Scrum

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.

In 2008, 84% of all Agile projects used Scrum

Source: December 2008 Global Agile Company Online Survey

Jeff and Ken formulate Scrum

Scrum presented at OOPSLA 1995

PLOP

Agile Manifesto

Agile Alliance

Books

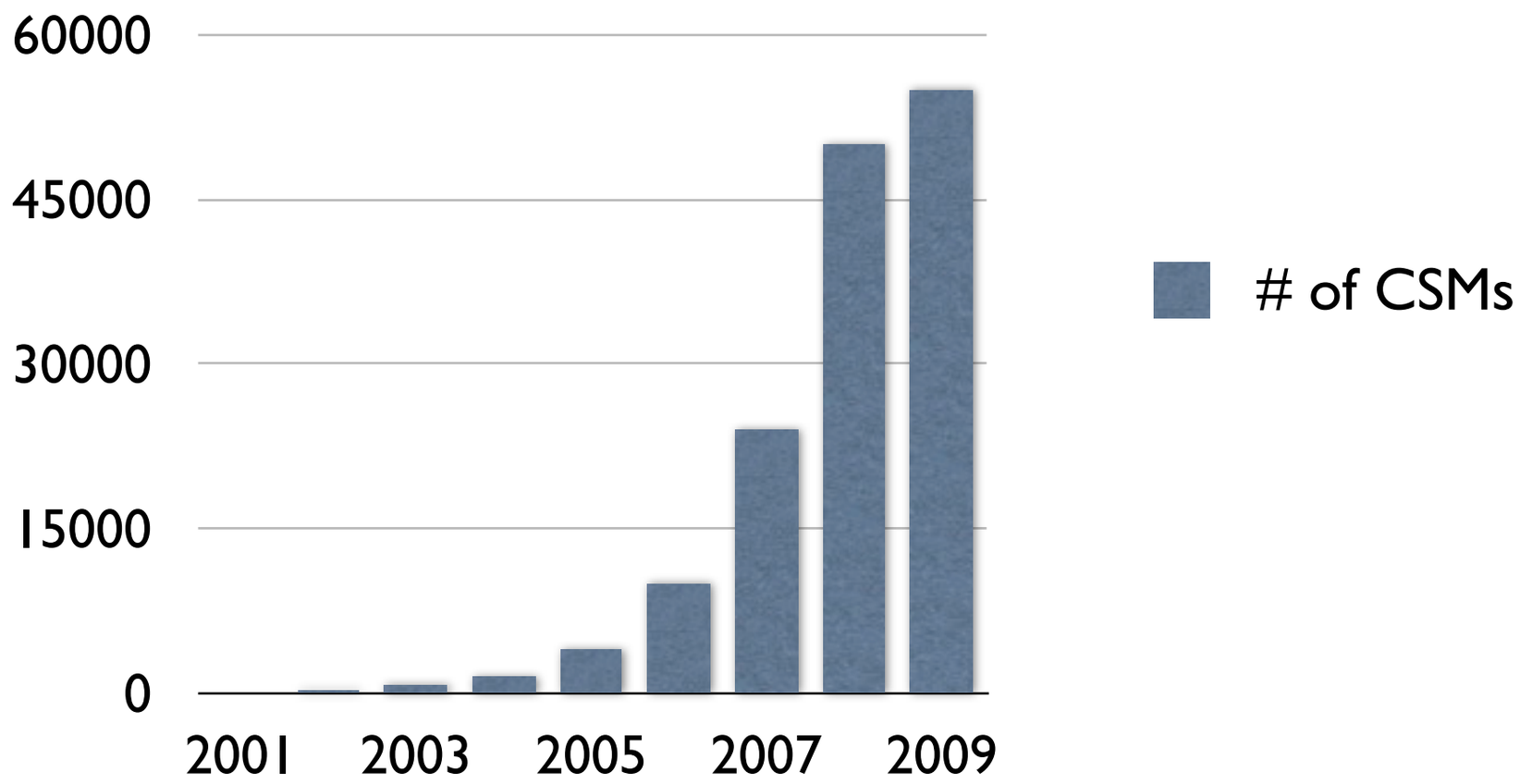
CSM

Scrum Alliance

CST

Gathering Stockholm

1991 1995 1998 2001 2002 2003 2008



Scrum Values

Transparency

Empiricism

Self-Organization

Integrity

Delivery

Great News

Time Boxes being widely used

Waterfall being used less

Agile values are public

Not So Great News (underneath waterfall we found)

Developers have trouble with increments

Customers would rather throw the responsibility over the wall

IT middle-management is resistant

Transparency is avoided

Command and control is prevalent

Scrum

Empirical process for managing the development and deployment of complex products.

Empiricism is dependent on frequent **inspection and adaptation** to reach goal.

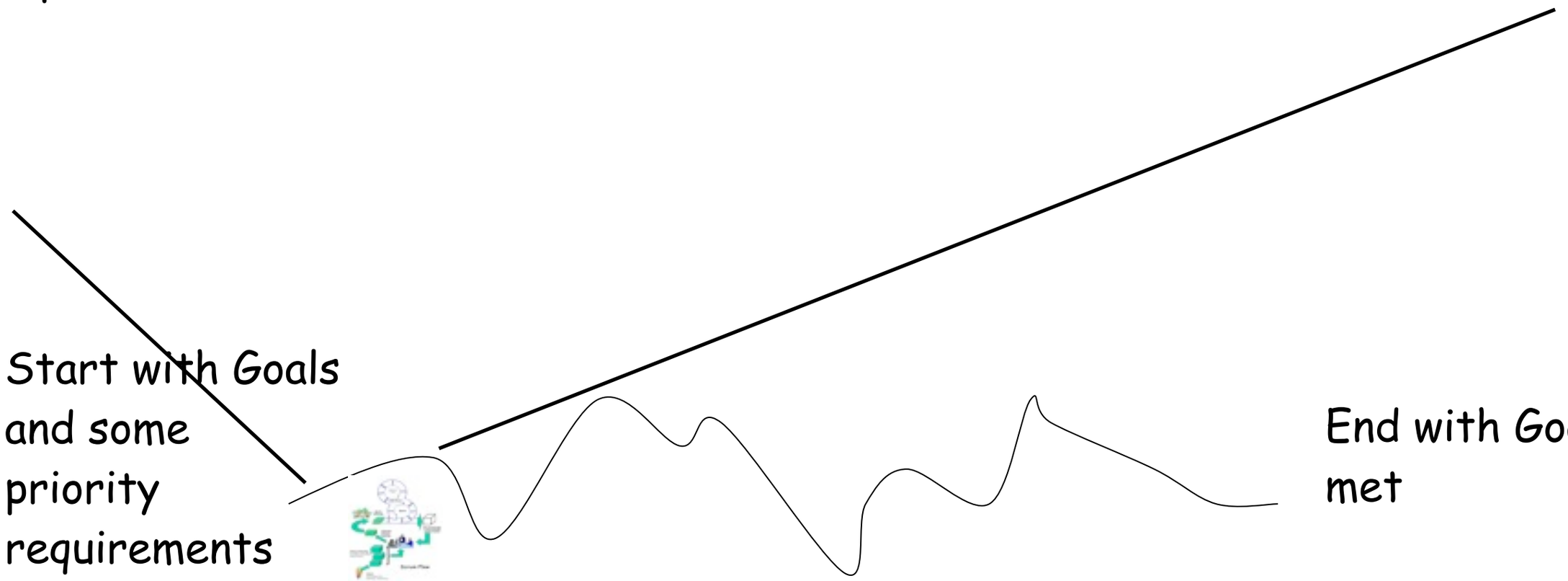
Inspection is dependent on **transparency**.

Scrum rests on the four legs of **iterative** development that generates done **increments** of functionality using **self-managing teams** that are **cross-functional**.

Defined, Predictive

Start with Plan and all requirements

End with all requirements completed

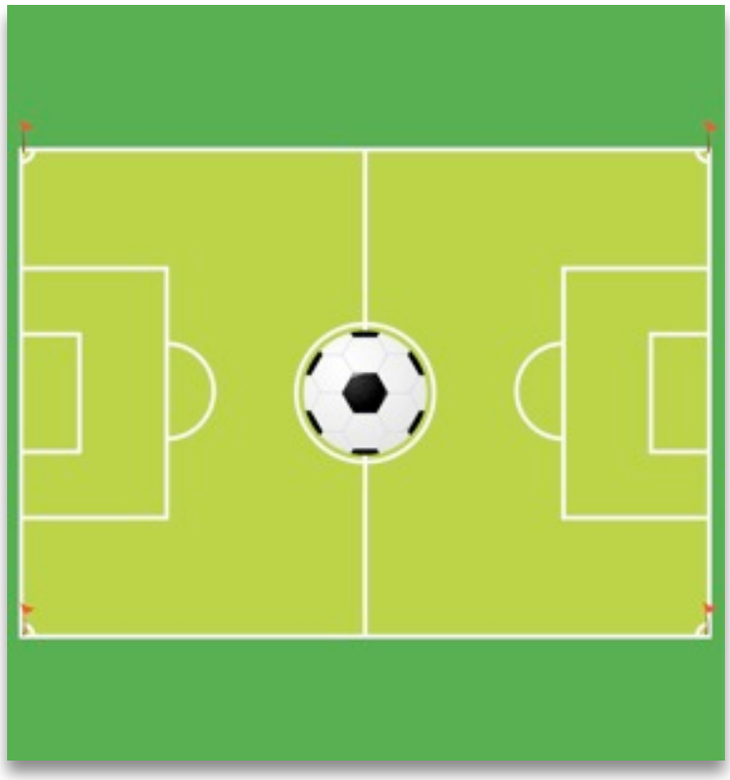


Start with Goals and some priority requirements

End with Goals met

Scrum - Empirical

Scrum is not a methodology that will make you develop better products. Scrum does not provide the answers to how to build quality software faster.



Scrum is a tool you can use to find out what you need to do to build quality software faster.

Scrum

SCRUM GUIDE

This guide explains how to use Scrum to build products. In doing so, it will describe how the framework and its artifacts, time-boxes, roles and rules work together. Scrum does not include techniques and processes for building products; however, it will point out the efficacy and flaws of these techniques and processes.

Scrum is a framework for developing complex products and systems. It is grounded in empirical process control theory*. Scrum employs an iterative, incremental approach to optimize predictability and control risk. Within each iteration, Scrum employs self-organizing, cross-functional Teams to optimize flexibility and productivity.

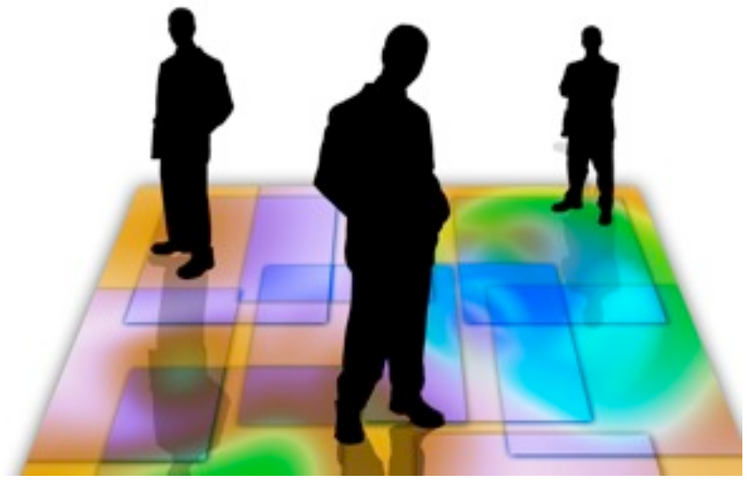
The heart of Scrum is a ***Sprint***. A Sprint is one iteration of a month or less that is of consistent length throughout a development effort. All Sprints use the same Scrum framework, and all Sprints end with an increment of the end product that is potentially releasable. The increment is a complete slice, or piece, of the finished product or system that is developed by the end of an iteration, or Sprint. One Sprint starts immediately after the prior Sprint ends.

http://www.scrumalliance.org/resources/

Scrum Does Not Require Team Collocation

However, with Scrum, you can measure the productivity of collocation

Self-Organizing
20 PBI/100k Euro



Not Self-Organizing
5 PBI/100k Euro

ScrumBut

"There isn't a single football game that we would have lost if we had not run out of time."

Vince Lombardi, coach, GreenBay Packers

Scrum is a tool that you can use to:

- Increase productivity
- Increase predictability
- Increase risk management capabilities
- Increase the value of products and systems
- Increase quality
- Improving the morale and pleasure of the developers, product managers, customers and stakeholders

Scrum's use in spreading in product companies and engineering organizations.

Many of them suffer from one or more of the following:

- Release schedules slipping
- Stabilization at end of release taking longer and longer
- Releases are taking longer and longer
- Planning seems to take too long
- Changes are hard to introduce mid-release
- Quality is deteriorating
- Death marches are hurting morale

However,

When they use Scrum, they run into ScrumButs.

ScrumButs are reasons why they can't take full advantage of Scrum to solve the problems and realize the benefits.

A ScrumBut has a particular syntax:

(ScrumBut) (Reason)(Workaround)

An example ScrumBut is:

"We use Scrum, but (The Daily Scrum meetings are too much overhead) (so we only have them once a week, unless we need them more often)"

ScrumBut Sources

- Business doesn't want to be involved;
- Everyone wants their stuff first and can't agree;
- **Teams don't know how to self-organize;**
- People aren't available to work on teams full-time;
- **Teams don't see a need for a daily Scrum;**
- Teams can't get a piece of functionality done in one Sprint;
- Teams don't have the skills to "do" something
- **Teams can't fit regression and integration testing into the same Sprint as development;**
- **ScrumMaster tells the team what to do and how to do it;**
- Other managers, including functional managers, can't stay out of a Sprint;
- Important things come up that require interrupting the Sprint;
- The Sprints can't start until all of the other groups do their up-front work (usability, architecture, database, etc.);
- Other groups are building hardware or using waterfall.

ScrumBut 1

‹We use Scrum, but›

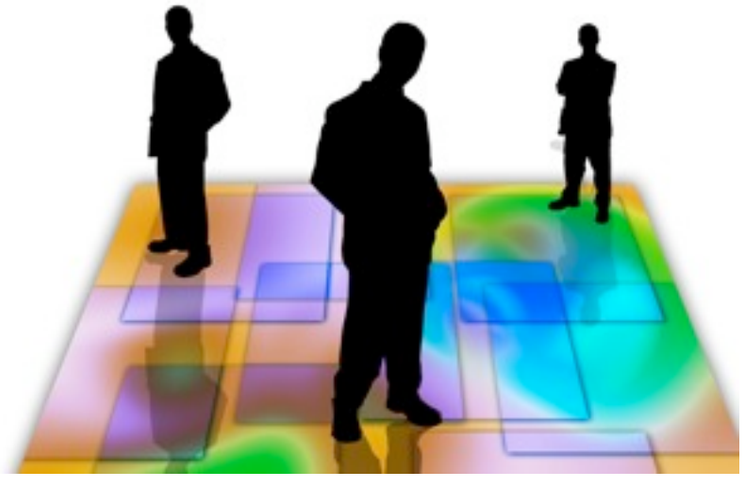
‹we don't need to meet every day for a status meeting›

‹so we only get together when we need to share something.›

Dysfunction: The team is not self-organizing

Detection: Ask the team to demonstrate how it intends to deliver what it committed to for the Sprint

Self-Organizing
20 PBI/100k Euro



Not Self-Organizing
5 PBI/100k Euro

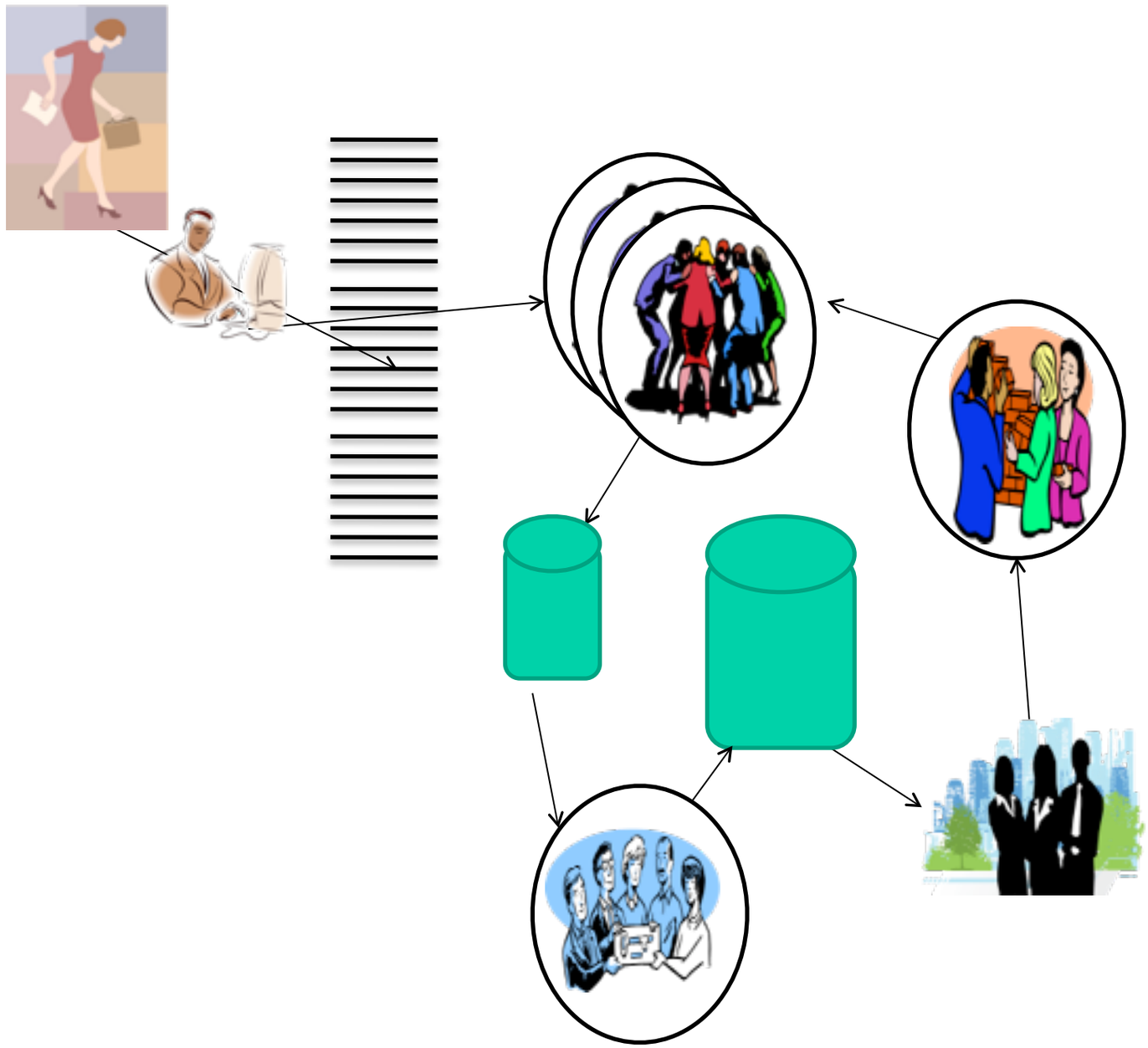
Exercise – Command and Control

1. Form pairs.
2. Assign one person the boss, the other is the worker.
3. The boss can give the following commands – Go, Stop, Right, Left, Faster, Slower
4. The worker must follow the boss's commands.
5. The boss is responsible for having the worker proceed 60 normal paces within two minutes, from the time "go" is said, until "stop" is said, by the moderator.
6. The boss can command the worker but not touch the worker.

Exercise – Self-organization

1. With the same teams as before, except everyone is a worker and responsible for figuring out how to proceed during the exercise by him or herself.
2. Each team proceeds 60 normal paces within two minutes, from the time "go" is said, until "stop" is said, by the moderator.

Small Modifications



We've made some modification to Scrum to fit our circumstances. Do you have any suggestions?

ScrumBut 2

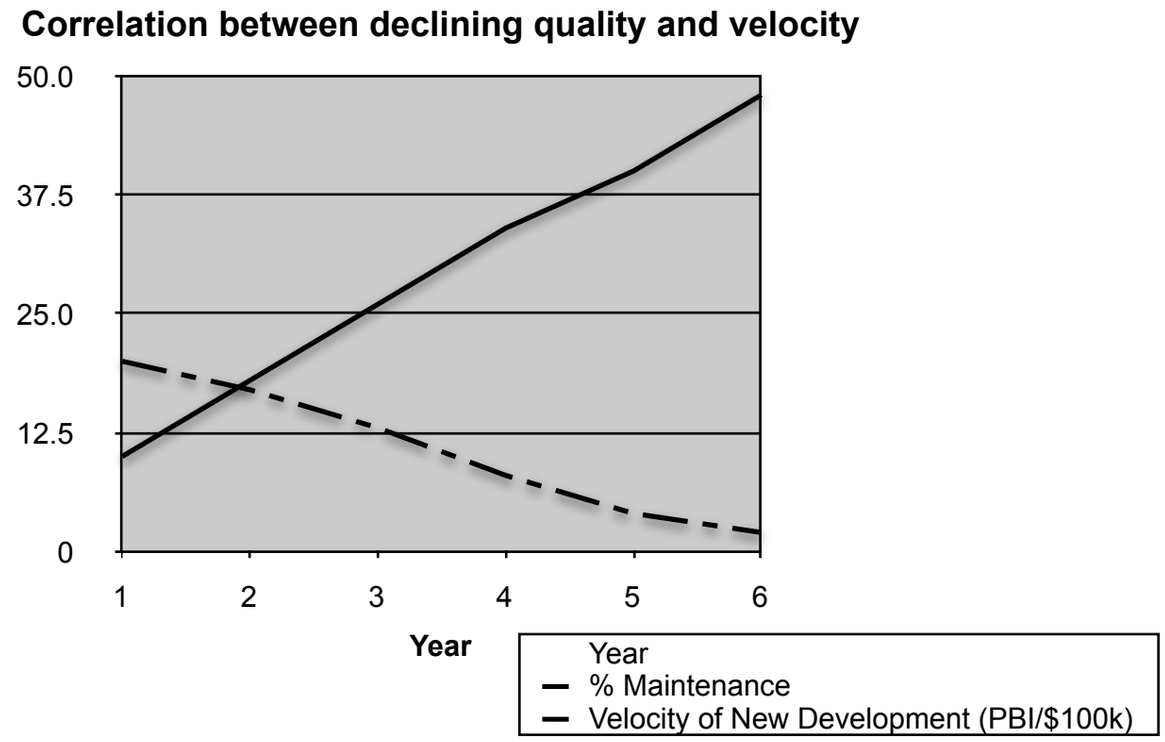
‹We use Scrum, but›

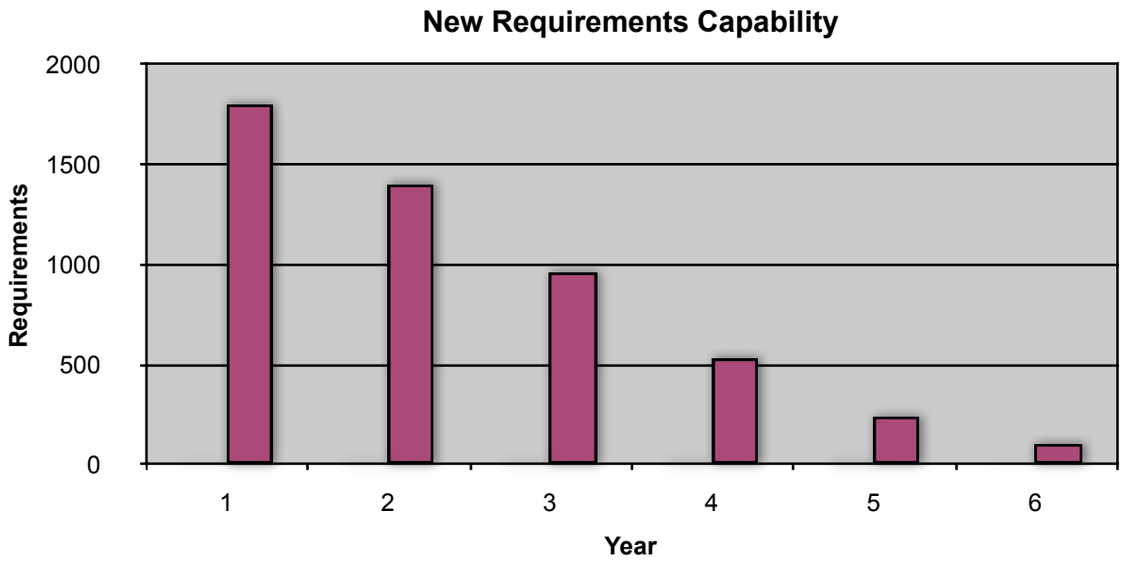
‹we can't get everything done in the development Sprint›

‹so have several develolpment Sprints and then we pull it together with a stabilization Sprint.›

Dysfunction: Transparency is lost, the team doesn't meet its commitments, and undone work accumulates

Detection: Ask the Scrum team what the definition of "done" is.





Dysfunction - perhaps only 25% of all developers are capable of creating a potentially shippable increment of functionality within a Sprint.

They don't have:

- the skills for acceptance test driven development, refactoring, test driven development.
- the followed practices of coding standards, coding reviews, and design reviews;
- the tooling for continuous builds with automated test harnesses for unit, regression, and performance testing.

Work Item	Usual	Rec. start	Done
Requirements analysis	25	25	25
Design of architectural components (UI, System, Data	15	15	15
Design review	0	5	5
Design of tests (system, user acceptance, integration)	0	10	10
Design review	0	3	3
Design of documentation	0	2	2
Design Review	0	1	1
Refactoring of existing design	0	0	8
Design of unit tests for new code	0	3	3
Design of unit tests for code to be refactored	0	3	3
Writing new code	10	7	7
Writing refactored code	6	3	3
Code review (or pair programming)	0	4	4
Write functional tests	8	4	4
Write integration tests	0	4	4
Write documentation	4	4	4
Unit test code	0	2	2
Identify and rectify defects	0	2	2
Subsystem/team build	6	2	2
Identify and rectify defects	1	1	1
Unit test for subsystem/team code	0	2	2
Identify and rectify defects	0	2	2
System/integration build	1	1	1
Identify and rectify defects	0	2	2
System, functional tests	1	2	2
Identify and rectify defects	1	2	4
Integration tests	0	0	2
Identify and rectify defects	0	0	5
Performance tests	0	0	1
Identify and rectify defects	0	0	2
Security tests	0	0	1
Identify and rectify defects	0	0	2
Regression test	0	2	2
Identify and rectify defects	0	8	8
Documentation test	0	1	2
Identify and rectify defects	0	1	1
Total work expended requirement	78	118	148
Work remaining per requirement	65	30	0

The ScrumAlliance is starting a Certified Scrum Developer (CSD) program.

Teams that are able to select product backlog item(s) and turn them into an increment of potentially shippable product functionality will be certified.

CSD program in Munich at the October, 2009 ScrumGathering: Teams will be taught how to build an increment of potentially shippable product functionality within the Scrum framework using Microsoft .NET technology, VSTS and the Scrum Conchango/EMC template.


© 2002 Universal Press Syndicate
4-12
www.ucomics.com
SCRUM!!
GRRRR! GRRRR! GRRRR!!

Questions?