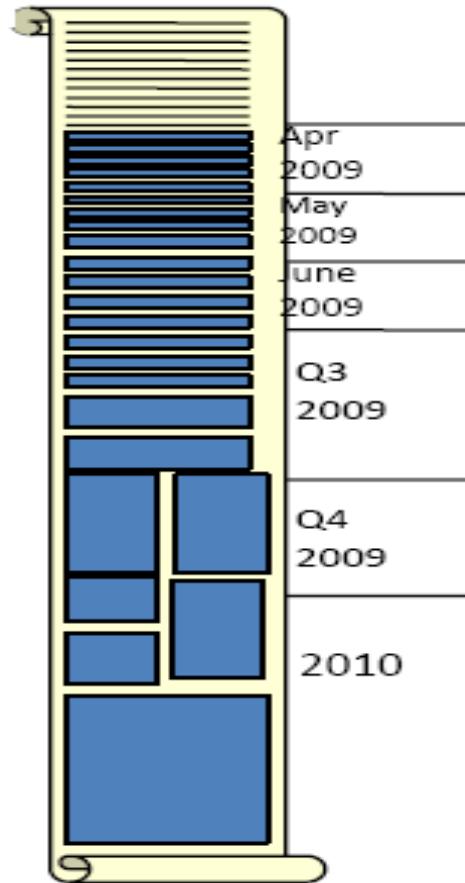
Potential Shippable Product Increment



Retrospective

Product Backlog

Detailed Properly
Estimated
Emergent
Prioritized



User Story

Card

Conversion

Confirmation

User Story

As an <type of user>

Front Side

I can/I want to <some goal>

So that <some reason>

Est: xx

Pri: xx

Acceptance Criteria

Back Side

.

.

.

Notes

.

.

“As a system admin, I want to see the alarm being reported when board is missing, so that I can know when certain board signal is lost”

Alarm reported with right slot ID

Know the time the alarm happen

Alarm be reset when board is re-plugged in

Severity of the alarm can be configured from 1~5

Release Planning - What is Good Story

Independent: *Implement in any order, no overlap in primary concepts*

Negotiable: *Possibly more or less of a feature*

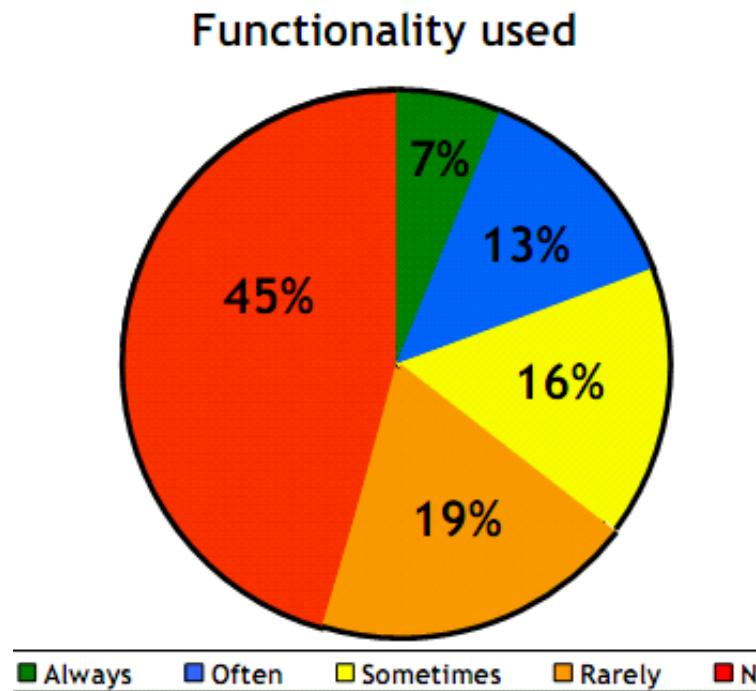
Valuable: *Story needs to be valuable to the customer*

Estimable: Understood well enough to be estimated at the level required

Small: *If too large, scope, deliverability, and testability risks incurred*

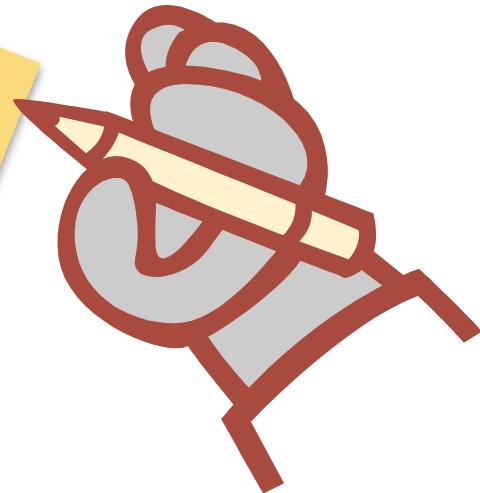
Testable: *Must have a verifiable definition of done—executable requirements are a best practice*

Valueable



From 2006 Standish Report: Based on analysis of more than 40,000 Projects over 10 years

Exercise – Writing User Story

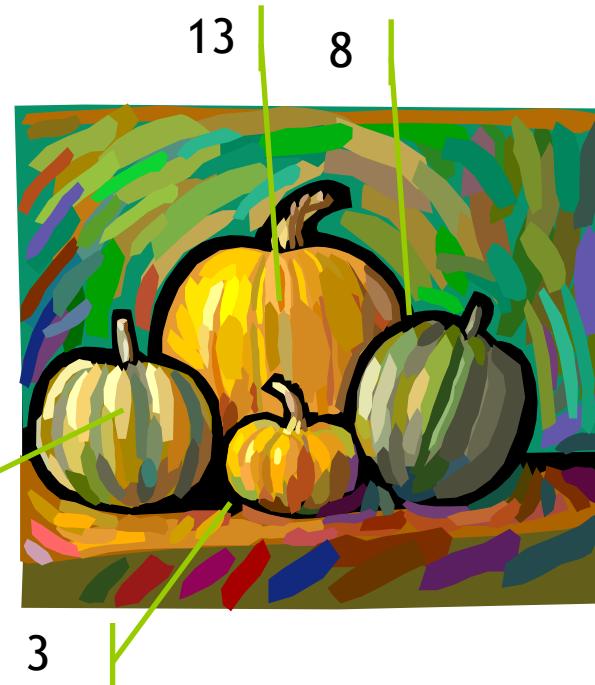


Story Size Estimation



5
(base point)

Comparative to
one base point



Release Planning – Planning Poker

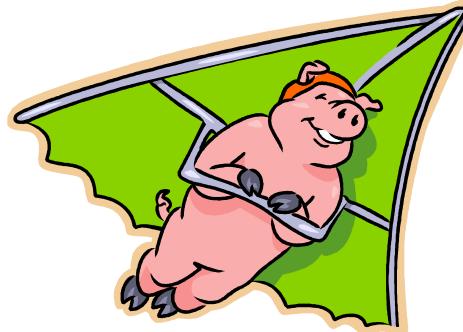
- Product Owner explains story
- Team discusses work involved
- Everyone estimates individually
- Everyone reveals estimates simultaneously
- Lowest and highest estimates are justified
- Repeat until estimates converge



Exercise – User Story Size Estimation



Why does Planning Poker work



**Those Who Do the Work
Make the Estimates**



Independent

Confirmation



Conversation



Acceptance testing user story

DS bit rate L2	US bit rate L2	DS bit rate L3	US bit rate L3	L1 state	L2 state	L3 state	BG init time	DS actual intlv delay L1	US actual intlv delay L1	DS actual intlv delay L2	US actual intlv delay L2	DS actual intlv delay L3	US actual intlv delay L3	alarms	comment
40 Mb/s	10 Mb/s	40 Mb/s	10 Mb/s	part of BG	part of BG	part of BG	90s	5ms	8ms	5ms	8ms	5ms	8ms	none	typical success case
40 Mb/s	10 Mb/s	40 Mb/s	10 Mb/s	part of BG	part of BG	part of BG	>150s	5ms	8ms	5ms	8ms	5ms	8ms	none	not acceptable because out of spec (init time)!

Given... When... Then

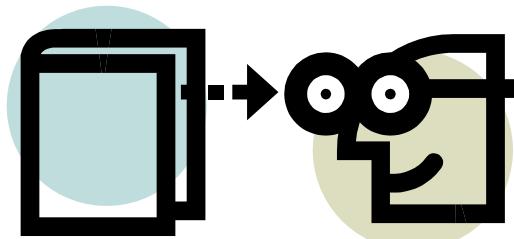
Prioritize the Product Backlog - How



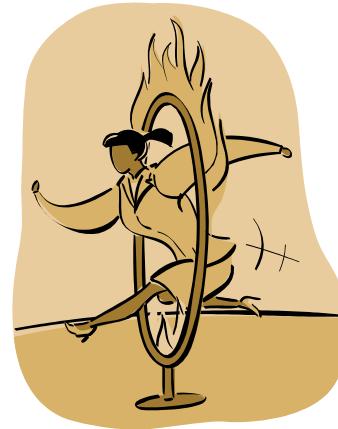
Financial Value



Cost of Development

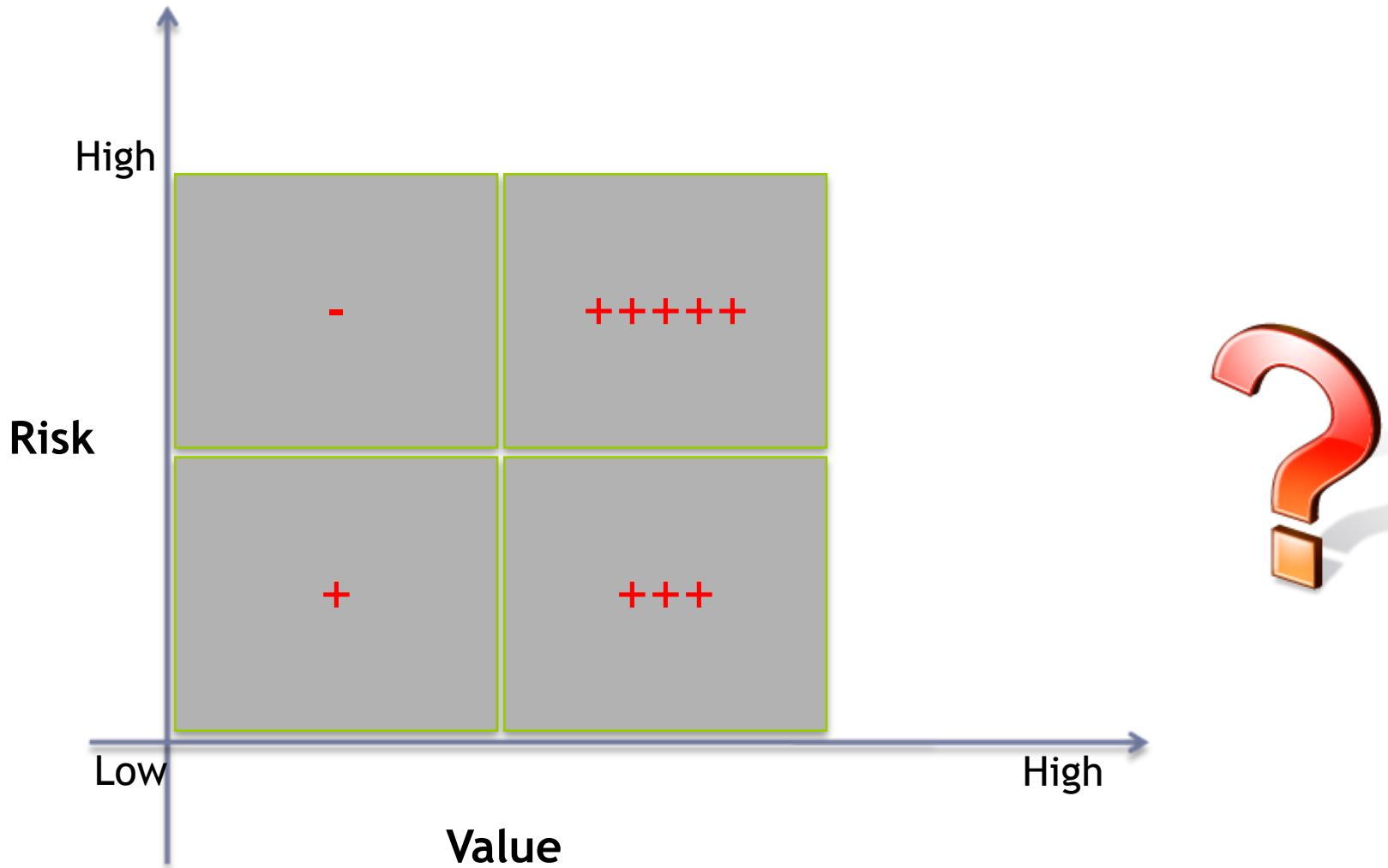


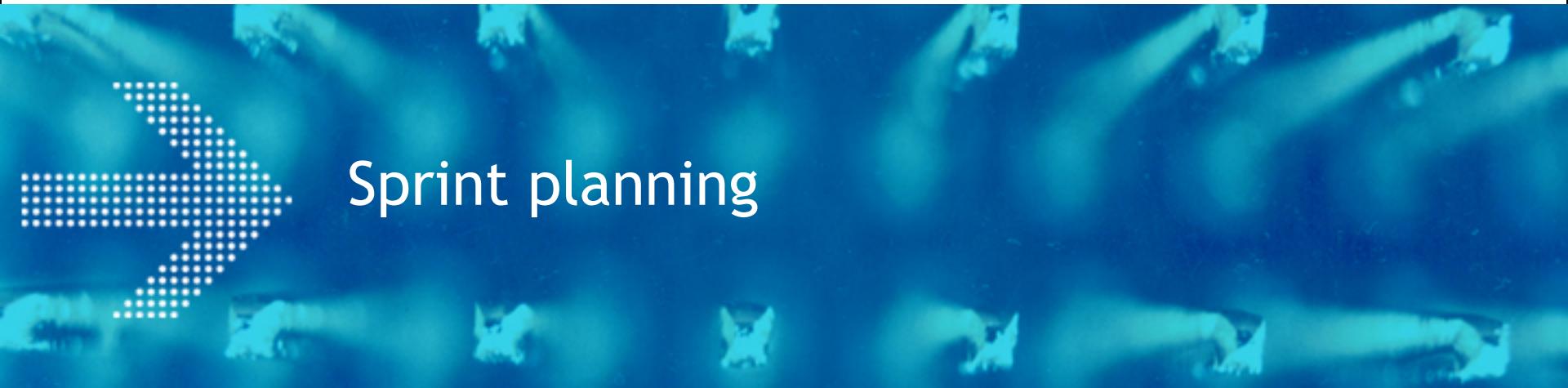
Knowledge Gained



Risk Removal

Prioritize the Product Backlog





Sprint planning

Sprint Planning

Release planning

Sprint Planning

Daily Run

Sprint Review



Vision, ROI



Product Owner



Team

Team Selects how much commit to do

Sprint planning Meeting



n Product backlog

Sprint Planning



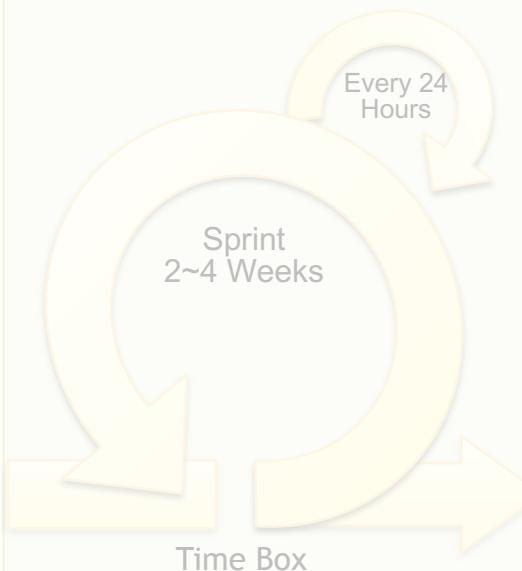
Sprint Backlog

Daily Run



Scrum Master

Every 24 Hours



Sprint Review



Daily Scrum Meeting



Demo

Potential Shippable Product Increment



Retrospective

Sprint Planning - Target

Target defined

- Stories committed for the sprint Decided
- Sprint Goal composed

Break down into detail plan

- Tasks (with idea hours) to get each story done defined
- Team commit to the plan

Informative Radar created

Sprint Planning - Informative Radar

Stories	Tasks (To be done)	On-going	Done	Approved

story

task

New task



Daily running

Daily Run

Release planning

Sprint Planning

Daily Run

Sprint Review



Vision, ROI



Product Owner

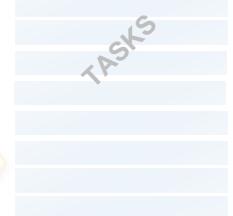


Team



Team Selects how much commit to do

Sprint planning Meeting



Sprint Backlog



Scrum Master

Every 24 Hours

Sprint 2~4 Weeks

Time Box



Daily Scrum Meeting



Demo

Potential Shippable Product Increment



Retrospective

Daily Running

Ruler

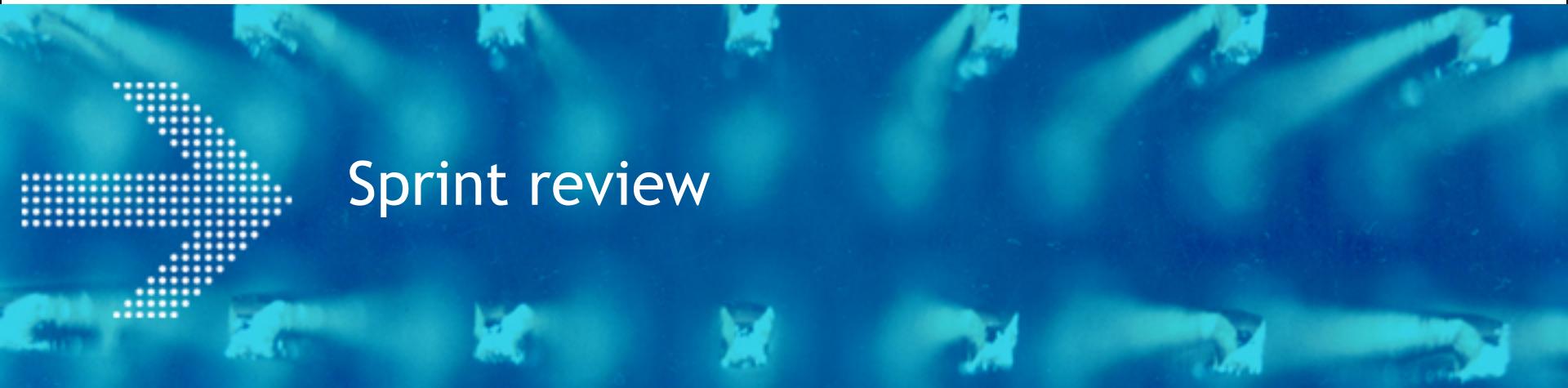
1. Same Time, Same Place, Every Day
2. Chickens must be quiet
3. 15 minutes or less, No solutions debate!

Provide answers to team for 3 questions

1. What did I complete yesterday?
2. What are I committing to today?
3. Do you have any roadblocks?

Sprint backlog

STORY	PLANNED	ONGOING	DONE
Associate user Send new User profile to IT via LT	2 User profiles (associate user via LT, associate) 1 User profile (associate via LT, associate)	1 User profile - Associate via LT 1 User profile - Associate via LT 1 User profile - Associate via LT	ATC - ONGOING WITH SEND PROFILE TO IT COMMITTEE 11 2 3 P/F
100 VALUES AND RATE PEOPLE - COMPANY INTEGRATION	FRAMEWORK IMPLEMENTATION TEST	UPDATE (impl) DEVELOP (impl)	
100 Test strategy for new profile creation when user selected SSO login	DETERMINE PROPOSE TEST ACTION(S) FOR PROF	INVESTIGATE AVAILABILITY OF PST/ATC	ESTIMATE WORK NEEDED (SST/mz)
100 Move existing PRO profile CARRIER DEFAULT	Profile in migration 1st with Vip enabled → migrate default Profile Create	Profile migration → send to LT → maz	
100 Move specific PRO profile CARRIER PROFILE	Profile in migration 1st with Vip enabled → migrate default Profile Create	Profile migration → send to LT → maz	
100 Move specific PRO profile CARRIER PROFILE	ATC - CREATE PROFILE 8	RENAME & CHECK MAPPER CARRIER PROFILE 8	PERMANENT STORAGE 3
100 Move specific PRO profile DELETE	CREATE PROFILE CARRIER PROFILE 8		
100 Move specific PRO profile DELETE	ATC DELETE PROFILE 9	MIGRATE & CHECK PERMANENT STORAGE 4	DELETE PROFILE ERROR CASES 5ST
100 Move existing PRO profile CARRIER PROFILE	INPUT SERVICE + PARAMS + CONDITIONS	VALIDATE REMOTE STORAGE WITH LT	



Sprint review

Sprint Review

Release planning

Sprint Planning

Daily Run

Sprint Review



Vision, ROI



Product Owner

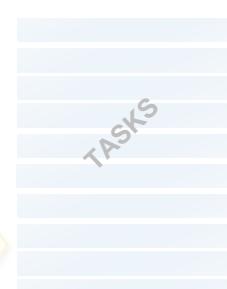


Team



Team Selects
how much
commit to do

Sprint planning
Meeting



Sprint Backlog



Scrum Master



Daily Scrum Meeting

Every 24
Hours

Sprint
2~4 Weeks

Time Box



Demo

Potential Shippable
Product Increment



Retrospective

Sprint Review – Demo

- ❑ Invite the whole world
- ❑ Team presents what was accomplished during the Sprint
 - Make it Informal and fun
 - Demo not selling or presenting impressive “slide ware.”
 - Stakeholders are encouraged to provide feedback and suggest priorities for the next iteration.
- ❑ Outcome
 - Knowledge sharing
 - Get feedback
 - Input for planning the next Sprint
 - Motivated team

Sprint Review – Retrospective

Making good teams Great

Sprint Review – Retrospective (cont.)

Who

- Team, Scrum Master, PO (depends)

Steps

- Recall: Gathering Information
- Brainstorming: Generate Insight
- Decide what to do (identify solution, improvements)

Why it works

- Short feedback time
- Short feedback loop
- Detailed information, deployable
- For my team

Sprint Review – Retrospective (cont.)

Key success factors

- Concrete Action
- Engaged people
- Focus Topic
- Uncover root cause

Three legs of Scrum



Scrum But - a Path to Fail

- ❑ No stand up
- ❑ No test for each sprint
- ❑ Still control rather than self-organize
- ❑ No retrospective
- ❑ No product backlog
- ❑ No Product owner
- ❑ Done is not Done
- ❑ No timeboxing
- ❑

Summarize – what is scrum about (1/2)

□ Doing right things

- Requirement are described in the view of customer by PO -- User story
- Run time feedback for the work done
 - Daily collaboration, clarification and confirmation
 - Feedback are highly appreciated, e.g. story approving, Demo, etc.
- Valid change to adapt to reality are welcome
- Doing right things @ right time/ sequence
 - Requirement are sorted in priority
 - Development are in the sequence of priority defined

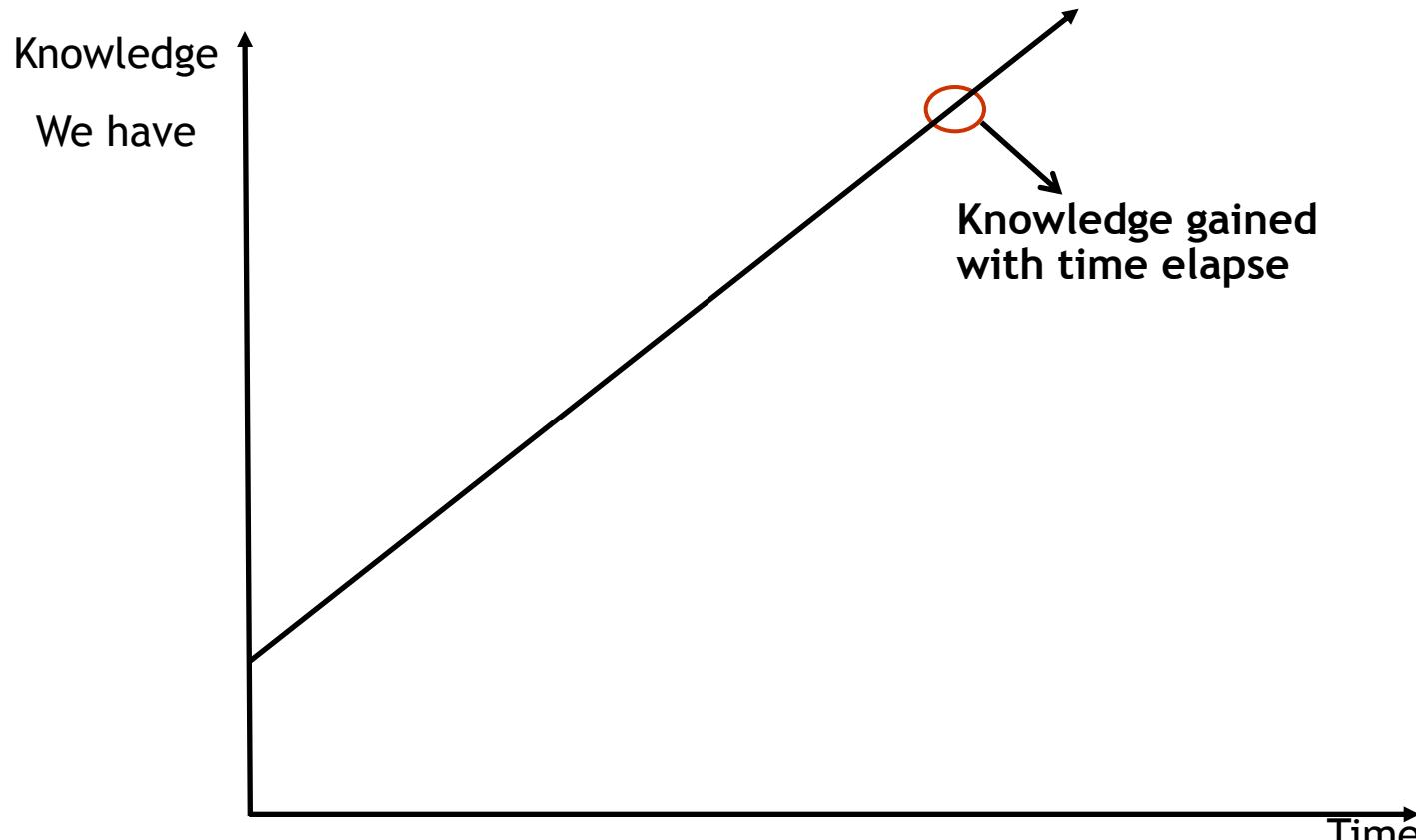
Summarize – what is scrum about (2/2)

- Doing things Right (by self organized team)
 - Team collaborate in the most efficient way
 - Team make decision on how to get things done,
 - Team commit for the result

- Continuous Improving
 - Get every thing visible
 - Shortest feedback loop
 - Clear and strong accountability
 - Regular retrospective

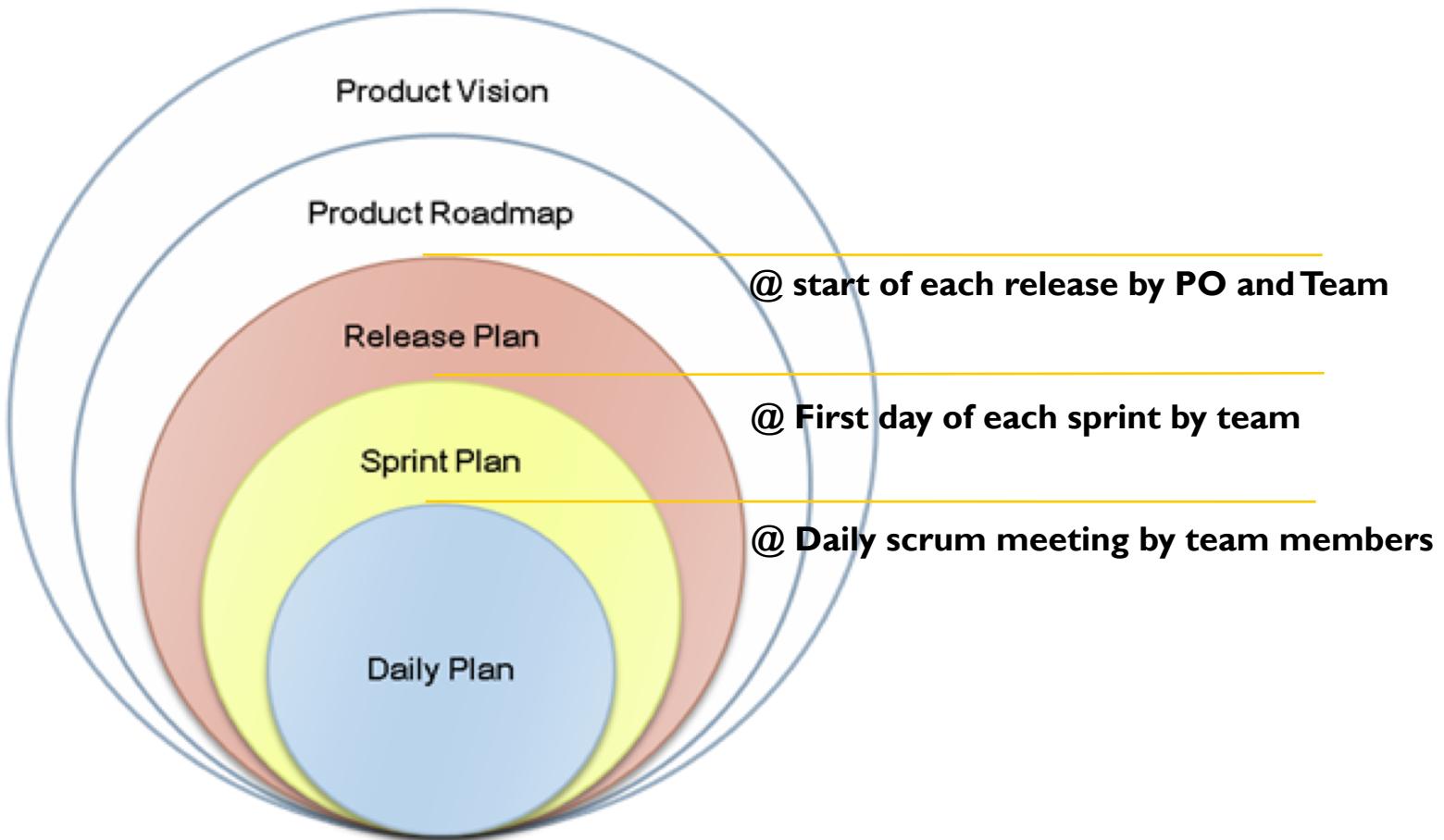
3 Release Planning

Why Software Development is So Difficult?



Software development is discovery

Mike Cohn's “Planning Onion”



Team's Speed - Velocity

Velocity is the amount of story points a team can deliver in a sprint

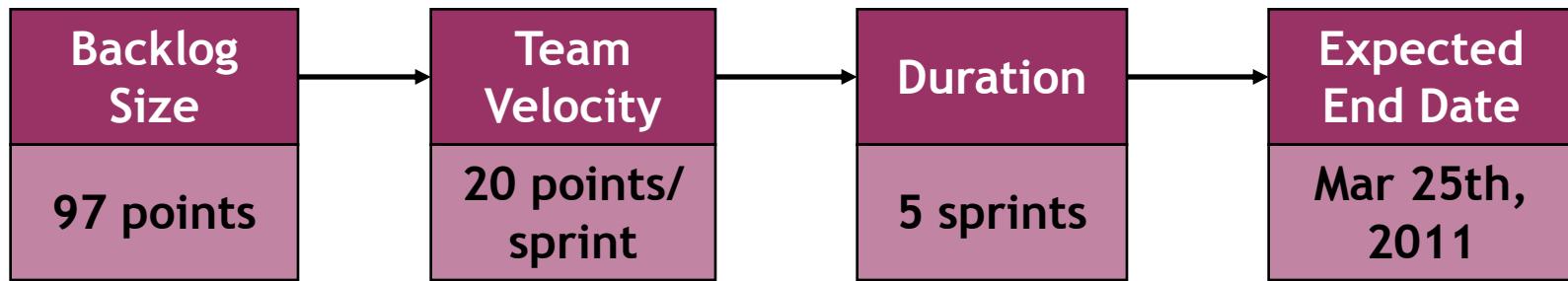
To estimate velocity:

- use historical values, if available
- run a few sprints, if possible
- break down a few items into tasks and make educated guess

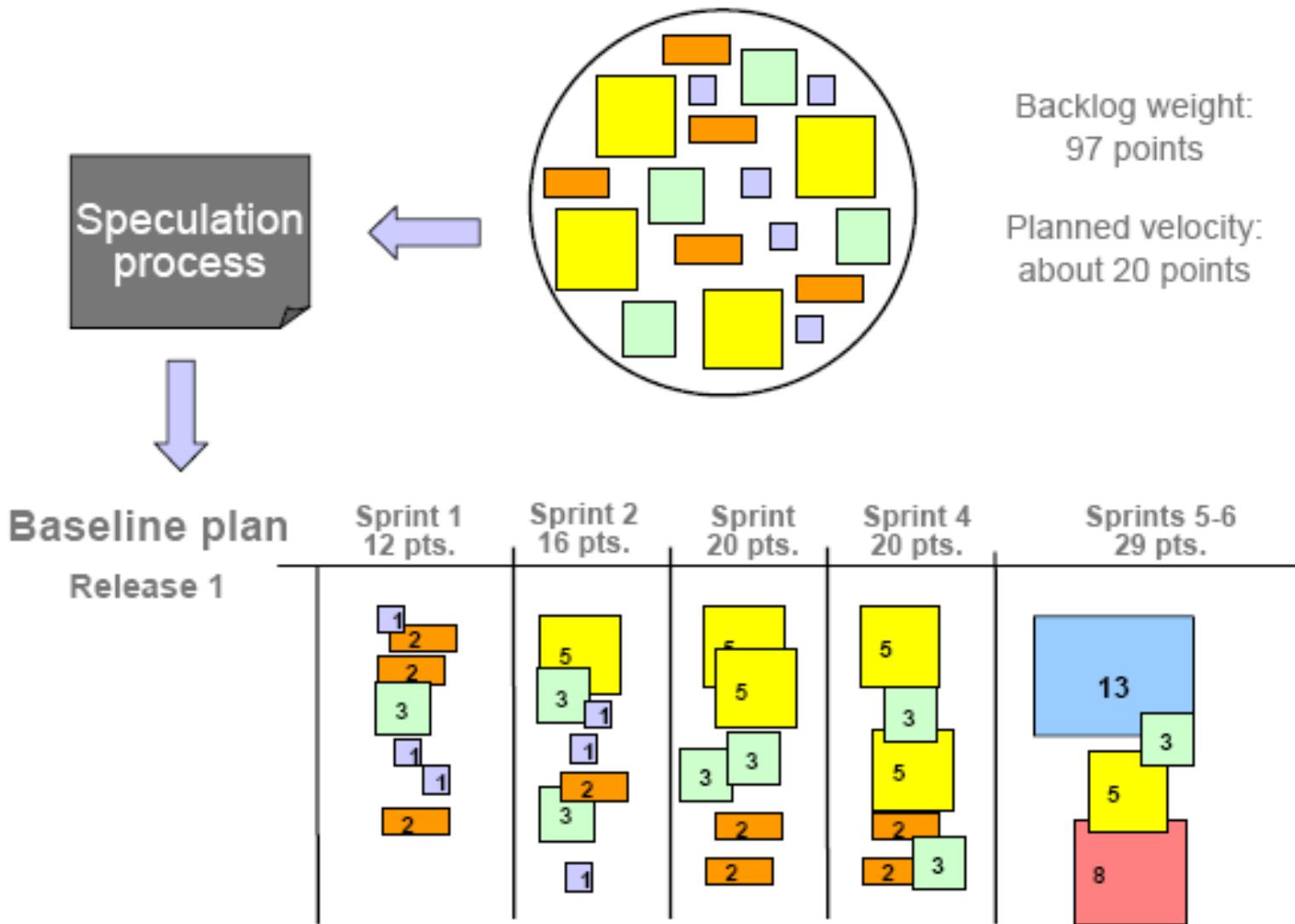
The pace is planned but it will change during the project lifetime

Use velocity range to express uncertainty

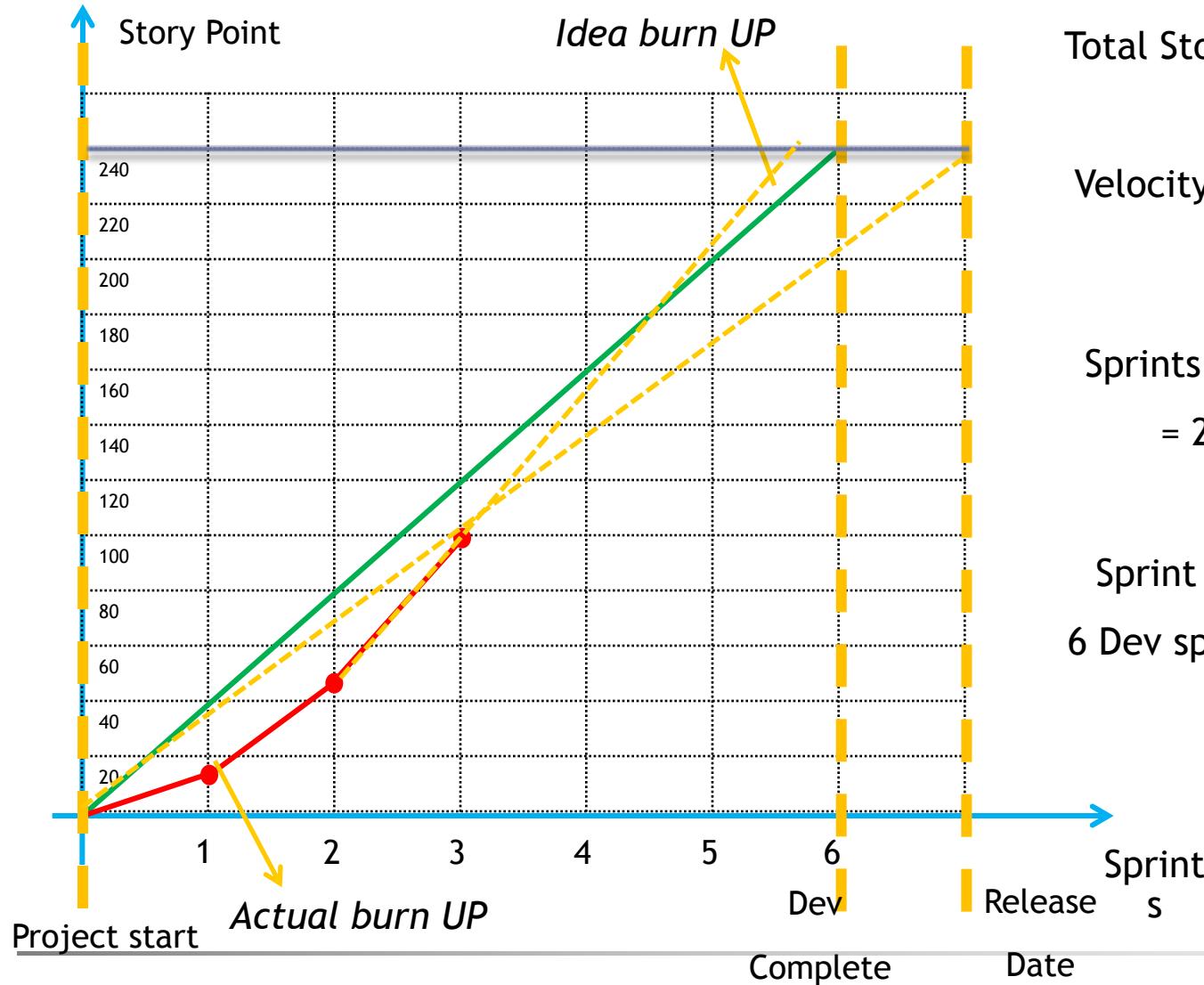
From Velocity to Duration



Release Plan



Release Burn Up (Forecast)



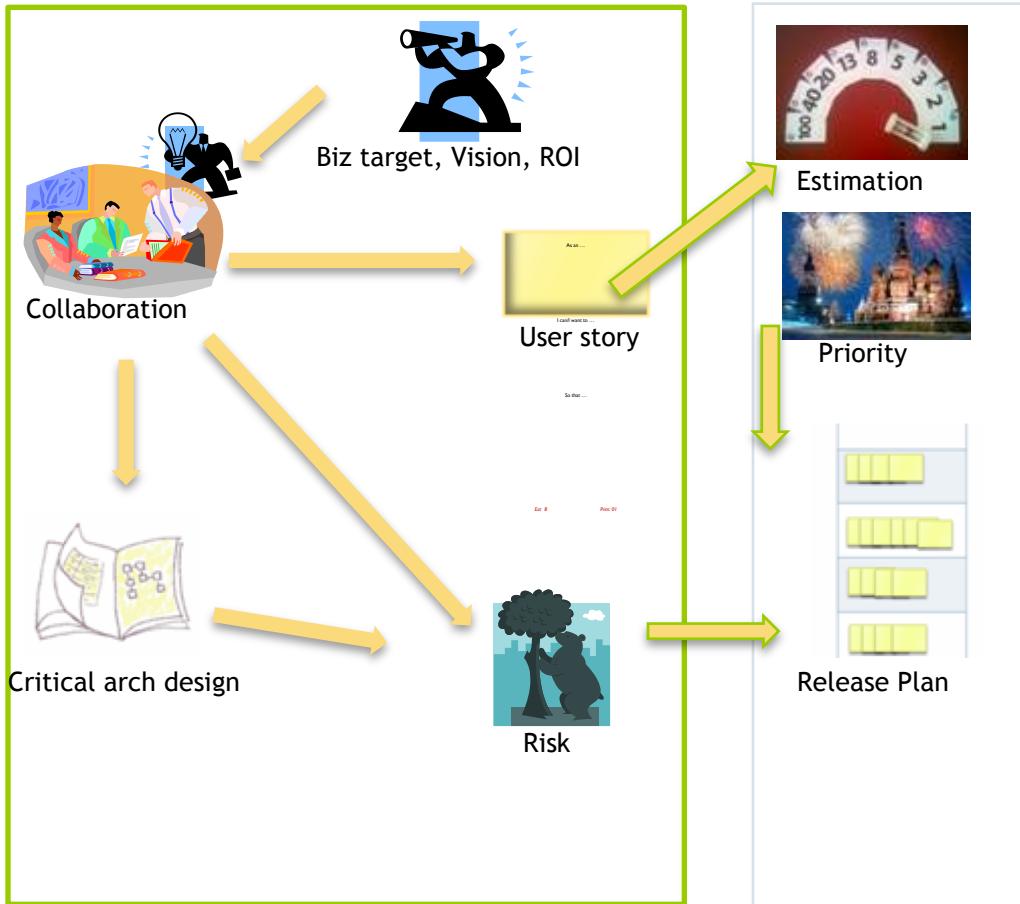
Total Story point = ~240

Velocity = ~40 points/
Sprints

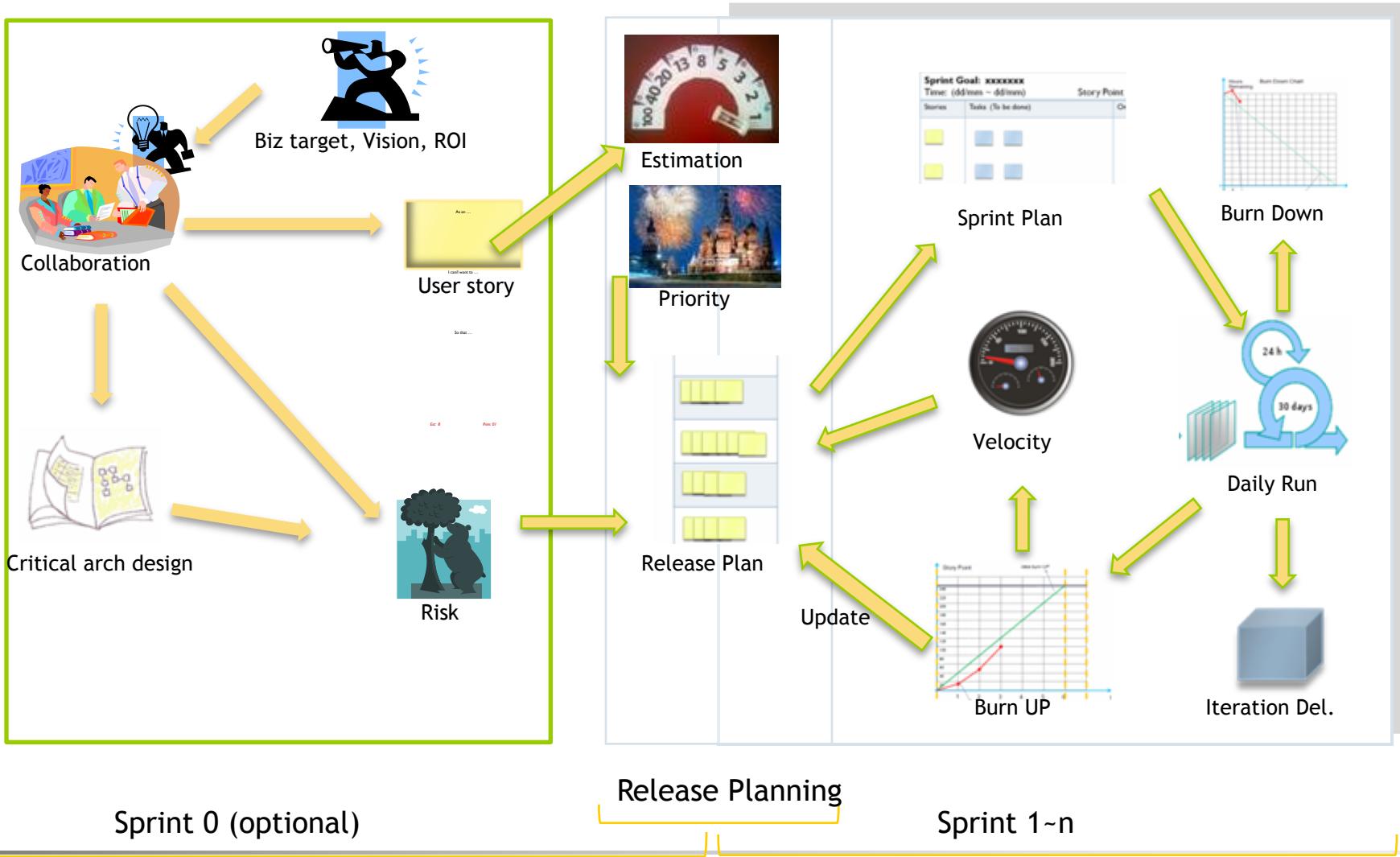
Sprints to get it Done
 $= 240/40 = 6$

Sprint planned = 7 =
6 Dev sprints + 1 buffer
sprint

Sprint 0



Try it All Together



Scrum is Simple

But

Simple does Not Mean Easy

It Needs

- Commitment of the organization
- Commitment of the team
- Highly self-disciplined
- Long time inspection and adaption
- Pursue of tech. excellence to sustain (Agile Engineering Practice)

Engineering Practice

- Test driven Development
- Pair programming
- Refactoring
- Simple Design
- Continuous Integration
- Acceptance Test Driven Development
- Emergent Design
- Agile Modeling
-

4 Engineering Practices

Individual Practices

- Test Driven Development
- Pair Programming
- Simple Design
- Refactoring
- Continuous Integration

Test Driven Development – How It Works

- Write test code first. Write only enough test code to demonstrate a failure.
- Write code to make the failing test pass. Write only enough code to pass the test.
- Clean up the code and tests.

RED, GREEN, REFACTOR

... and with quick feedback loop

Test Driven Development – An Example

The screenshot shows a Java development environment with two code editors and a results window.

Math.java:

```
1 package com.whtr.redqueen;
2
3 public class Math {
4
5     public int add(int a,int b){
6
7         //这里并没有计算加法，返回一个假的数据
8         return 2;
9     }
10    功能代码
11
12 }
13
14 }
```

MathTest.java:

```
1 package redqueenTest;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Assert;
6 import org.junit.Test;
7
8 import com.whtr.redqueen.Math;
9
10 public class MathTest {
11
12     @Test
13     public void test() {
14
15         Math math = new Math();
16
17         int result =math.add(2, 2);
18
19         Assert.assertEquals(4, result);
20
21     }
22
23 }
24
```

JUnit Results:

Finished after 0.019 seconds

Runs: 1/1	Errors: 0	Failures: 1
-----------	-----------	-------------

redqueenTest.MathTest [Runner: JUnit 4] (0.001 s)

test (0.001 s) 运行单元测试效果

Failure Trace

```
java.lang.AssertionError: expected:<4> but was:<2>
at redqueenTest.MathTest.test(MathTest.java:19)
```

<http://blog.csdn.net/u013565163>

Test Driven Development – An Example

The screenshot shows a Java development environment with three main panes:

- Left Pane (Math.java):** Contains the code for a class named `Math`. It has a single method `add` that takes two integers `a` and `b` and returns their sum. A red annotation "编写功能代码" (Write functional code) is placed next to the return statement.
- Middle Pane (MathTest.java):** Contains the test code for `Math`. It imports JUnit assertions and the `Math` class. The `MathTest` class contains a single test method `test` that creates a new `Math` object, calls its `add` method with parameters 2 and 2, and asserts that the result equals 4. A red annotation "输入参数，判断返回的结果结果是否和期望一致" (Input parameters, judge whether the returned result is consistent with the expected result) is placed next to the assertion line.
- Right Pane (JUnit View):** Shows the test results. It indicates that the test was run in 0.015 seconds, with 1 run, 0 errors, and 0 failures. The test name is `redqueenTest.MathTest [Runner: JUnit 4] (0.000 s)`, and the duration is `test (0.000 s)`. A red annotation "返回结果和期望一致，变绿色" (The result is consistent with the expectation, turn green) is placed near the test status bar.

<http://blog.csdn.net/u013565163>

Test Driven Development – Why It Works

- Focus on external behavior
- Full automation test list
- No unused code
- With Pair Programming: Ping-pong work style
- Less test & function leak

Pair Programming – How It Works

- 2 people

Roles during pair: Driver & Navigator

- Environment:

Share 1 high performance work station

1 monitors, 2 keyboards, 2 mouse

Sit comfortably



Pair Programming – Why We Need It

- Continuous Review: fast feedback loop
- Continuous learning: study from people
- Less mistake, higher quality
- Pair programming is a thinking tool



Refactoring – How It Works

- Three strikes and you refactor
- Understand bad smells
- Build test system
- In small steps

Refactoring – How It Works (cont.)

■ Bad smells example

Duplicated Code

Long method

Large class

Long parameter list

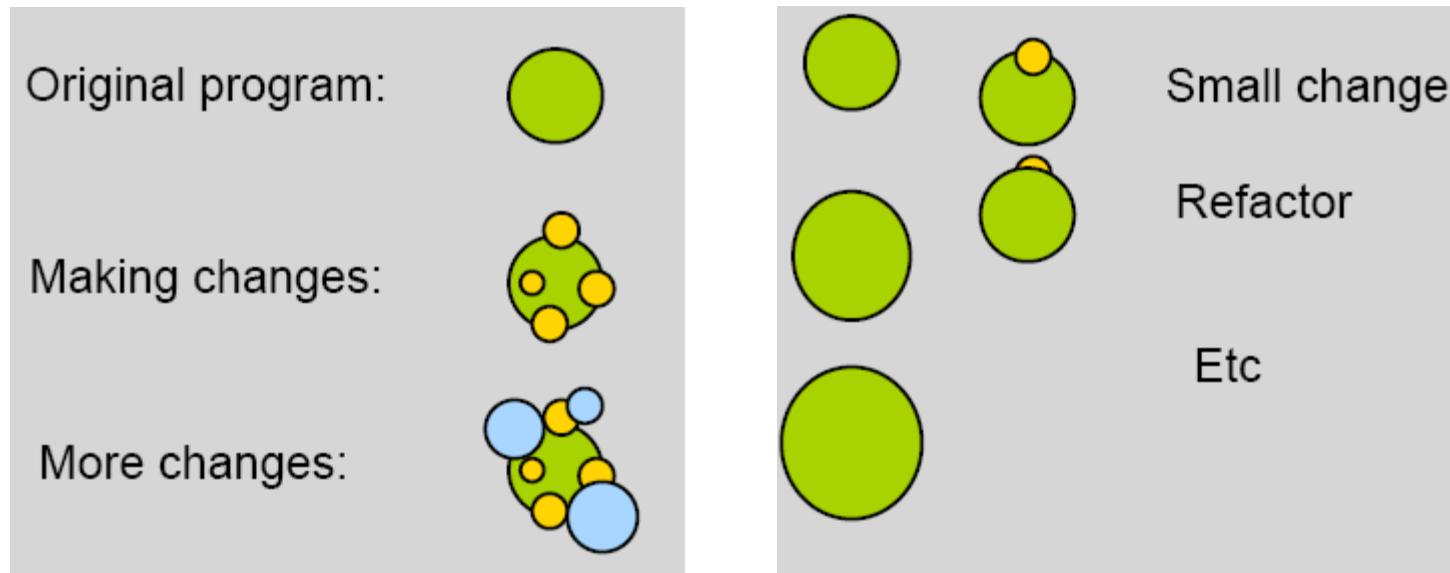
High cyclomatic complexity

...

Refer to 22 bad smells in Refactoring – Improving the design of existing Code

Refactoring – Why We Need It

- Sustainable
- Easy for extension and maintenance
- Fix the broken window



Simple Design – How it Works

■ What's simple

Work

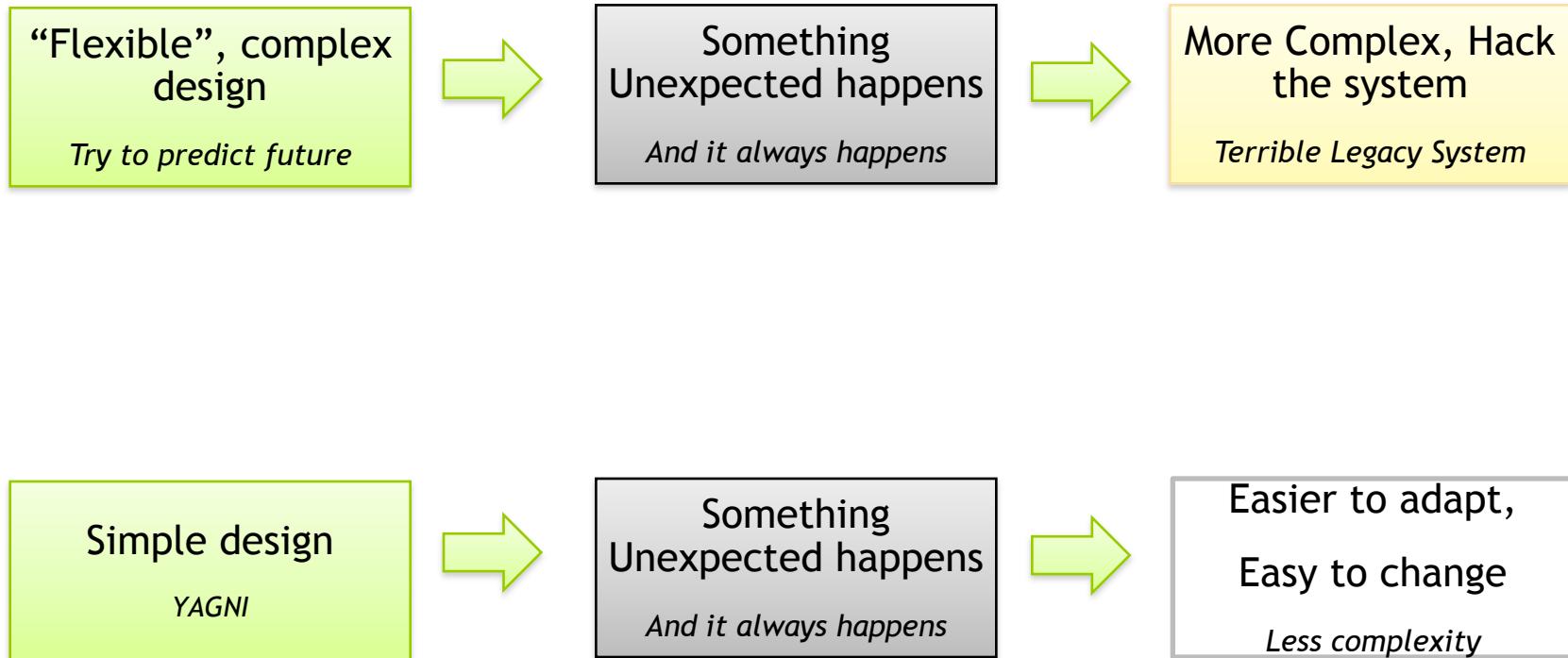
Communicative

No duplication

Minimal class and method

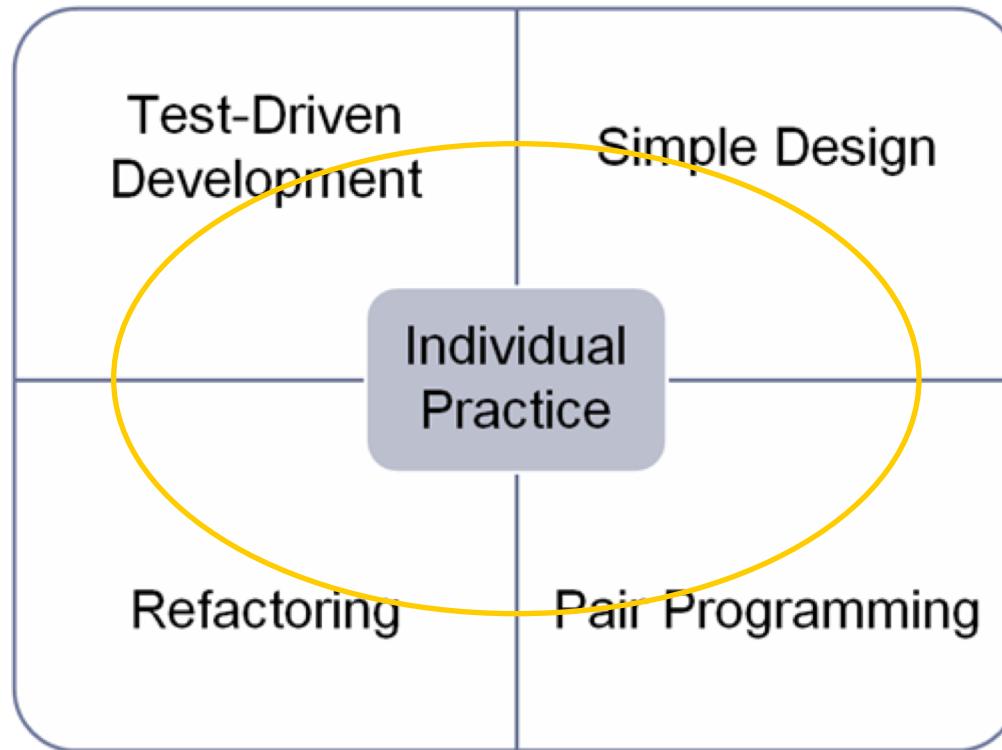
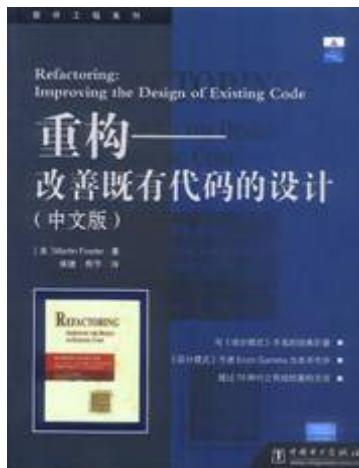
- Kent-Beck's definition

Simple Design – Why We Need It

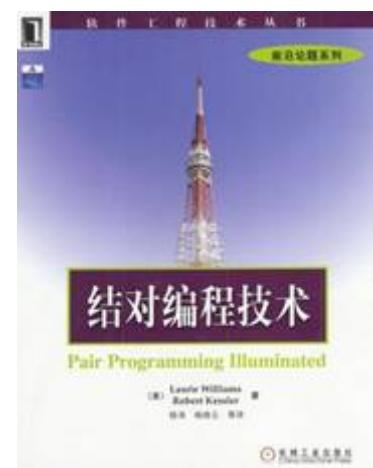
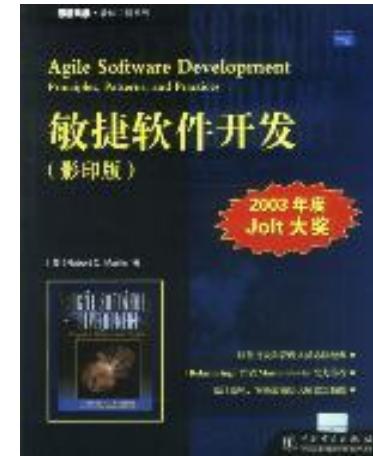


Simple design not simplistic

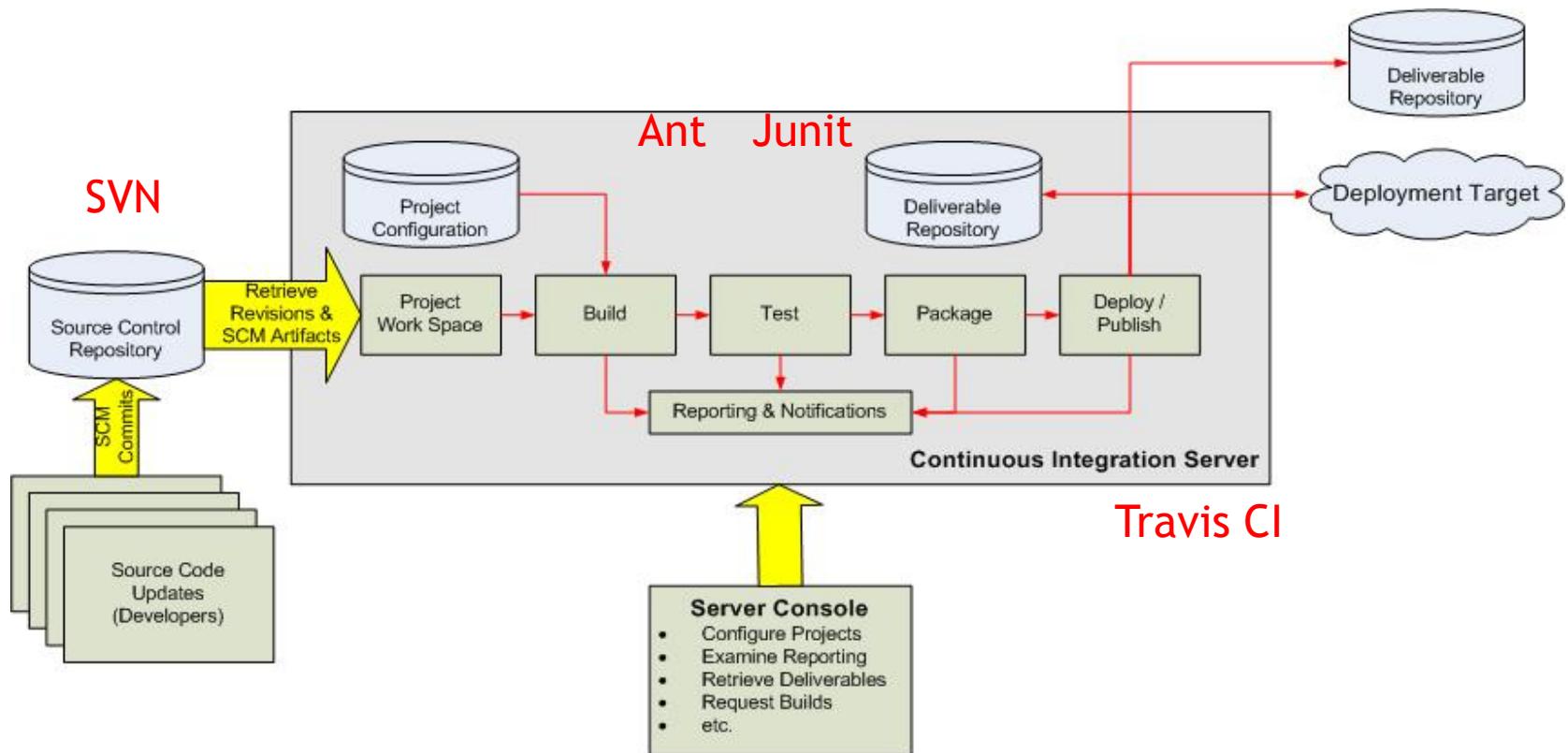
Engineering Practices Support Each Other



The weakness of one is covered by the strengths of others.

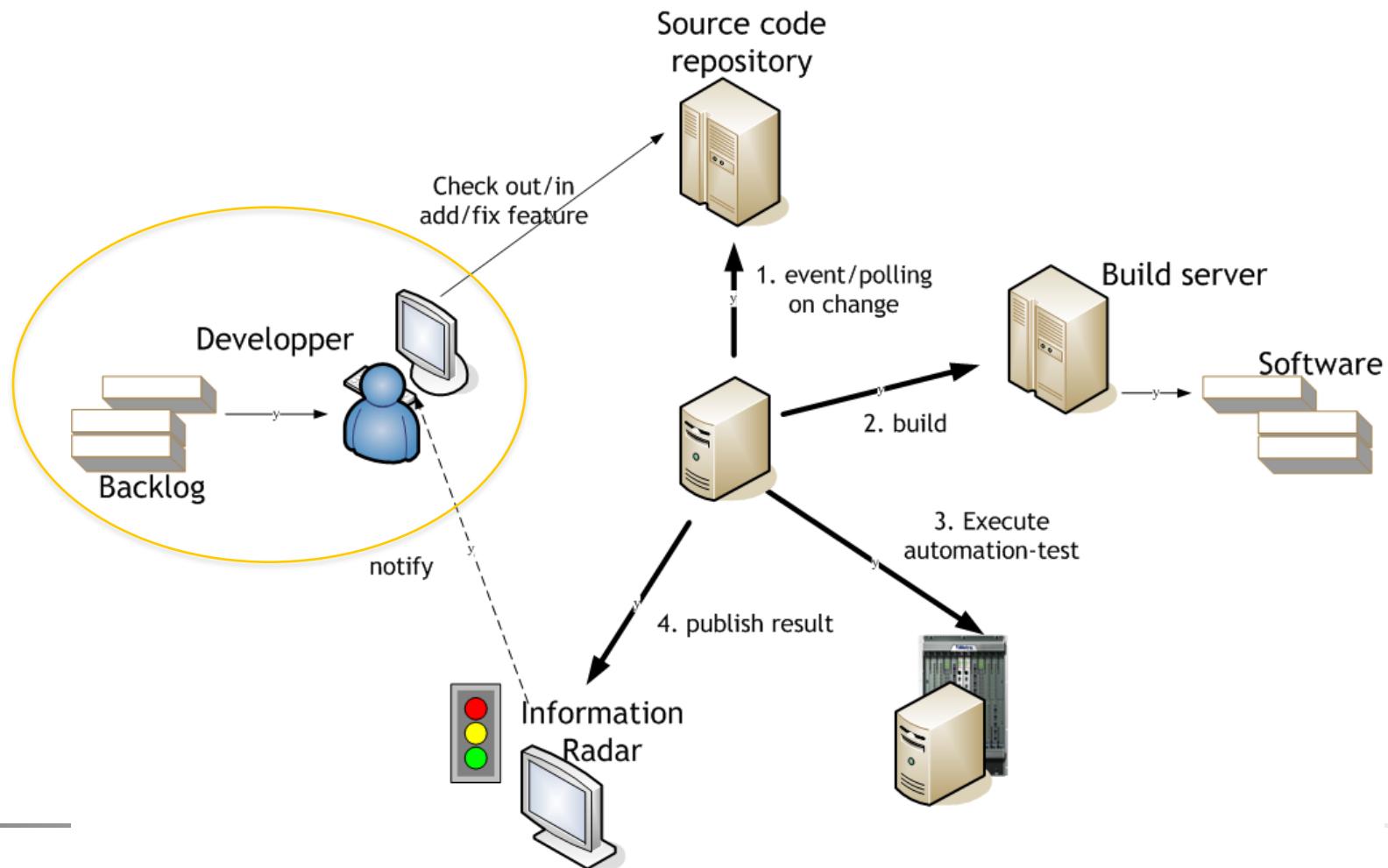


Continuous Integration – How It Works



Feedback Effectively

*verified by an automated build
(including test)*



Continuous Integration – Why We Need It

Continuous Integration is a **developer practice** with the goal to always keep a **working system** by making **small changes**, slowly growing the system and **integrating them at least daily** on the **mainline** typically supported by a **CI system** with lots of **automated tests**

- From the Book “practices for scaling lean & agile development”

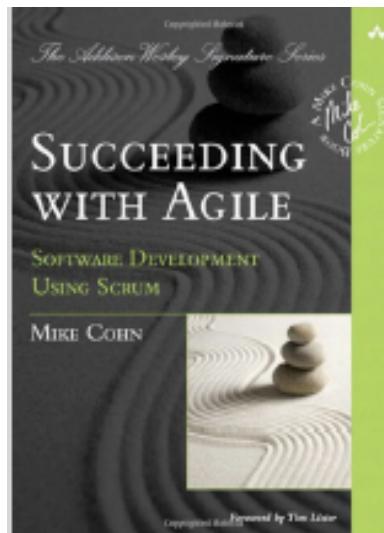
5

Reading

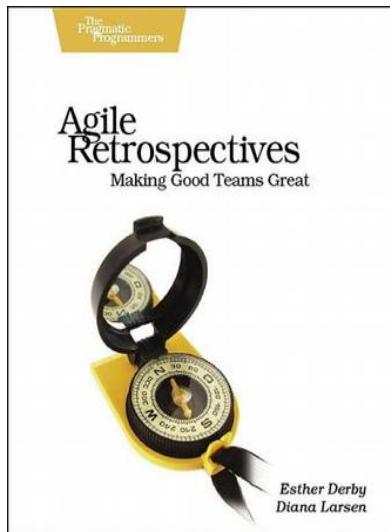
Reading Recommendations - for Start



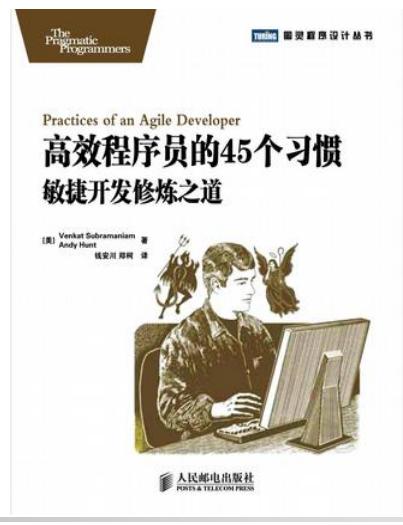
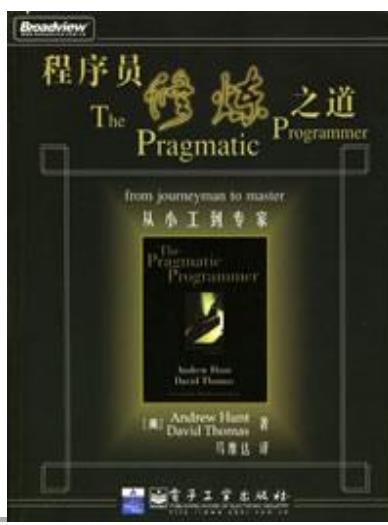
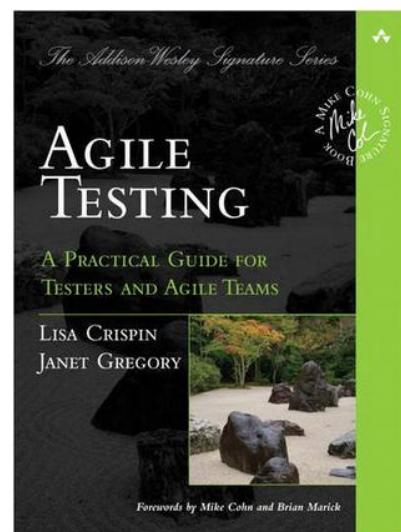
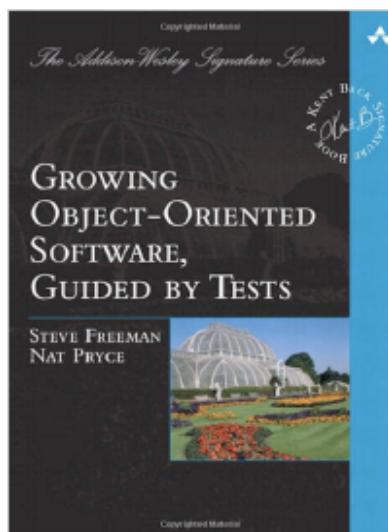
Reading Recommendations - For Scrum Practitioner



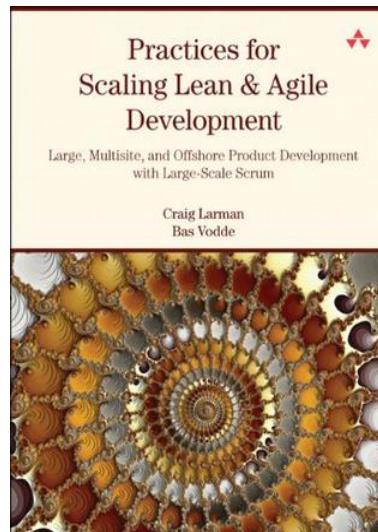
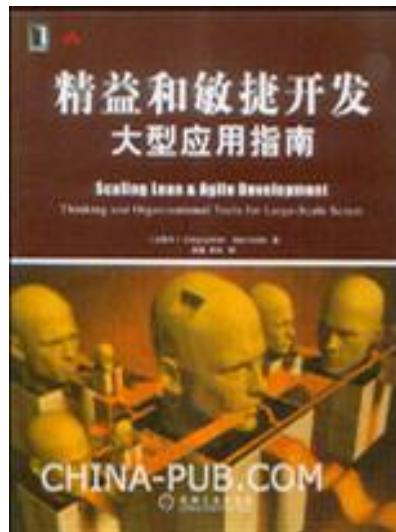
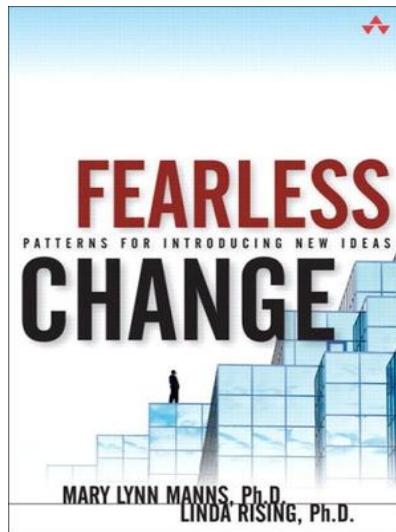
Reading Recommendations - For Agile Coaching



Reading Recommendations - For Programmer



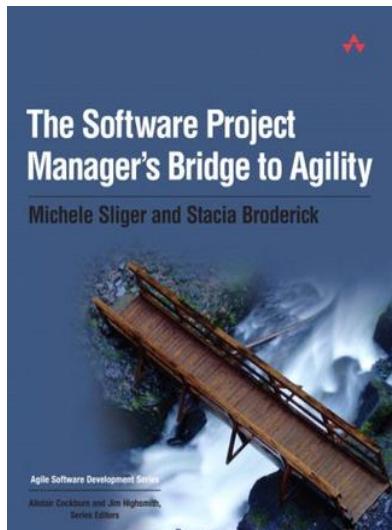
Reading Recommendations - For Organization Transformation



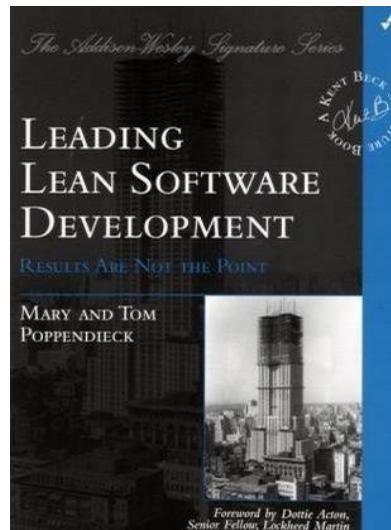
Reading Recommendations - For Project and Line Managers



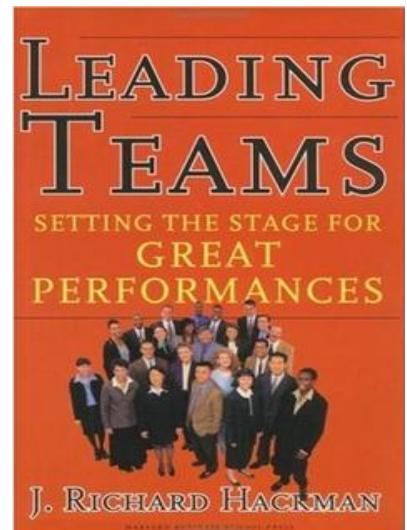
人民邮电出版社
POST & TELECOMPRESS



Agile Software Development Series
Alistair Cockburn and Jim Highsmith,
Series Editors



Foreword by Dottie Action,
Senior Fellow, Lockheed Martin



J. RICHARD HACKMAN

Internal Resource

CPG CTO Be Agile Website:

<https://acos.alcatel-lucent.com/projects/be-agile/>

CPG CTO Be Lean Website:

<https://acos.alcatel-lucent.com/projects/lean/>

CPG China Agile Community of Practice

Website: <https://acos.alcatel-lucent.com/wiki/g/be-agile/China%20Agile%20Community>

Mail List: be-agile-china@acos.alcatel-lucent.com

6

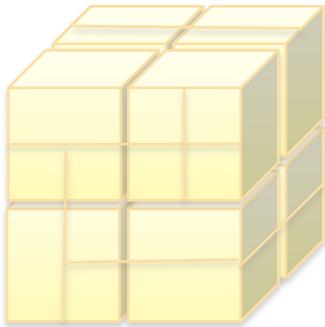
Summary - What Does It Mean for Us



Vision
ROI



Define & Split up the product



Large group spending a long time building a big thing



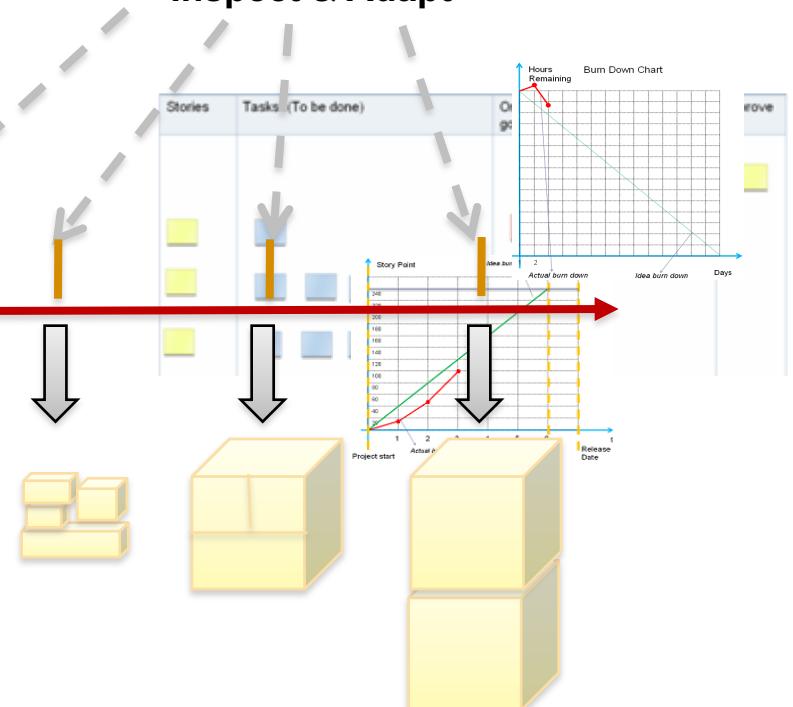
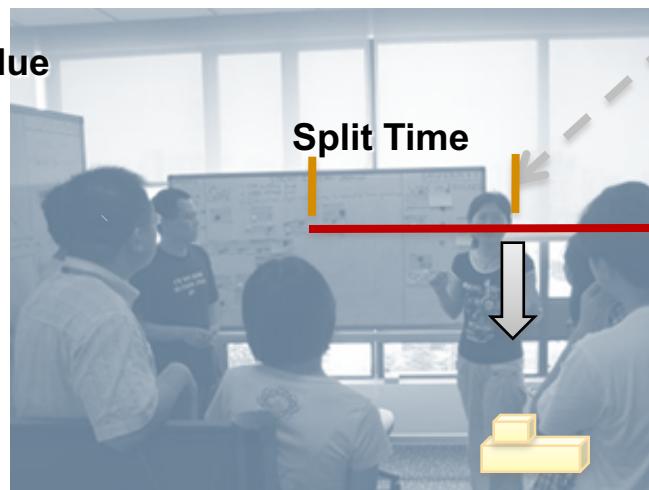
Small teams spending a little time building small things
but doing integration regularly to see the whole

Kai Zen

改善 善

Change Good

Inspect & Adapt



Optimize Business Value

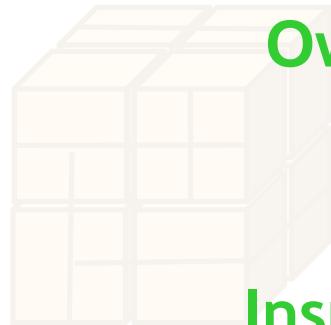




Vision
ROI



Define & scope



Large group spending a long time building a big thing



Small teams spending a little time building small things

Close collaborative between team and Product Owner from Biz is key for customer value fulfillment and Biz Success



Change Good

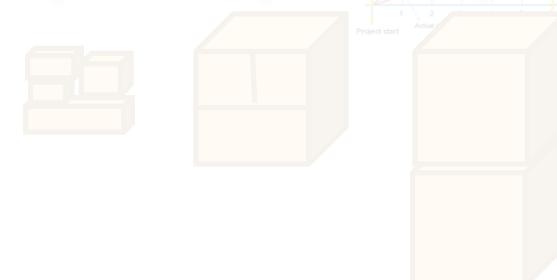
Inspect and adapt is essential to make good team great

Tech. practices and pursuit of tech. excellence to get sustainable efficiency

Optimize Business Value



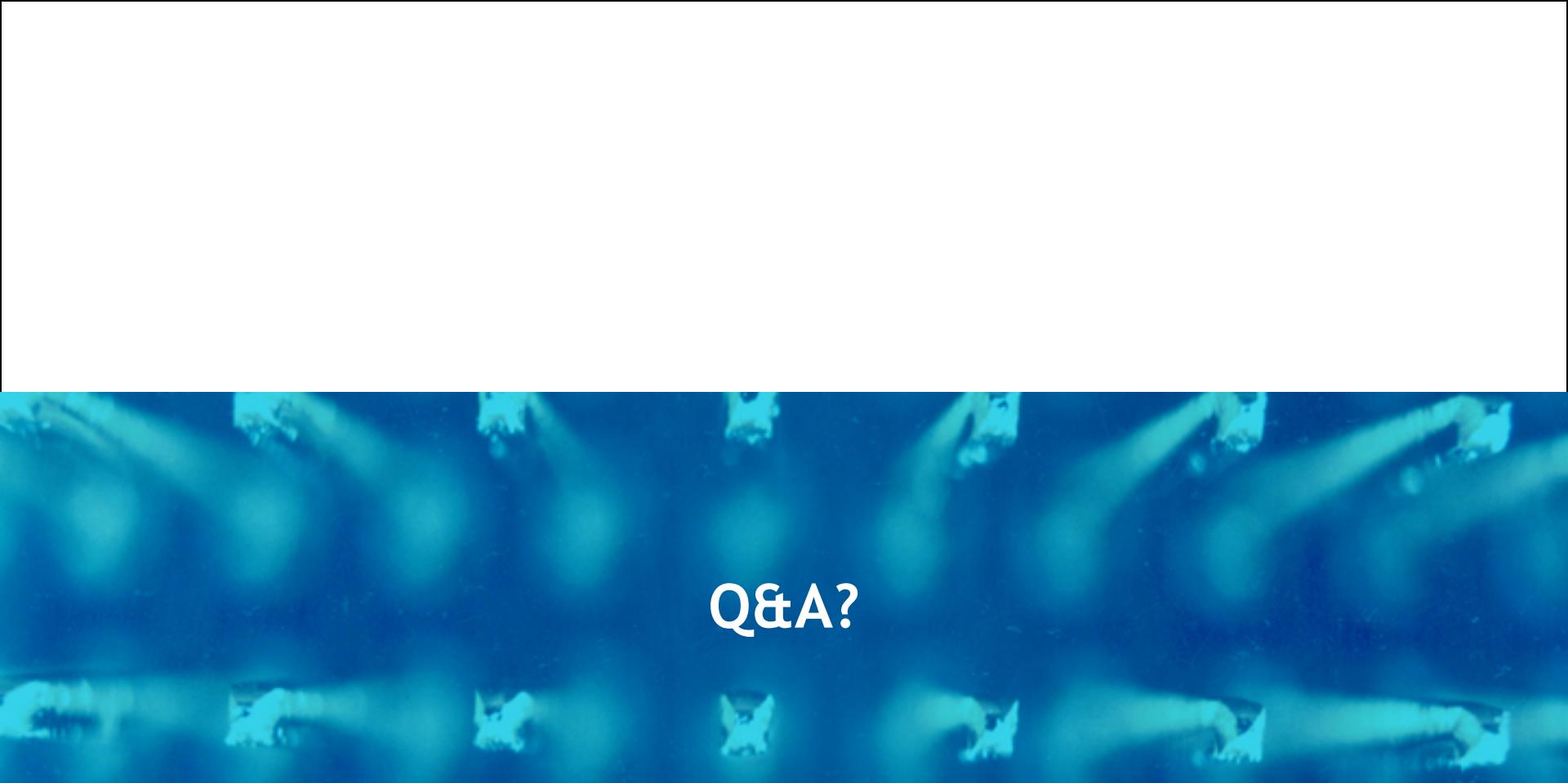
Split Time



Perfection is a Direction not a Place - Henrik Kniberg



Agile is a Tool Not a Goal



Q&A?

