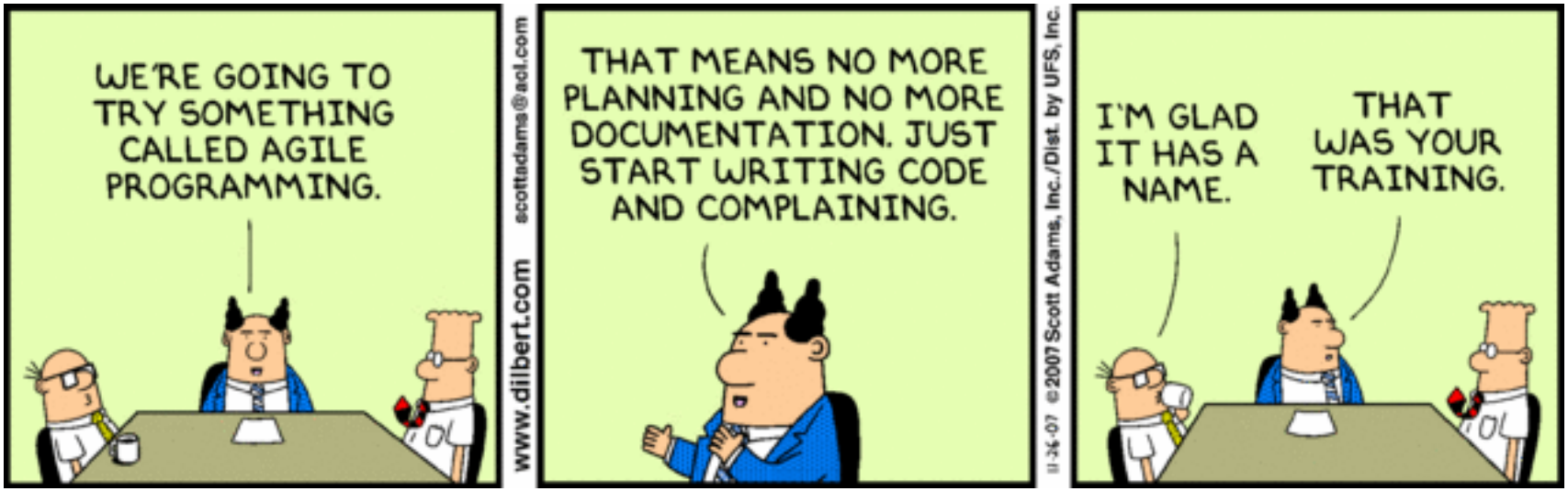
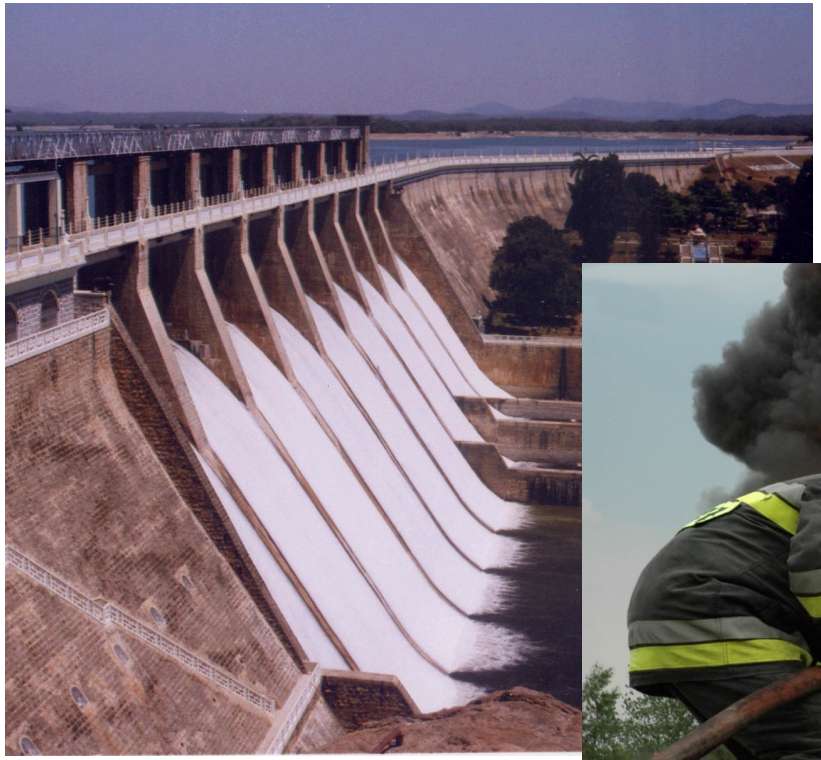
Agile Scrum immersion

Introduction to Agile

What is Agile?



Two Process Extremes



4 Agile core Values

- 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.

Individuals and Interactions
Over Processes and Tools

Working Software

Over Comprehensive Documentation

Customer collaboration

Over Contract Negotiation

Responding to Change

Over Following a Plan

4 Agile core Values

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.

http://agilemanifesto.org/

12 Agile Principles

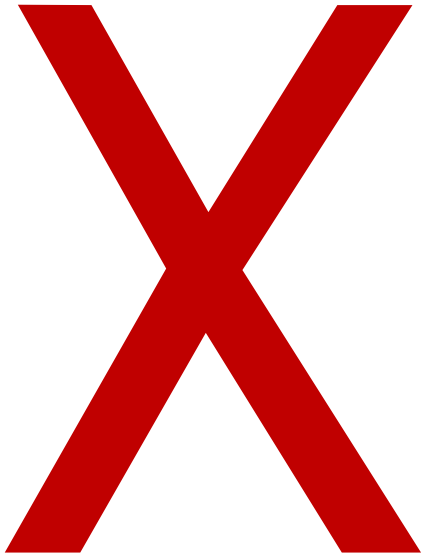
1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The more efficient and effective method of conveying information to and within a development team is face to face conversation.

12 Agile Principles

7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.
9. 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

Agile is not…

- A process
- A one-stop solution
- An excuse to skip:
  - Planning
  - Documentation
  - Design

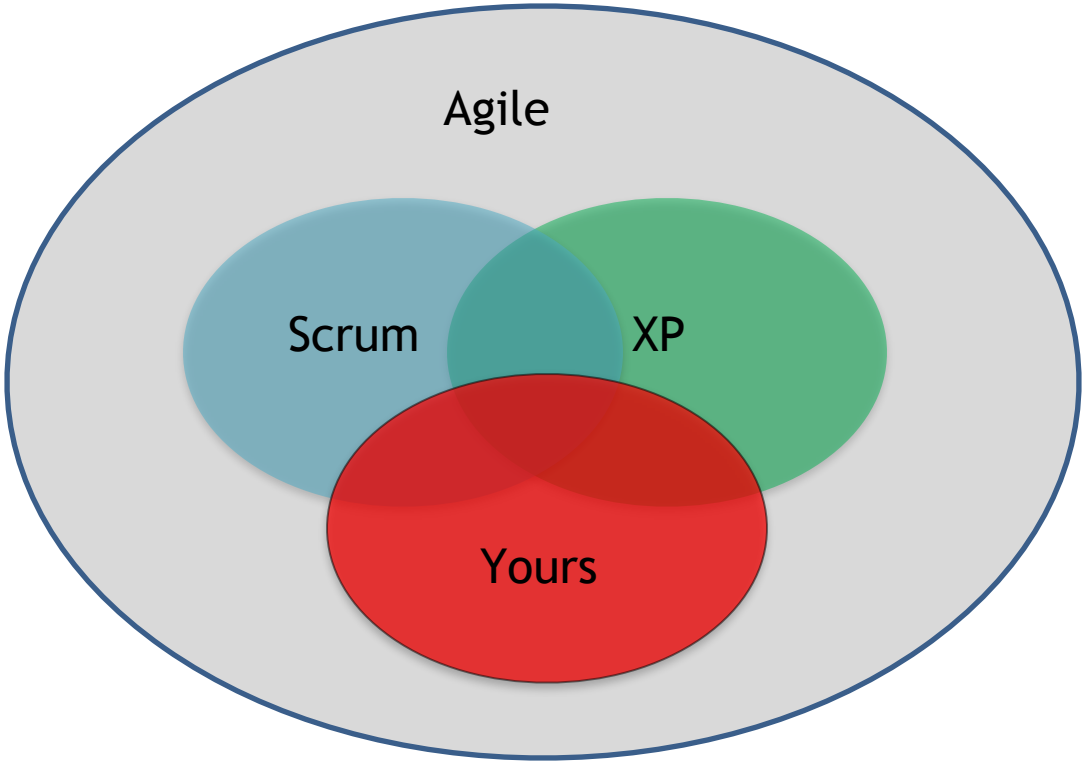


Agile is…

- A set of values and principles
- Too abstract

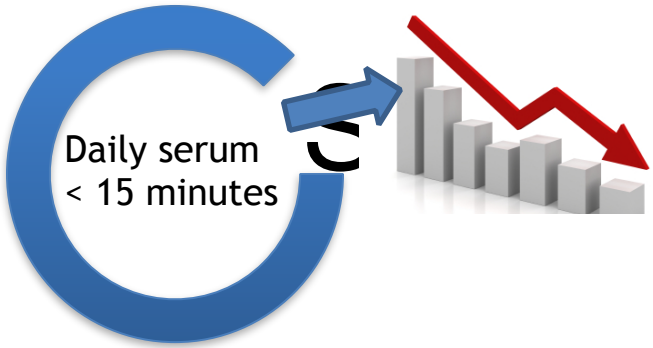
- So how do you implement abstract values and principles?

Agile Implementations



- Business Case
- Financing
- Scope & Approach
- Contracts
- Initial Release Plan
- Assemble Team

Product Owner
Establishes vision and
Prioritizes Product Backlog



Burn Down

Sprint
1 to 4 weeks

Demo
½ Day

Sprint retrospective
Half Day

Sprint Planning
1 day
- Acceptance Defined
- Team commits
- Tasks created

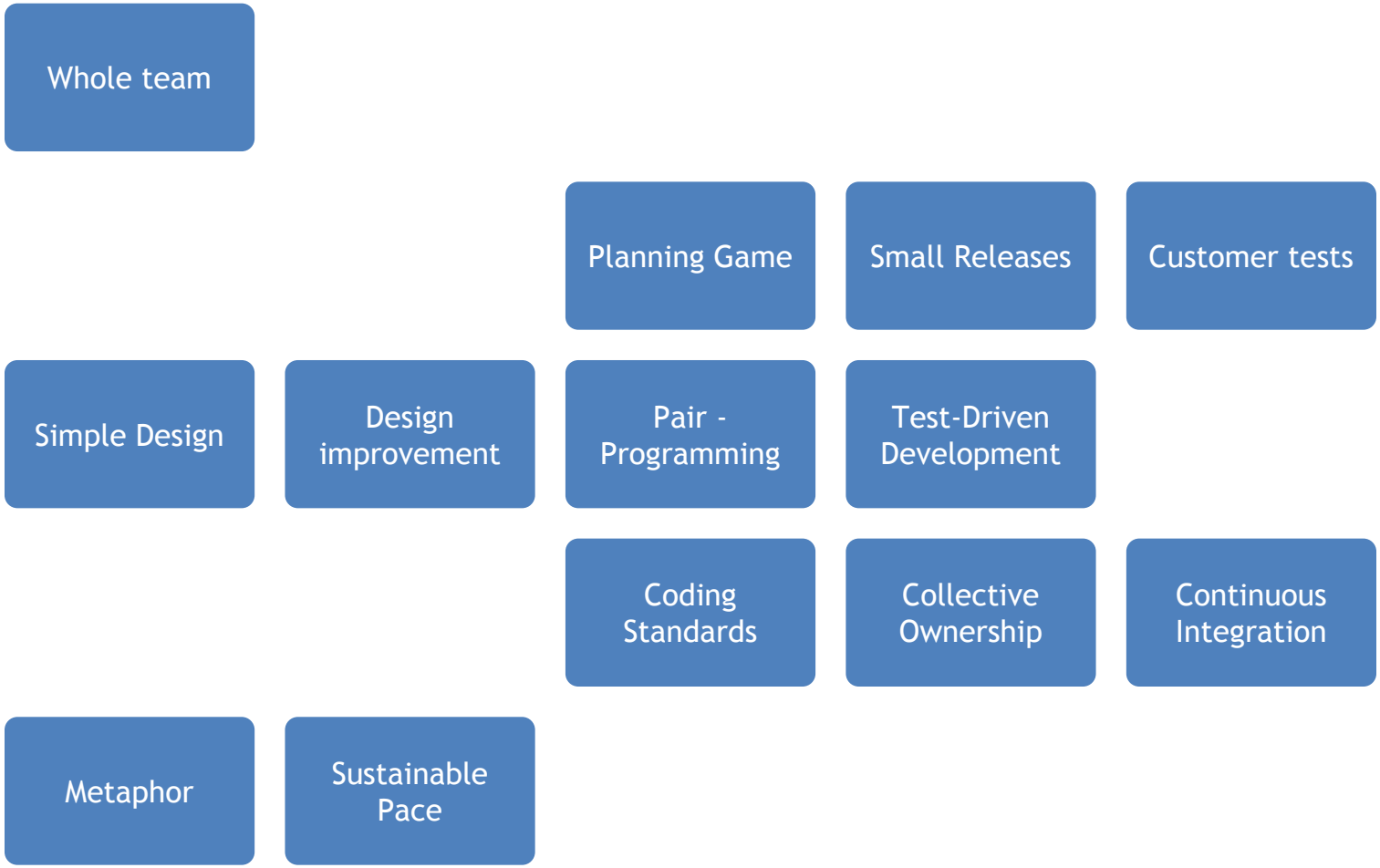
Team (BA, QA, Dev, etc.) creates and estimates Sprint Backlog (tasks)

Demonstrable Release

Velocity

Burn up

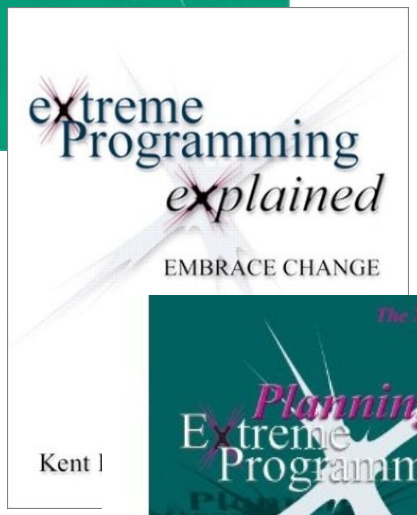
eXtreme Programming

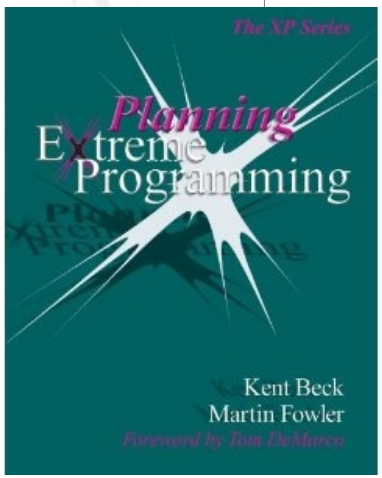


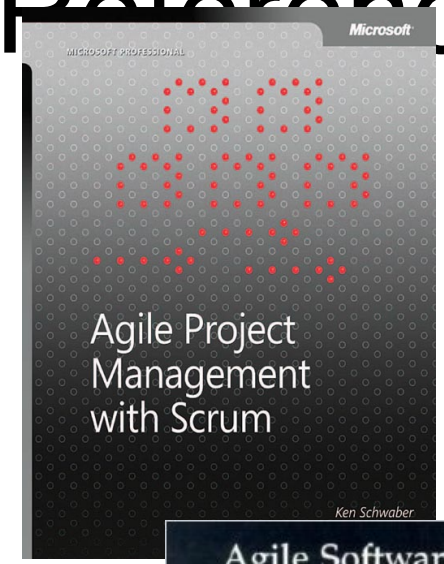
So, agile development teams take responsibility for creating the highest value increment of the highest quality software at a sustainable pace.

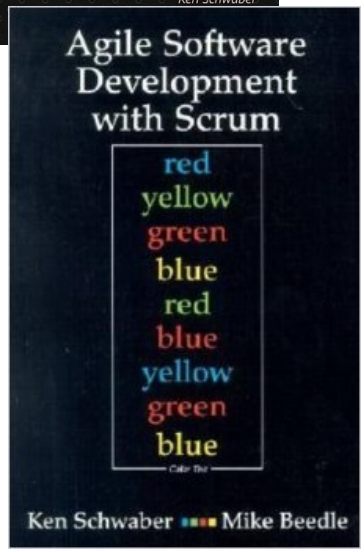
References

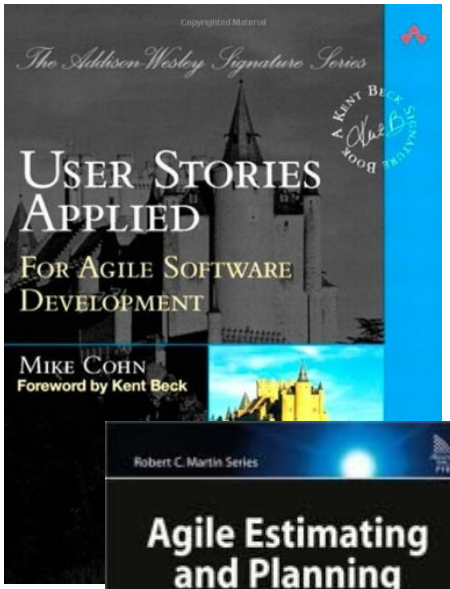

The XP Series
Testing
Extreme
Programming

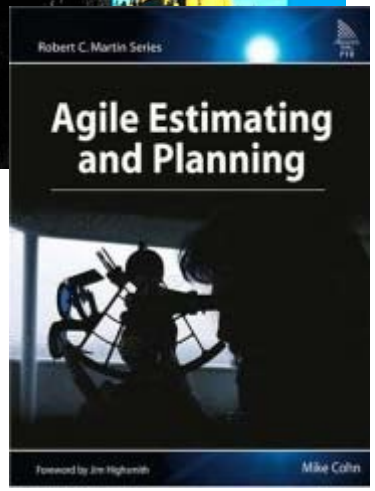

extreme
Programming
explained
EMBRACE CHANGE
Kent


The XP Series
Planning
Extreme
Programming
Kent Beck
Martin Fowler
Foreword by Tom DeMarco


Microsoft
MICROSOFT PROFESSIONAL
Agile Project
Management
with Scrum
Ken Schwaber


Agile Software
Development
with Scrum
red
yellow
green
blue
red
blue
yellow
green
blue
Ken Schwaber
Mike Beedle

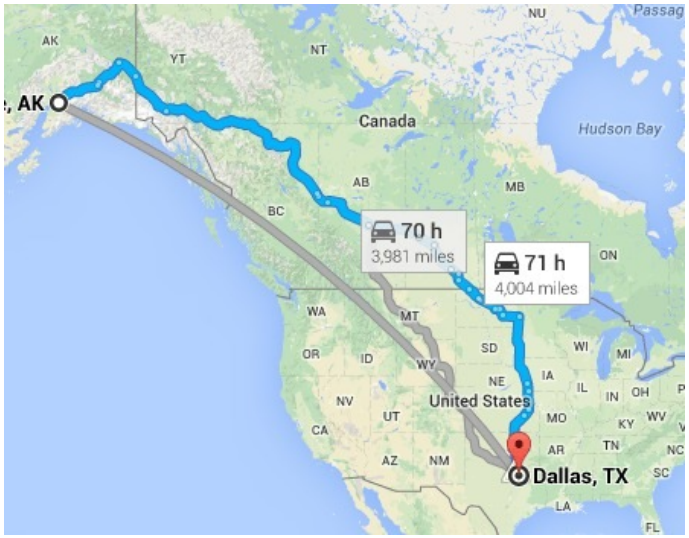

The Addison Wesley Signature Series
A KENT BECK SIGNATURE BOOK
User Stories
Applied
For Agile Software
Development
Mike Cohn
Foreword by Kent Beck


Robert C. Martin Series
Agile Estimating
and Planning
Mike Cohn

Exercise

Alaskan Road trip

- Starting in Dallas, your team needs to be in Anchorage Alaska ASAP
  - 4000 miles distance
  - You must drive
- Plan your journey
  - Create a Timeline
  - List what you will need for the trip
  - List the steps you will take to get there



Agile Scrum Immersion

Introduction to Scrum

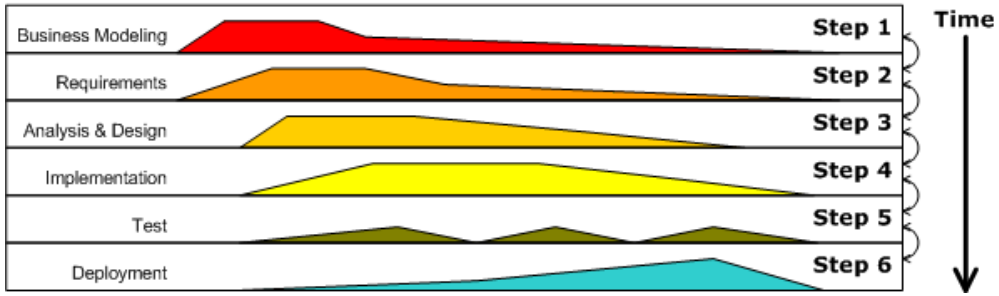
Parkinson's Law

Work expands to
Fill the time
Available for its
Completion.

Waterfall vs. Iterative

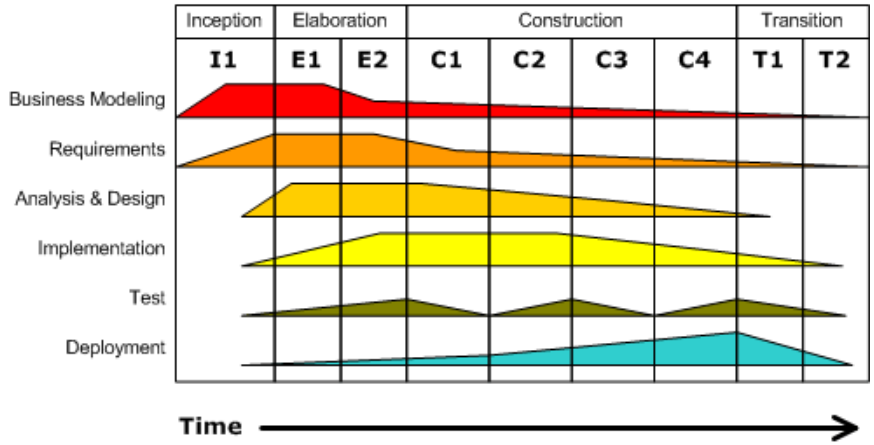
Waterfall Development

Business value is delivered at the end of the project after
A completing a series of single-discipline steps.

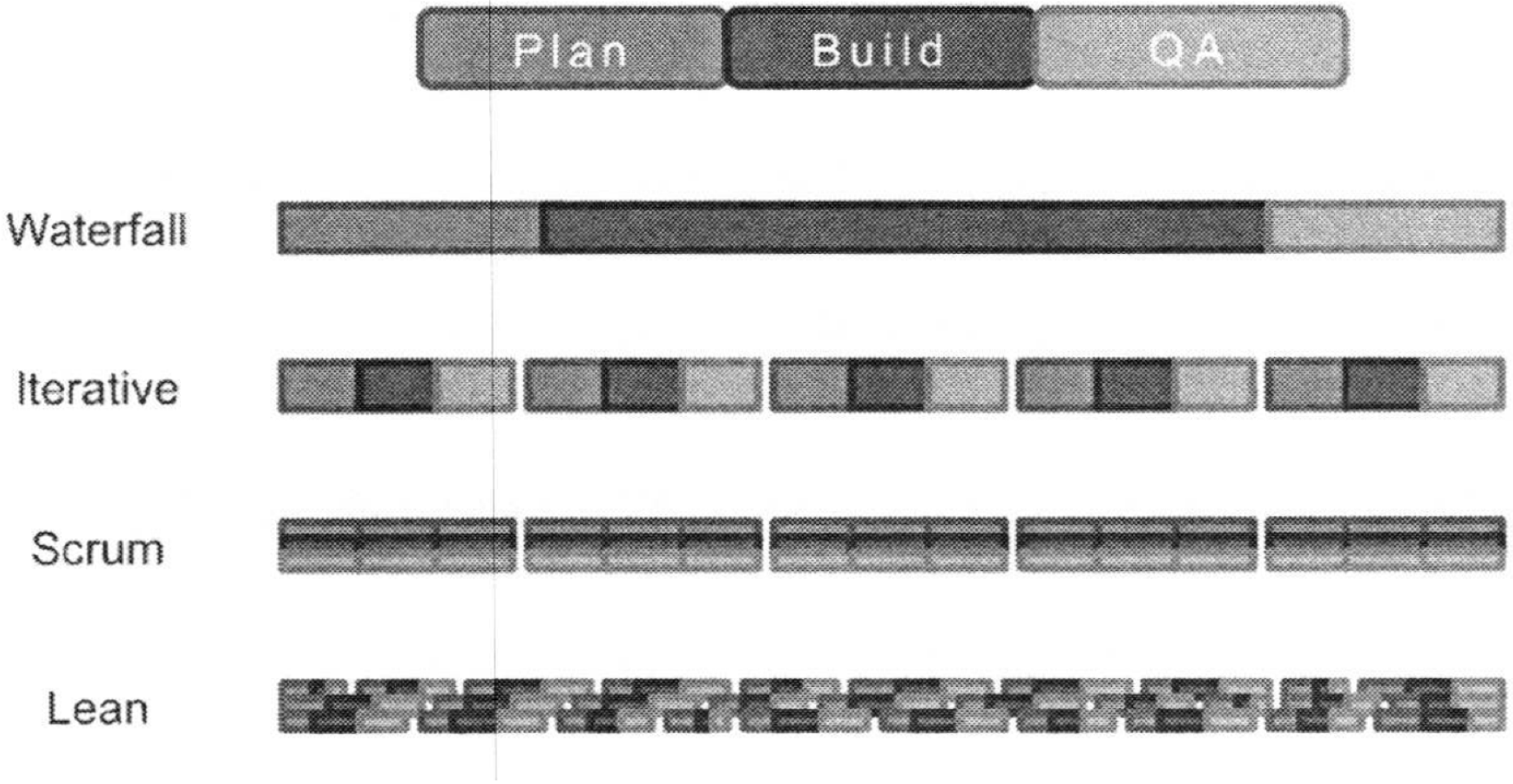


Iterative Development

Business value is delivered incrementally in
Time-boxed cross-discipline iterations



Differences in Processes



Scrum

- Business Case
- Financing
- Scope & Approach
- Contracts
- Initial Release Plan
- Assemble Team

Product Owner
Establishes vision and
Prioritizes Product Backlog

Burn Down

Daily scrum
< 15 minutes

Sprint
1 to 4 weeks

Demo
½ Day

Sprint retrospective
Half Day

Sprint Planning
1 day
- Acceptance Defined
- Team commits
- Tasks created

Team (BA, QA, Dev, etc.) creates and estimates Sprint Backlog (tasks)

Demonstrable Release

Velocity

Burn up

- Business Case
- Financing
- Scope & Approach
- Contracts
- Initial Release Plan
- Assemble Team

Product Owner
Establishes vision and
Prioritizes Product Backlog

Product Backlog

Sprint Planning

- Business Case
- Financing
- Scope & Approach
- Contracts
- Initial Release Plan
- Assemble Team

Product Owner
Establishes vision and
Prioritizes Product Backlog

Sprint Planning
1 day

- Acceptance Defined
- Team commits
- Tasks created

Sprint Backlog

- Business Case
- Financing
- Scope & Approach
- Contracts
- Initial Release Plan
- Assemble Team

Product Owner
Establishes vision and
Prioritizes Product Backlog

Sprint Planning
1 day

- Acceptance Defined
- Team commits
- Tasks created

Team (BA, QA, Dev, etc.) creates
and estimates Sprint Backlog (tasks)

Sprint

- Business Case
- Financing
- Scope & Approach
- Contracts
- Initial Release Plan
- Assemble Team

Product Owner
Establishes vision and
Prioritizes Product Backlog

Sprint Planning
1 day
- Acceptance Defined
- Team commits
- Tasks created

Sprint
1 to 4 weeks

Team (BA, QA, Dev, etc.) creates and estimates Sprint Backlog (tasks)

Demonstrable Release

Scrum

- Business Case
- Financing
- Scope & Approach
- Contracts
- Initial Release Plan
- Assemble Team

Product Owner
Establishes vision and
Prioritizes Product Backlog

Daily scrum
< 15 minutes

Sprint
1 to 4 weeks

Sprint Planning
1 day
- Acceptance Defined
- Team commits
- Tasks created

Team (BA, QA, Dev, etc.) creates and estimates Sprint Backlog (tasks)

Demonstrable Release

Sprint Burn Down

- Business Case
- Financing
- Scope & Approach
- Contracts
- Initial Release Plan
- Assemble Team

Product Owner
Establishes vision and
Prioritizes Product Backlog

Burn Down

Daily scrum
< 15 minutes

Sprint
1 to 4 weeks

Sprint Planning
1 day
- Acceptance Defined
- Team commits
- Tasks created

Demonstrable Release

Team (BA, QA, Dev, etc.) creates and estimates Sprint Backlog (tasks)

- Business Case
- Financing
- Scope & Approach
- Contracts
- Initial Release Plan
- Assemble Team

Product Owner
Establishes vision and
Prioritizes Product Backlog

Burn Down

Daily scrum
< 15 minutes

Retrospective & Demo

Sprint
1 to 4 weeks

Demo
½ Day

Sprint retrospective
Half Day

Sprint Planning
1 day
- Acceptance Defined
- Team commits
- Tasks created

Demonstrable Release

Team (BA, QA, Dev, etc.) creates and estimates Sprint Backlog (tasks)

- Business Case
- Financing
- Scope & Approach
- Contracts
- Initial Release Plan
- Assemble Team

Product Owner
Establishes vision and
Prioritizes Product Backlog

Burn Down

Daily scrum
< 15 minutes

Velocity

Sprint
1 to 4 weeks

Demo
½ Day

Sprint retrospective
Half Day

Sprint Planning
1 day
- Acceptance Defined
- Team commits
- Tasks created

Team (BA, QA, Dev, etc.) creates and estimates Sprint Backlog (tasks)

Demonstrable Release

Velocity

Burn up

For Agile Software

SCRUM Says…	Other poplar agile practices say…
Here is a framework for managing software projects. We will leave the implementation specifics up to you	*We can give you all the specifics you want!*
Produce a shippable product every iteration	*Produce a shippable product continuously (Lean Software Development)*

Agile Scrum Immersion

Scrum Rules & Practices

Scrum

- Business Case
- Financing
- Scope & Approach
- Contracts
- Initial Release Plan
- Assemble Team

Product Owner
Establishes vision and
Prioritizes Product Backlog

Burn Down

Daily scrum
< 15 minutes

Sprint
1 to 4 weeks

Demo
½ Day

Sprint retrospective
Half Day

Sprint Planning
1 day
- Acceptance Defined
- Team commits
- Tasks created

Team (BA, QA, Dev, etc.) creates and estimates Sprint Backlog (tasks)

Demonstrable Release

Velocity

Burn up

Scrum Rules & Practices

1. Roles of Scrum
2. Product Backlog
3. Sprint Planning
4. Daily Scrum
5. Demonstration
6. Retrospective

Exercise

Resort Brochure

- We want to create a Caribbean resort brochure enticing readers to book a vacation.
- We will build it over 2 Sprints.
  - 1 Sprint= 3 Days
  - 1 Day= 10 Minutes
  - Extra time for lecture and planning.

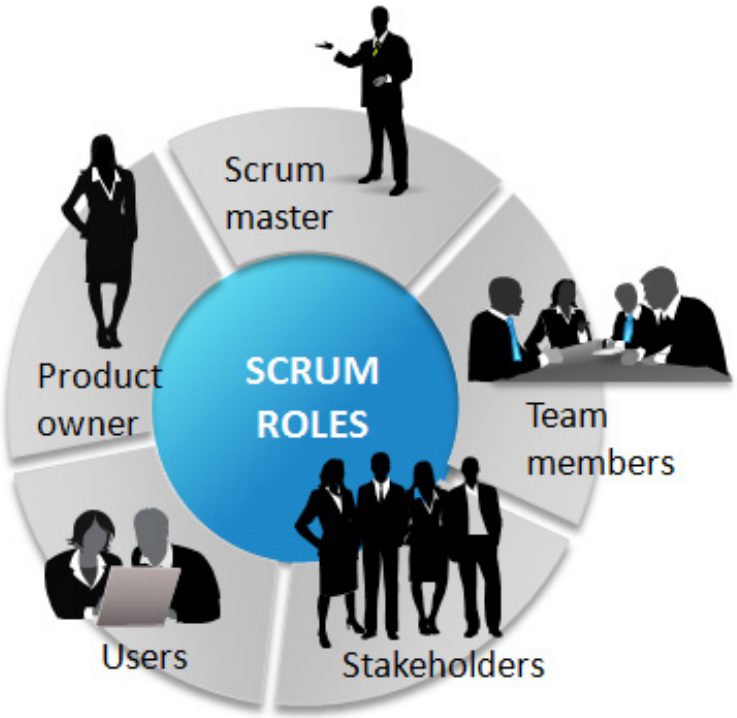
Exercise

Product backlog

- In teams, create a wish list for your ultimate resort.
- Use the 3x5 cards provided
- Be Creative!

Scrum Teams

- Scrum teams are optimally 7 +/- 2 people
- Scrum only defines 3 roles
  - Product Owner
  - Scrum Master
  - Team



Product Owner

- A single person representing the stakeholder(s) and /or customer(s)
- Has final authority (and therefore accountability) in creating and prioritizing the product backlog
- Must be available to the Team at any time

Scrum Master

- A facilitator for the Team and Product Owner
- Clears roadblocks for the Team
- Insulates the Team
- Usually track and report the Team's progress
- Informs people of the rules of Scrum

Scrum Master

- The Scrum Master is not a traditional project manager.
- Unlike command/control PMs, the Scrum Master takes a servant-leader role.
- This supports the Agile Principle:
  - Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The Team

- Cross-functional group that have committed to accomplishing goals of the sprint
- Architects, programmers, analysts, testers, UI designers, DBAs, etc.
- The Team self-organizes and has autonomy in how best to complete their tasks(and therefore accountability)

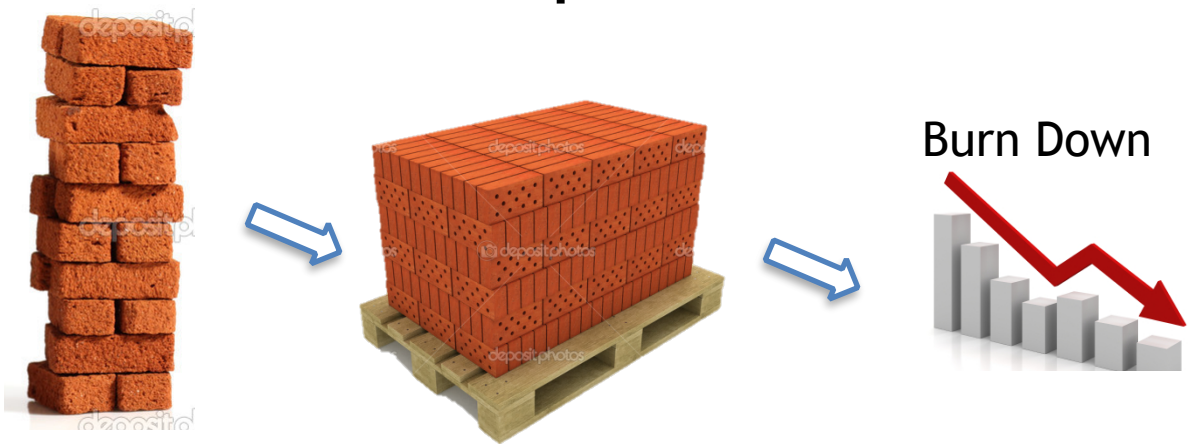
Exercise

Product owner

- Select a Product Owner
- Have the Product Owner establish the vision for the brochure by prioritizing all the features

Scrum Artifacts

- Scrum recommends two essential artifacts:
  - Product Backlog
  - Sprint Backlog
- Each of these backlogs provide information to products burn downs

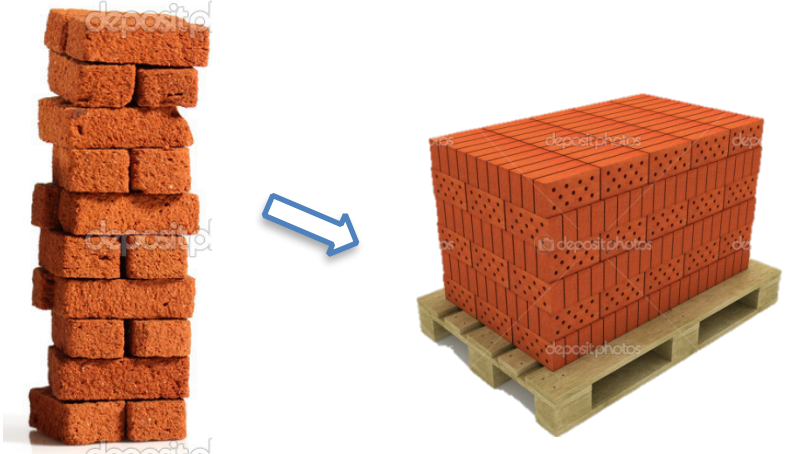


Sprint Planning –One Day

- First Half (Product Owner +Team)
  - Team commits to a set of features from the top of the product backlog
  - Product owner helps clarify requirements and defines acceptance criteria for sprint features
- Second Half (Team)
  - Using the acceptance criteria, team breaks each feature into tasks
  - Team estimates each task
  - Team members volunteer for few days' worth of tasks

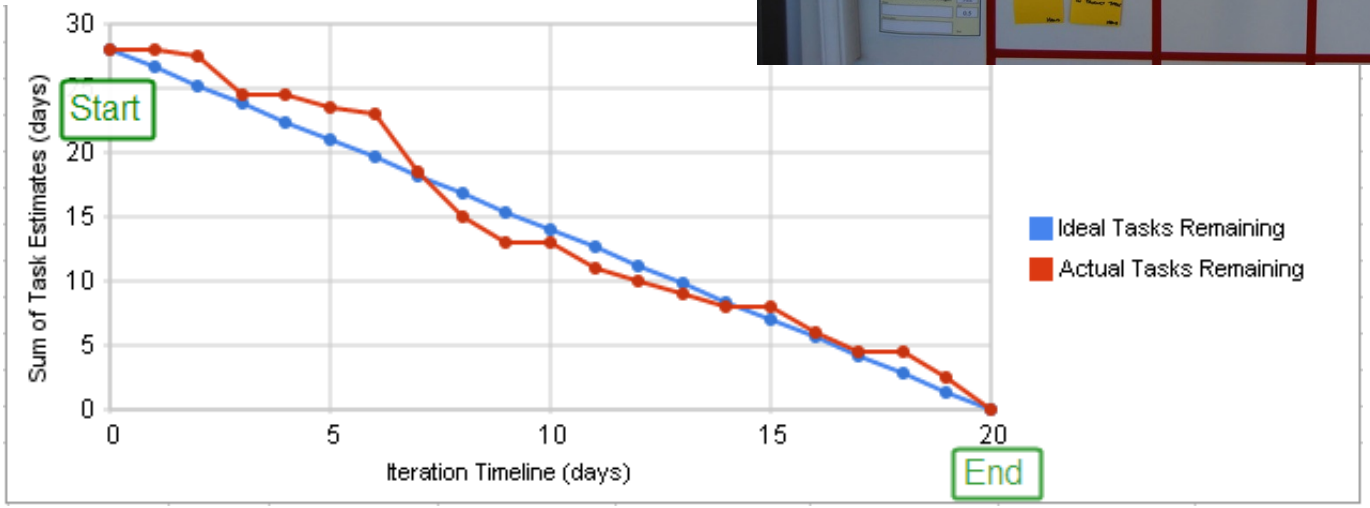
Features to tasks

- Once the Team has chosen the features for the sprint, task breakdown begins.
- This key activity is where much of the discussion around approach and design happen. Conversation is King!
- The Sprint backlog is the collection of these tasks.



Scrum Artifacts

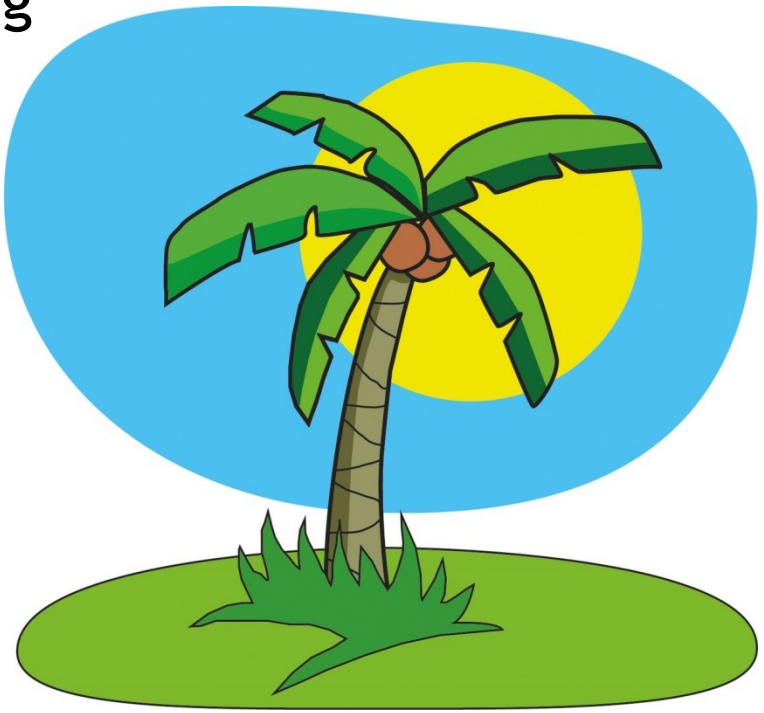
Backlogs,
Burnups, and
Burndowns can
Take many forms



Scrum Artifacts

Exercise

Sprint Backlog

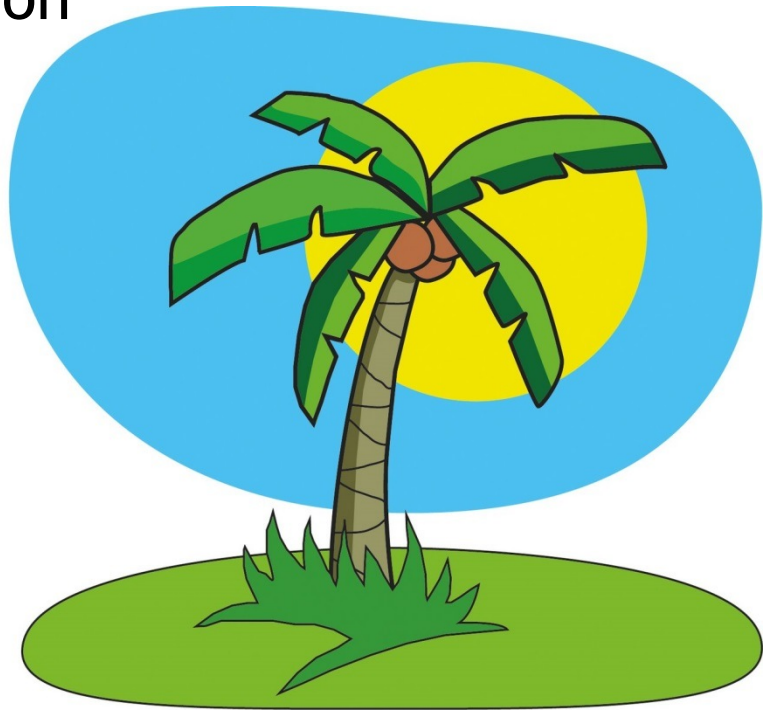


- The Team commits to what features they think they can be accomplished within the first Sprint
- Break the Sprint 1 features in to tasks (Sprint Backlog)
- Place the Sprint 1 Stories and tasks on a Task Board

Exercise

Implementation

- BEGIN DAY 1!
- Each person grabs a task and moves it to ‘Work in Progress’
- When task is complete grab another one



Daily Scrum/Stand Up…

…is less than 15 minutes and answers:

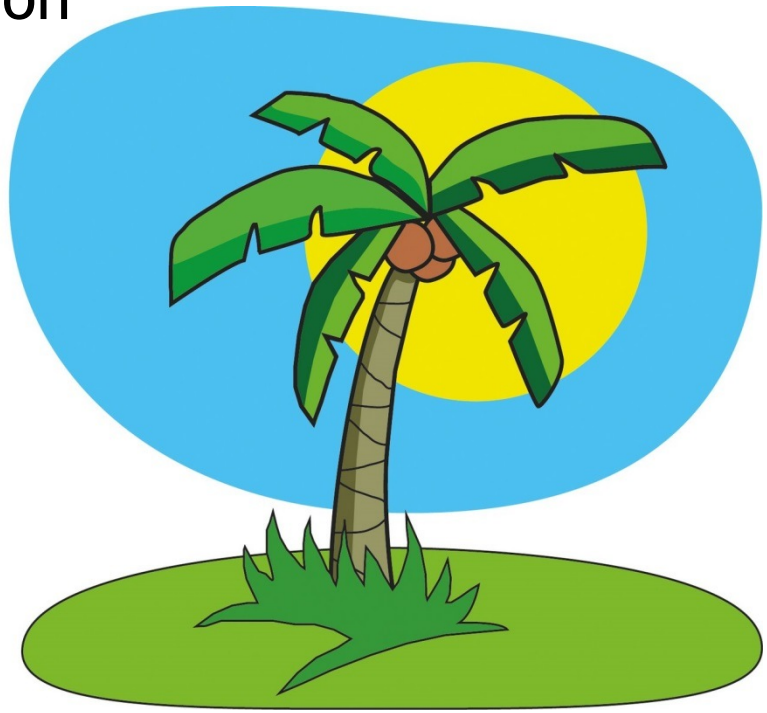
What did you accomplish?

What do you plan to accomplish?

What is keeping you from being successful?

Exercise

Implementation

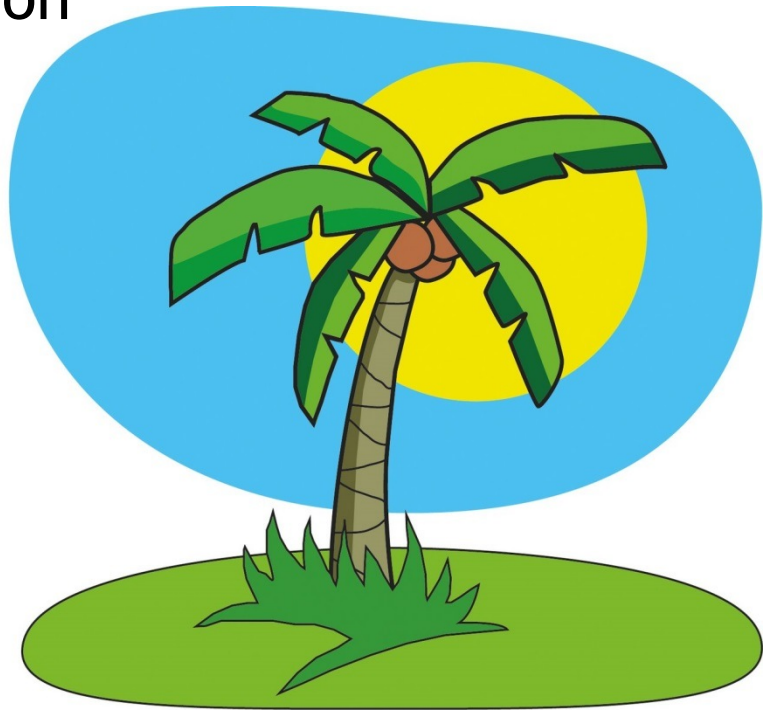


- BEGIN DAY 2!
- Start the day with a Daily Scrum
- Each team member moves their tasks to the appropriate column on the task board
- If you get stuck on a task, move it to ‘ blocked’

Exercise

Implementation

- BEGIN DAY 3!
- Rinse & Repeat!



Sprint Demonstration

- Team Meets with Stakeholders to demo the solution
- Product Manager describes the features completed
- Exceptions to success criteria called out
- Stakeholders make comments on the product
- Work Items are added to the Backlog

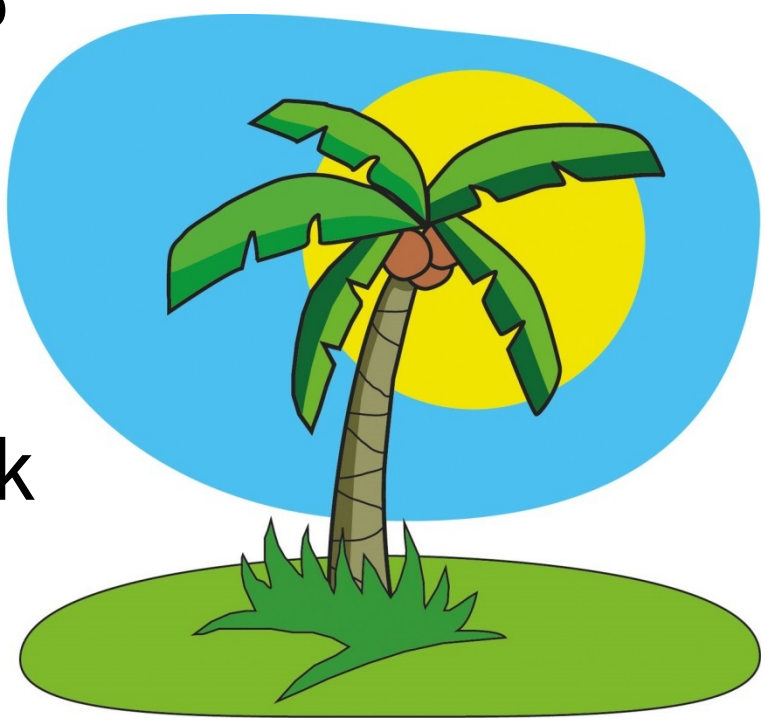
A word about Definition of Done

- Each team should devise and publish a DoD.
- The DoD is a list of value-add activities that must be complete before someone can claim “done”.
- DoD often applies at different levels of granularity: task, story, sprint, and release.
- The DoD should not be static, but it should always be visible.
- Discussion about DoD can often lead to process improvements.

Exercise

Sprint Demo

- Have a member of the team demo the brochure
- Have the Product Owner accept or reject the work



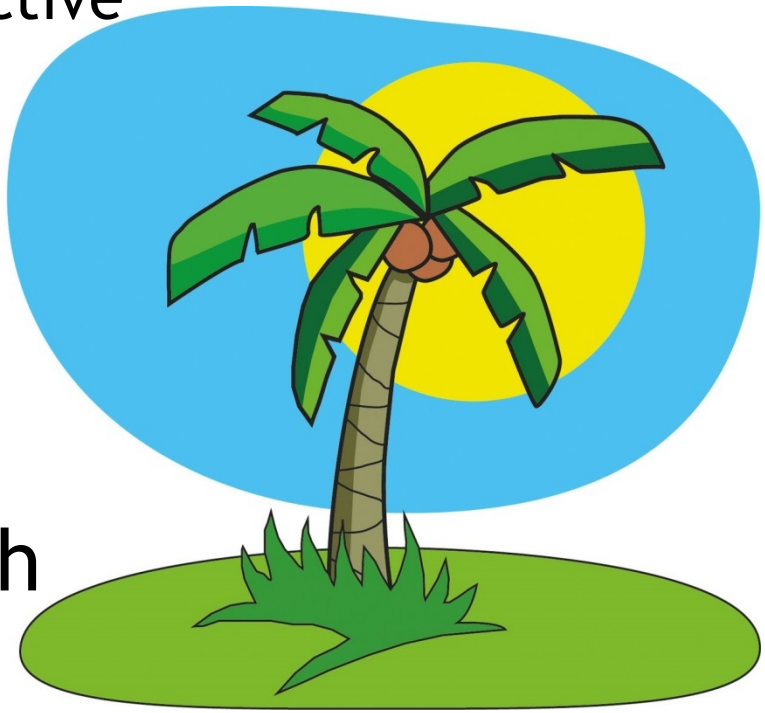
Sprint Retrospective

- What was good about the way we perform the iteration
- Three categories of focus : Environment , Team, Self
- What can we do to improve the next iteration
- Use silent writing to encourage everyone to feel comfortable making suggestions
- Immediately adopt improvements for next iteration ,promote improvements to other teams

Exercise

Sprint Retrospective

- Have a member of the team facilitate the retrospective
- Each member must add at least one item to each the+ and –column



Exercise

Sprint 2

- Do another 3-day sprint!

For Agile Practices…

SCRUM says…	Other poplar agile practices say…
Here are the rules. Follow them to great success!	*Evolve towards the rules, changing when you understand the consequences*
1. Roles of Scrum *2. Product Backlog* *3. Sprint Planning* *4. Daily Scrum* *5. Demonstration* *6. Retrospective*	
One Definition of done	*Done, done, DONE!*

Agile Scrum Immersion

Requirements

Capturing Requirements

- How do you capture requirements?

Requirements

Functional Requirements (Purposes)

Use Cases, User Stories, IEEE System Shall's

Non-Functional Requirements(Qualities)

Supplemental Specs, User Stories

Business Rules

Spread Sheets, web sites, brains etc.

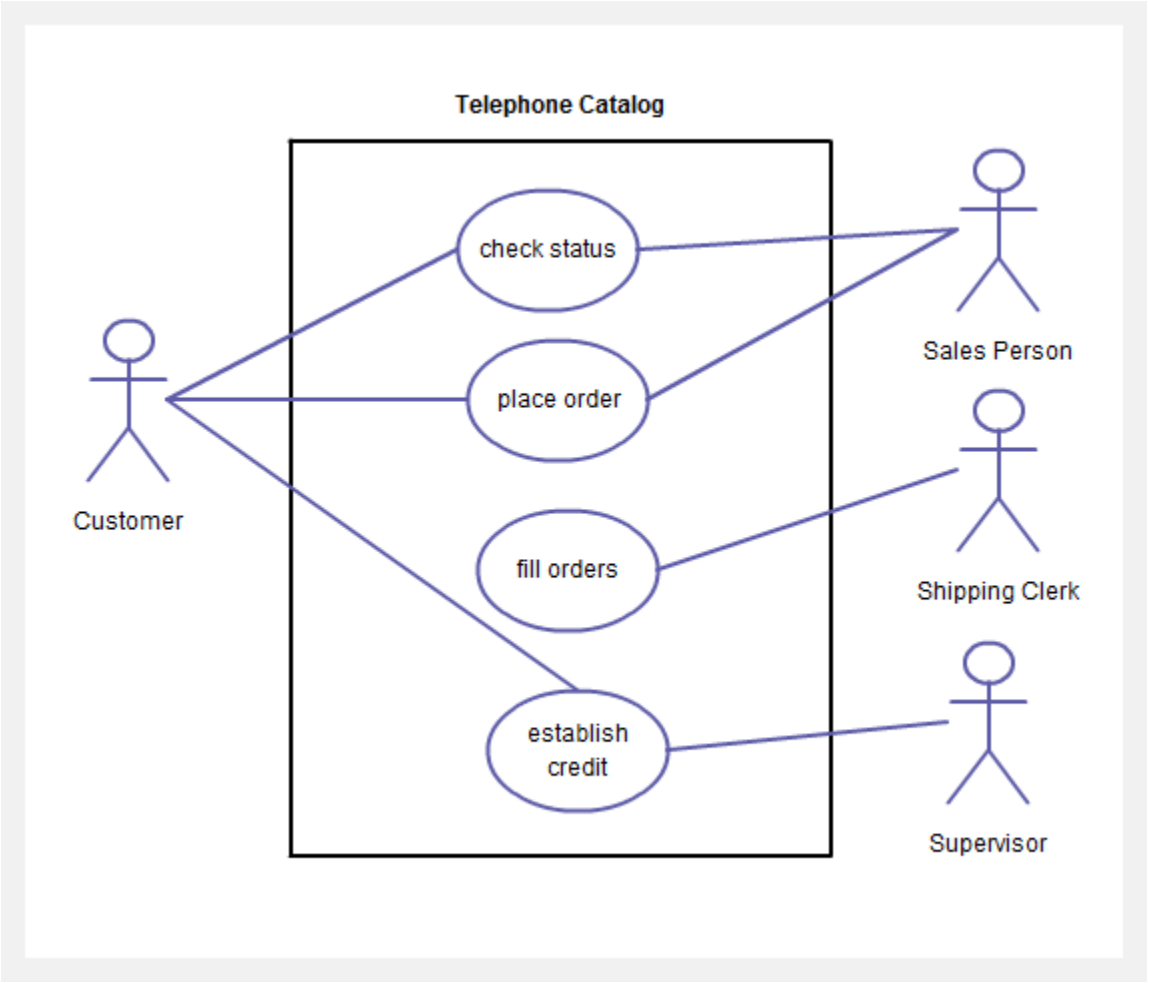
Requirements

What is your goal?

To Document?
- IEEE 830
- Detailed Use cases

To Represent?
- User Stories
- Use Case Briefs

Use Case Diagram



Use Case

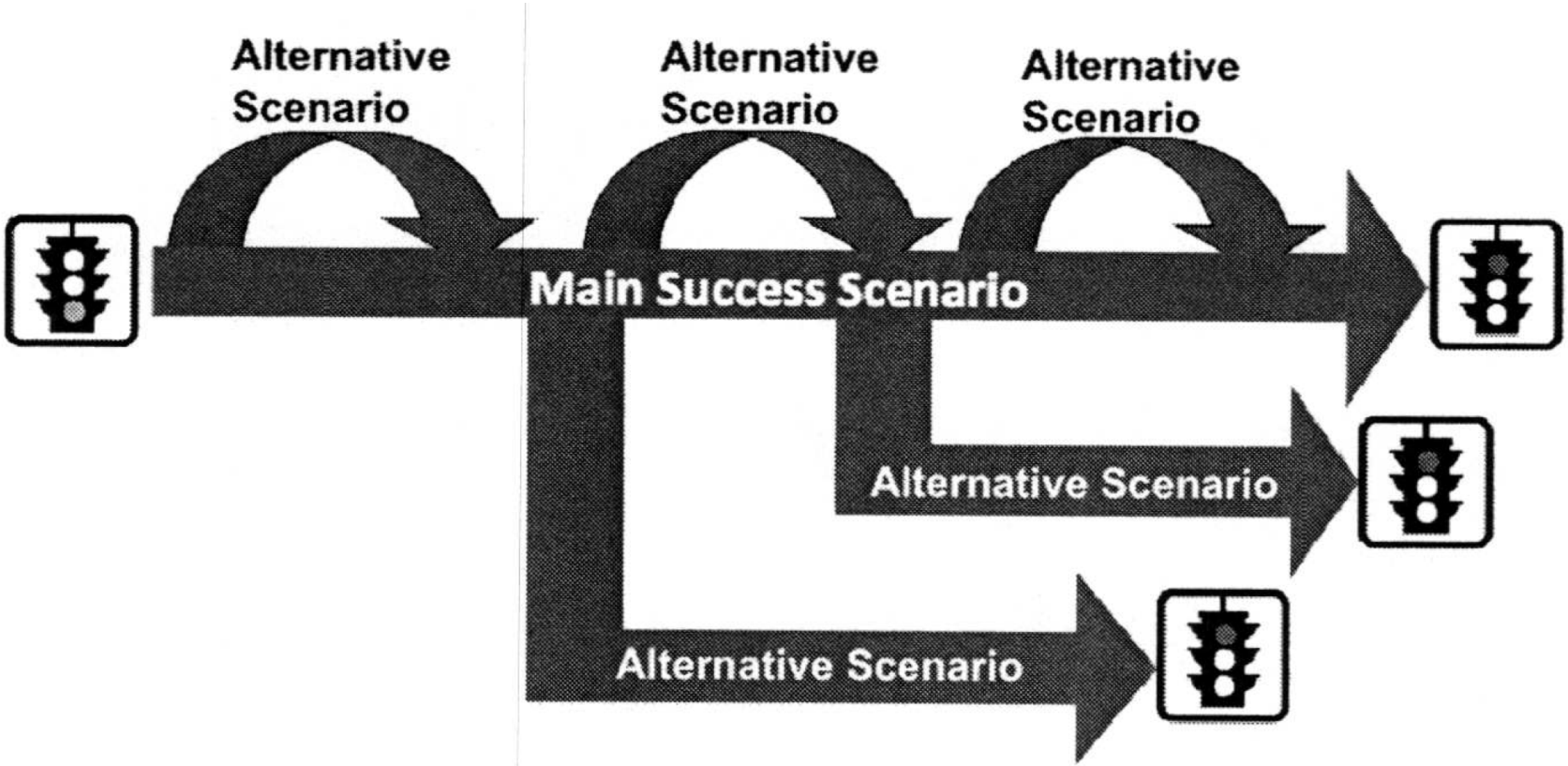
- A use case is a story of how users and the system work together to get something done.
- Breaks down the functionality of the system
- Originally developed to exploit the essential value of story –telling to teach and learn

Use Case: by the book

Typically, they are used to capture the requirements of a system, that is , what a system is supposed to do. The key concepts associated with use cases are actors, use cases, and the subject. The subject is the system under consideration to which the use cases apply. The users and any other systems that may interact with the subject are represented as actors. Actors always model entities that are outside the system. The required behavior of the subject is specified by one or more use cases, which are defined according to the need of actors.

UML Superstructure Specification

Scenarios


Alternative Scenario
Alternative Scenario
Alternative Scenario
Main Success Scenario
Alternative Scenario
Alternative Scenario

User Story

- Three C’s
  - Card
  - Conversation
  - Confirmation

Accuracy Vs. Precision

For years we've been taught that we must drive out ambiguity in requirements

That remains true, but does it follow that all ambiguity at all times is bad?

User Stories

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

User Story Example

As a Blog Reader, I want to
comment on a blog entry
so that I can
contribute to the
conversation.

Good Stories Are…

Independent

Negotiable

Valuable

Estimable

Small

Testable

Other Requirements

- **Business Rules** – detailed business or technical rules that are application independent. Credit card numbers must be between 12 and 16 characters in length .
- **Non-Functional Requirements** – attributes of the system or points of future variations.
  - While the use cases may specify limits (such as no more than ten books per order), all such limits should be run –time configurable.
  - Users should be able to find a book in 3 or fewer mouse clicks .

Non-Functional Categories

Usability
Scalability
Portability
Maintainability
Availability
Accessibility
Supportability
Security
Performance
Cost
Legal
Cultural
……

Non –Functional Story Example

As a Blogger, I want to

support many readers

so that I can

reach a wider audience

For Requirements…

SCRUM says…	Other poplar agile practices say
Keep requirements at a high level. (inch deep, mile wide)	*The Customer is part of the Team*
Put requierments in a product backlog	*Populate the product backlog using the User Story format*
Product Owner completely owns the product backlog	*Add Acceptance Criteria to user Stories*

Exercise
Requirements

- For the given project:
  - Brainstorm functional requirements / features (as user stories)
  - Brainstorm non –functional requirements(as user stories)

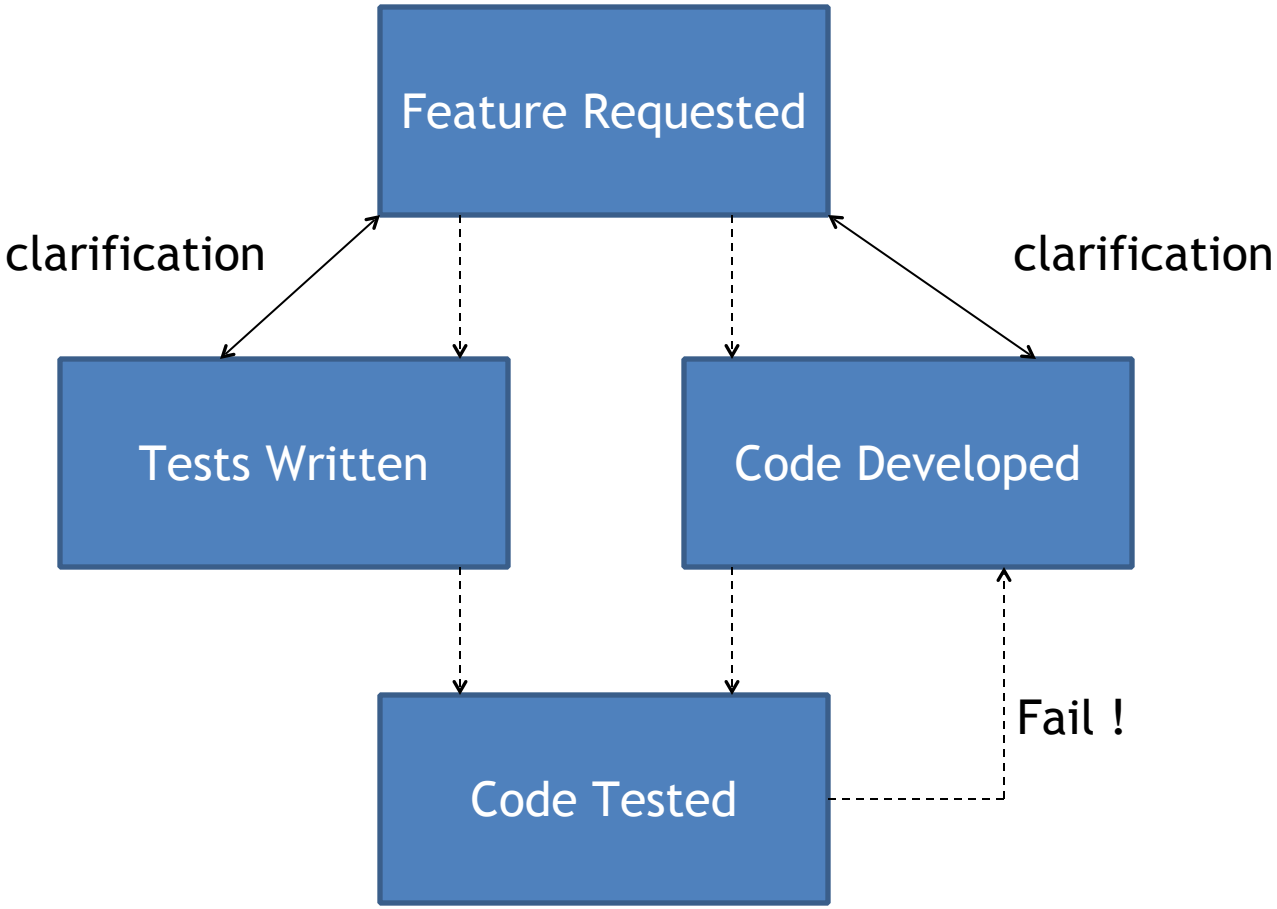
Agile Scrum Immersion

QA

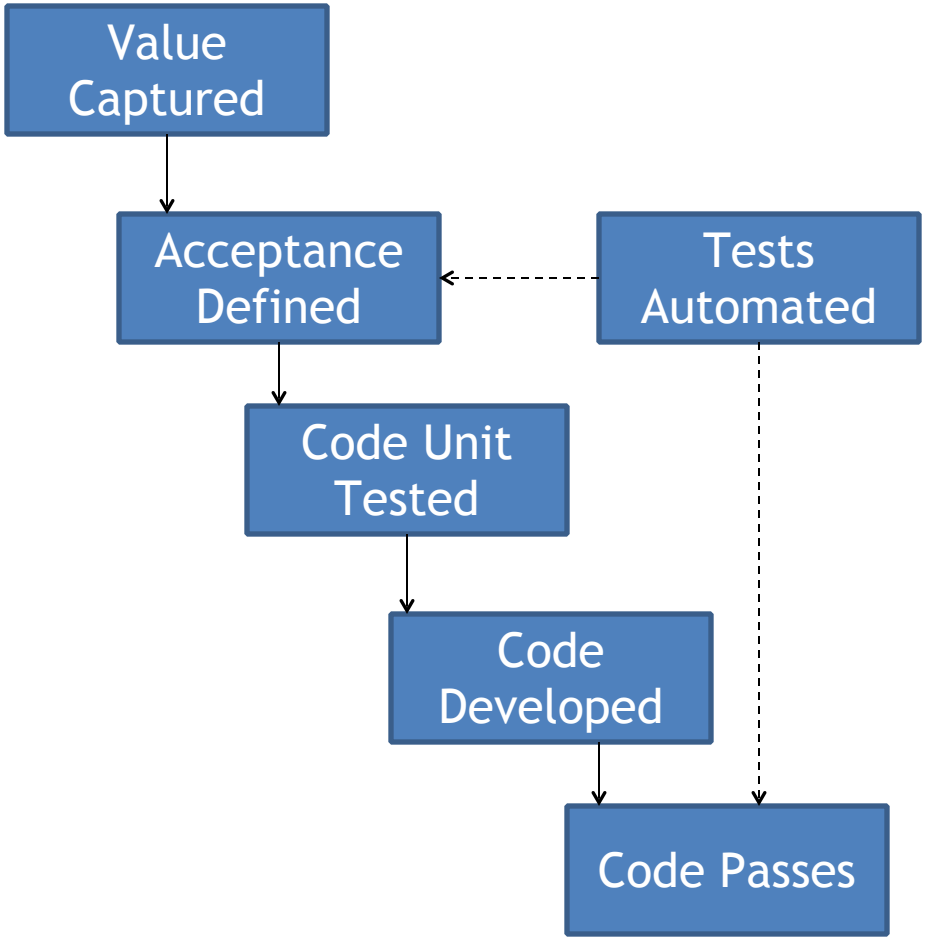
The Role of QA

To provide objective feedback to the stakeholders on quality of the software.

The Role of QA

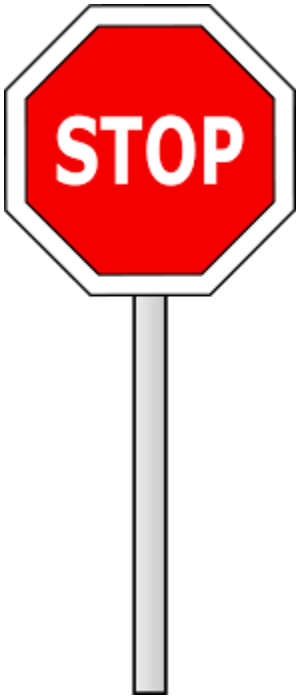


Agile QA

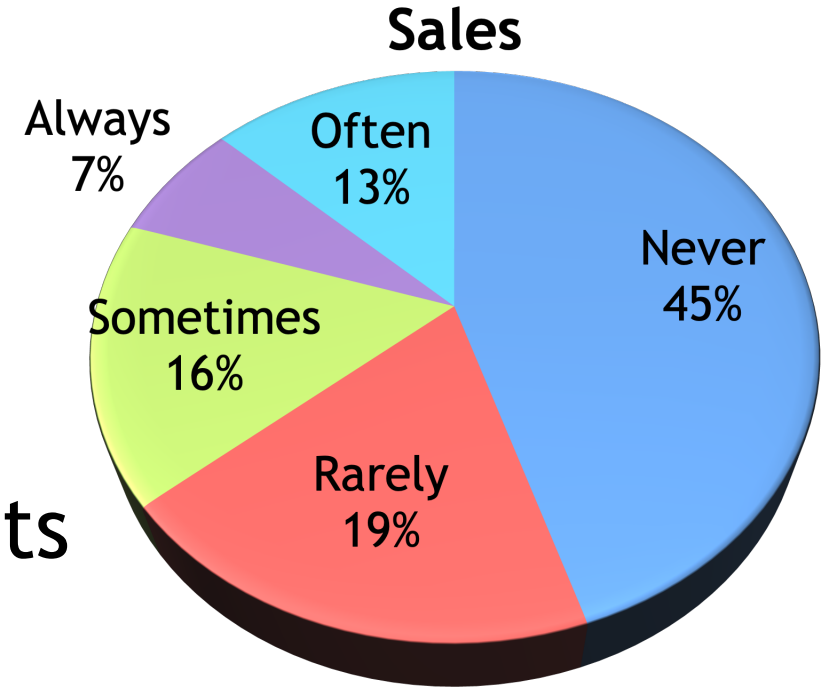

Value Captured
Acceptance Defined
Tests Automated
Code Unit Tested
Code Developed
Code Passes

The Problem

- Building the wrong thing
- Building it wrong



Building the Wrong Thing



- Changing Requirements
- Communication
- No Definition of “Done”

Building it Wrong

- Too many costly defects
- Poor maintainability
- Over-engineering
- Lack of confidence in the code

“I don’t want to touch that. It’ll take forever, and I don’t know what else will break if I do! I’ll just rewrite it.”

Solution

"Fix the RIGHT thing RIGHT."

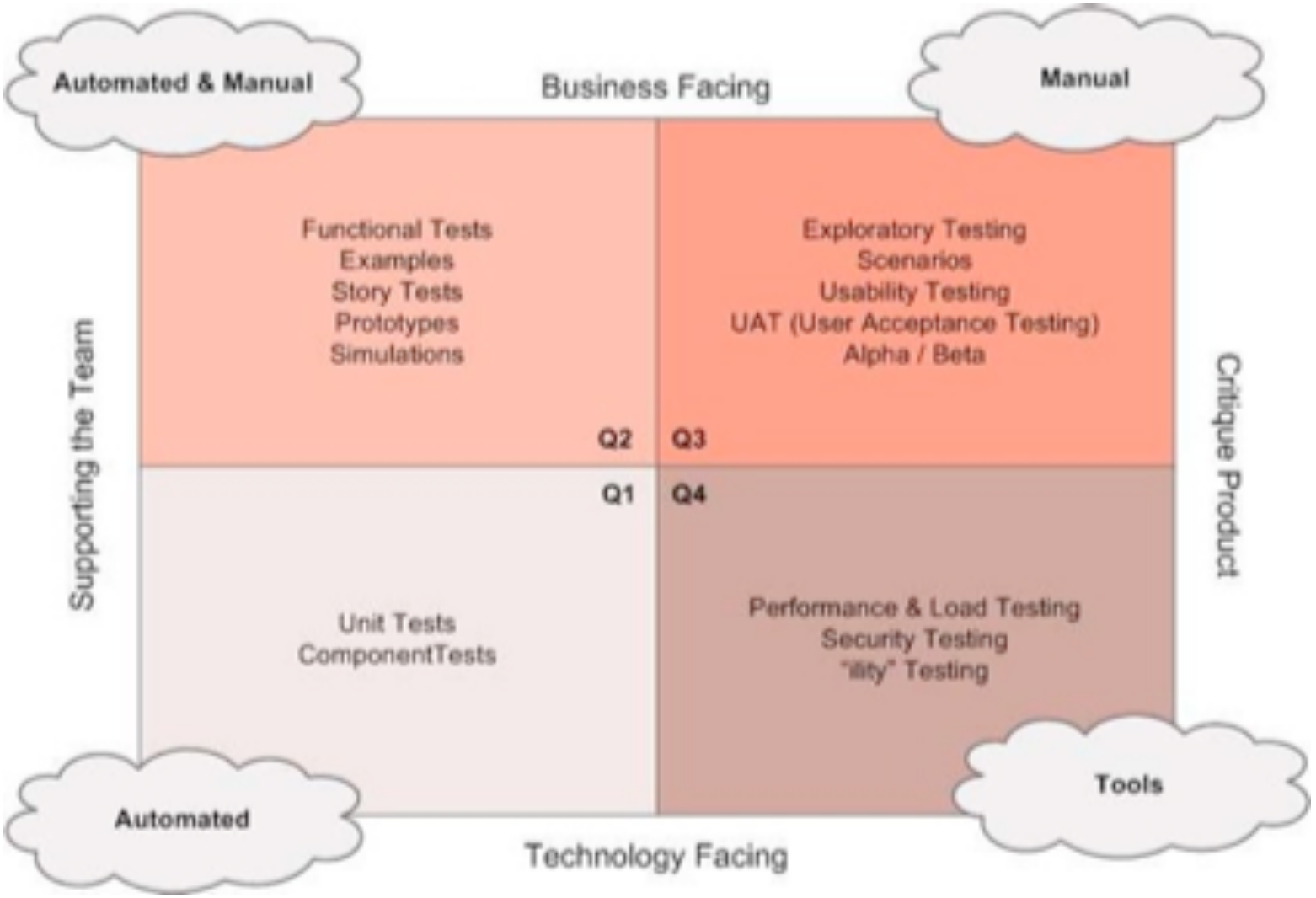
With

Acceptance Testing

And

Test Driven Development

Test Categorization



Setting Expectations

Acceptance Criteria define what the customer will see in order to approve the work as being complete.

- Test cases
- Operation contracts
- Before and After picture

Test Cases

Each test case describes in a step-by-step manner a specific interaction with the system and the expected response

Test 1

1. The user visits the home page.
   The home page displays a banner, … , search text box.
2. The user searches for “O’Dell”.
   The search results page with 1 book, Object-Oriented Methods, should display.
3. The user selects to buy Object-Oriented Methods.

Acceptance Tests

Given <a condition>

When <event occurs>

Then <system should…>

Acceptance Test

Test that…

Acceptance Tests

Demonstrate that…

Acceptance Test are…

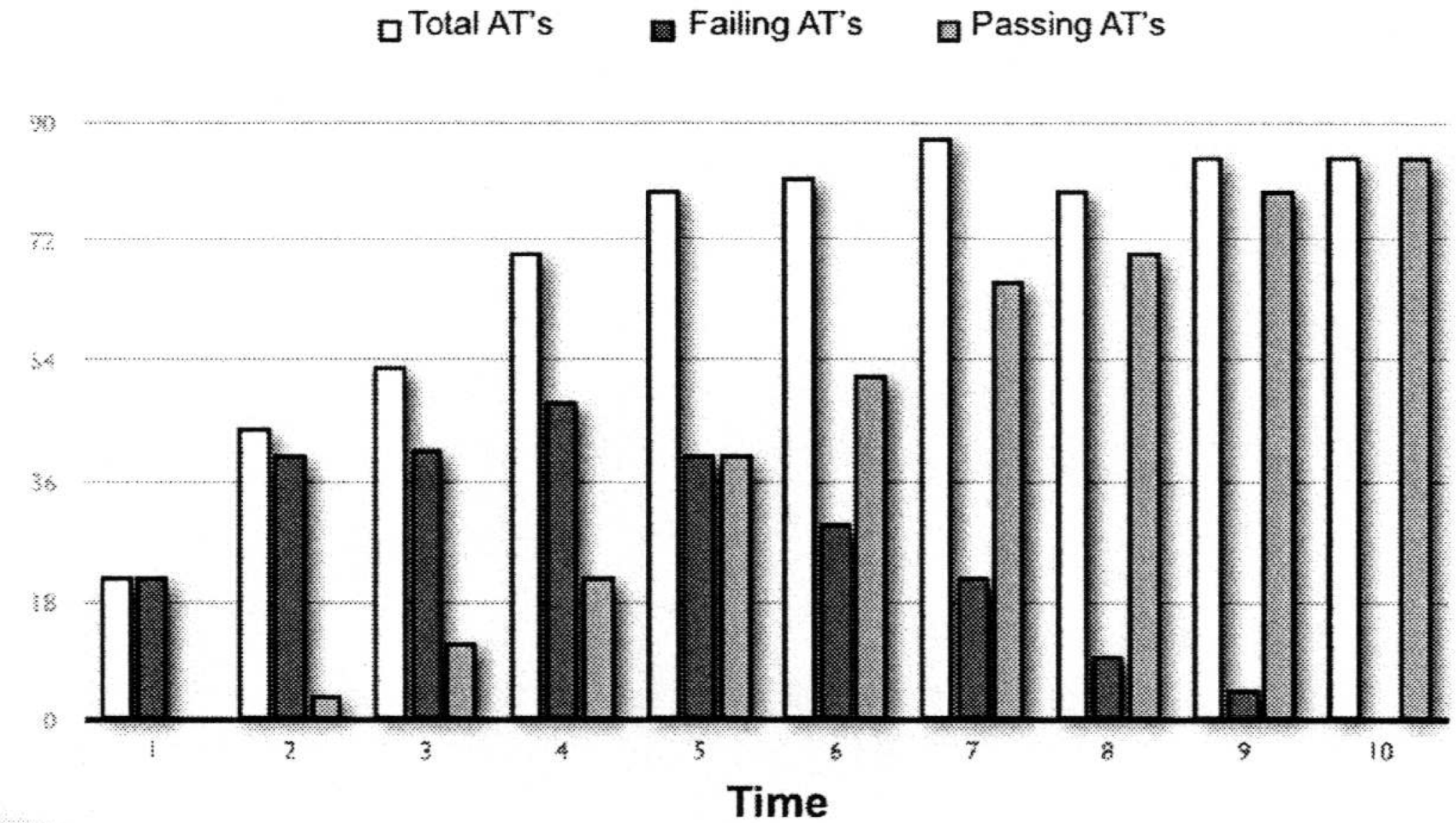
Success
Advance
Fail
Error

Acceptance Tests

Multiplication

First Number	Second Number	Product?
10	*4*	*40*
12.3	*5*	*61.5 expected*
		61 actual

The Goal


Total AT's
Failing AT's
Passing AT's
90
72
54
36
18
0
1
2
3
4
5
6
7
8
9
10
Time

Role of Agile QA

- Help define stories
- Add stories related to testability
- Add stories related to non-functional requirements (usability, performance, etc.)
- Organize non-functional testing
- Help define acceptance criteria for stories before they are completed
- Encourage and/or help developers write good unit tests
- Increase code coverage by adding more automated tests
- Perform exploratory testing on early builds

For QA…

SCRUM says…	Other poplar agile practices say…
Testers are part of the Team from day 1. They perform tasks with all the other pigs.	*Documenting both requirements and tests can be a very redundant activity*
Not much else…	*Write your requirements as test up front and eliminate waste.*
	Adorn user stories with acceptance tests in order to better define when it is considered done.
	Automate acceptance tests with tools such as FitNesse and GreenPepper.

Exercise
QA

- For the given project:
  - Define acceptance criteria for given stories

Agile Scrum Immersion

Planning and Estimation

Dilbert on Estimating

Scrum

- Business Case
- Financing
- Scope & Approach
- Contracts
- Initial Release Plan
- Assemble Team

Product Owner
Establishes vision and
Prioritizes Product Backlog

Burn Down

Daily scrum
< 15 minutes

Sprint
1 to 4 weeks

Demo
½ Day

Sprint retrospective
Half Day

Sprint Planning
1 day
- Acceptance Defined
- Team commits
- Tasks created

Team (BA, QA, Dev, etc.) creates and estimates Sprint Backlog (tasks)

Demonstrable Release

Velocity

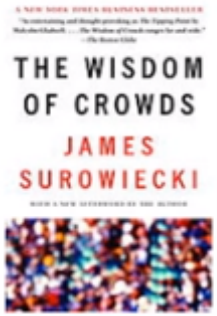
Burn up

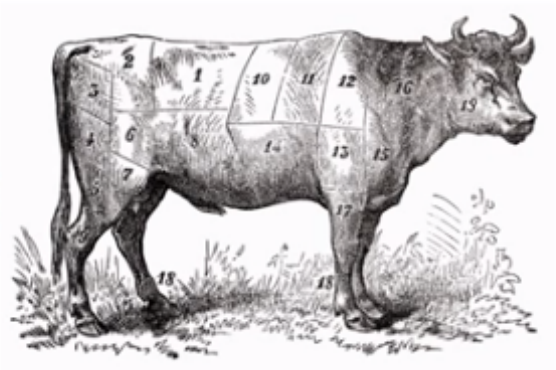
Sprint Planning – One day

- First Half (Product Owner + Team)
  - Team commits to a set of features from the top of the product backlog
  - Product owner helps clarify requirements and defines acceptance criteria for sprint features.
- Second Half (Team)
  - Using the acceptance criteria, team breaks each feature into tasks
  - Team estimates each task
  - Team members volunteer for a few days worth of tasks

Wisdom of Crowds

The Wisdom of Crowds





average of 800 guesses = 1,197
actual weight of the ox = 1,198

The Wisdom of Crowds, surowiecki, 2005

Planning Poker

Estimation Units

- Hours / Days
- Or something less concrete:
  - T-Shirt Sizes (xs, s, m, l, xl, xxl)
  - Story Points
  - Function Points
  - Gummy Bears

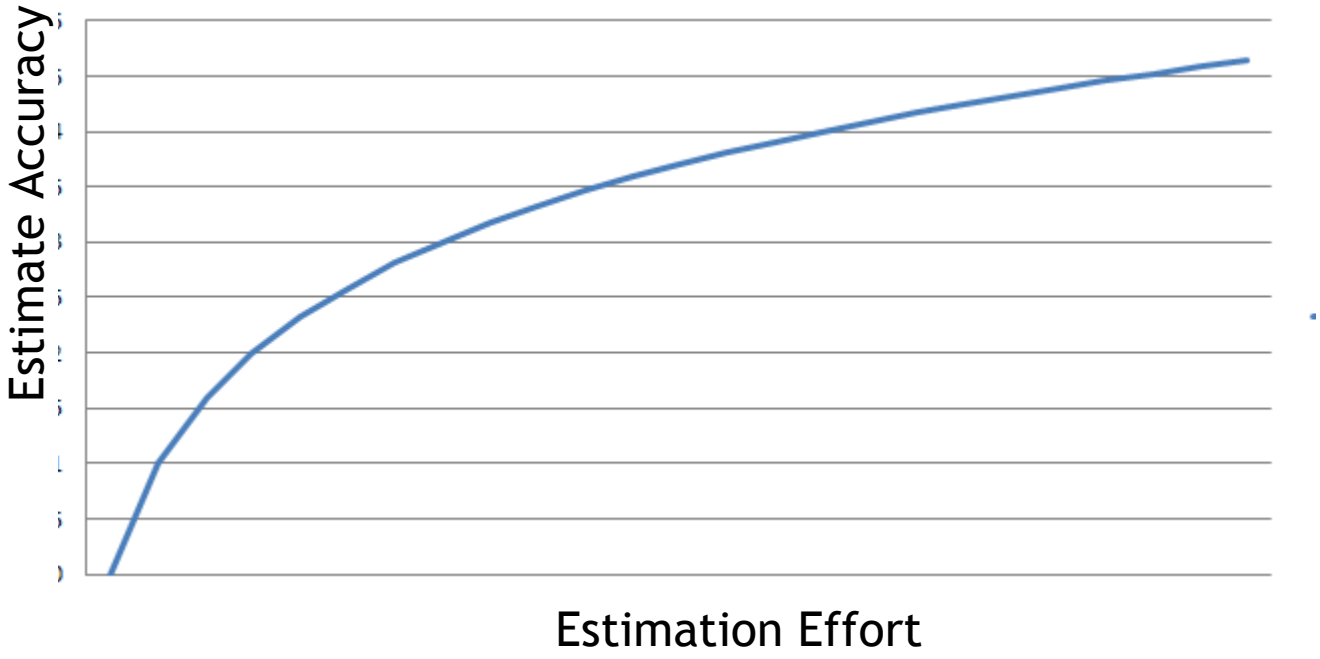
Exercise

Sizing Dogs

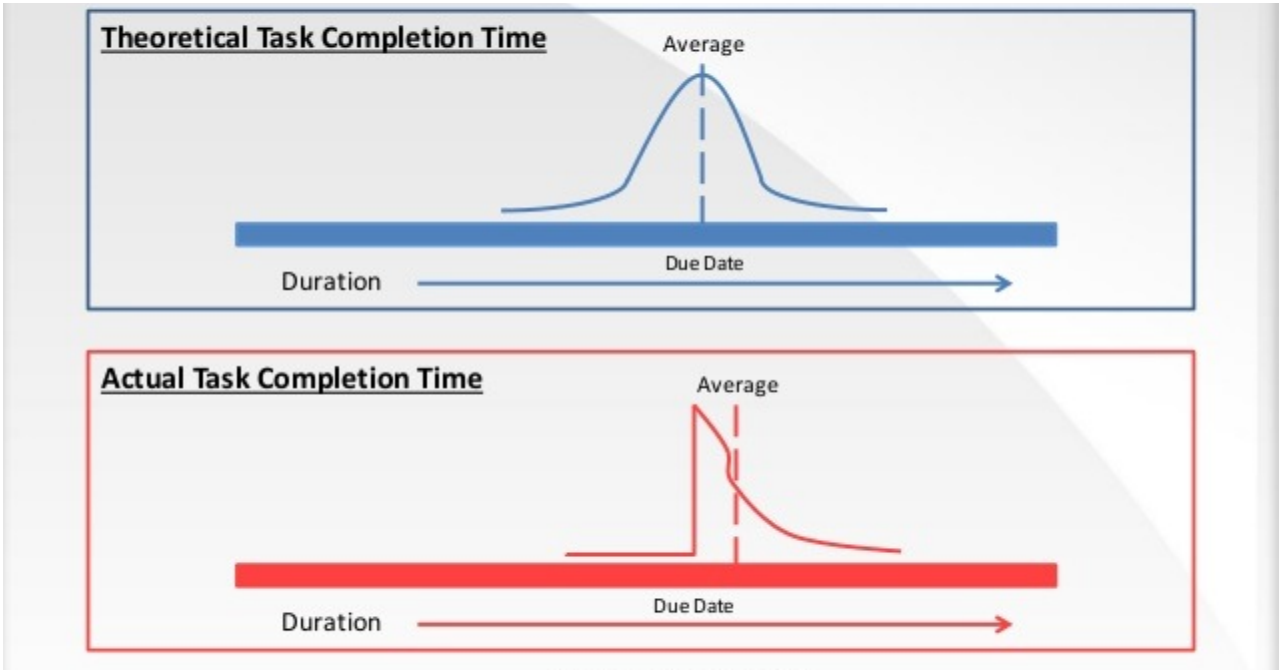
Size the dogs listed below:

- Chihuahua
- Great Dane
- Golden Retriever
- Standard Poodle
- Newfoundland
- Austrian Guildenbaur

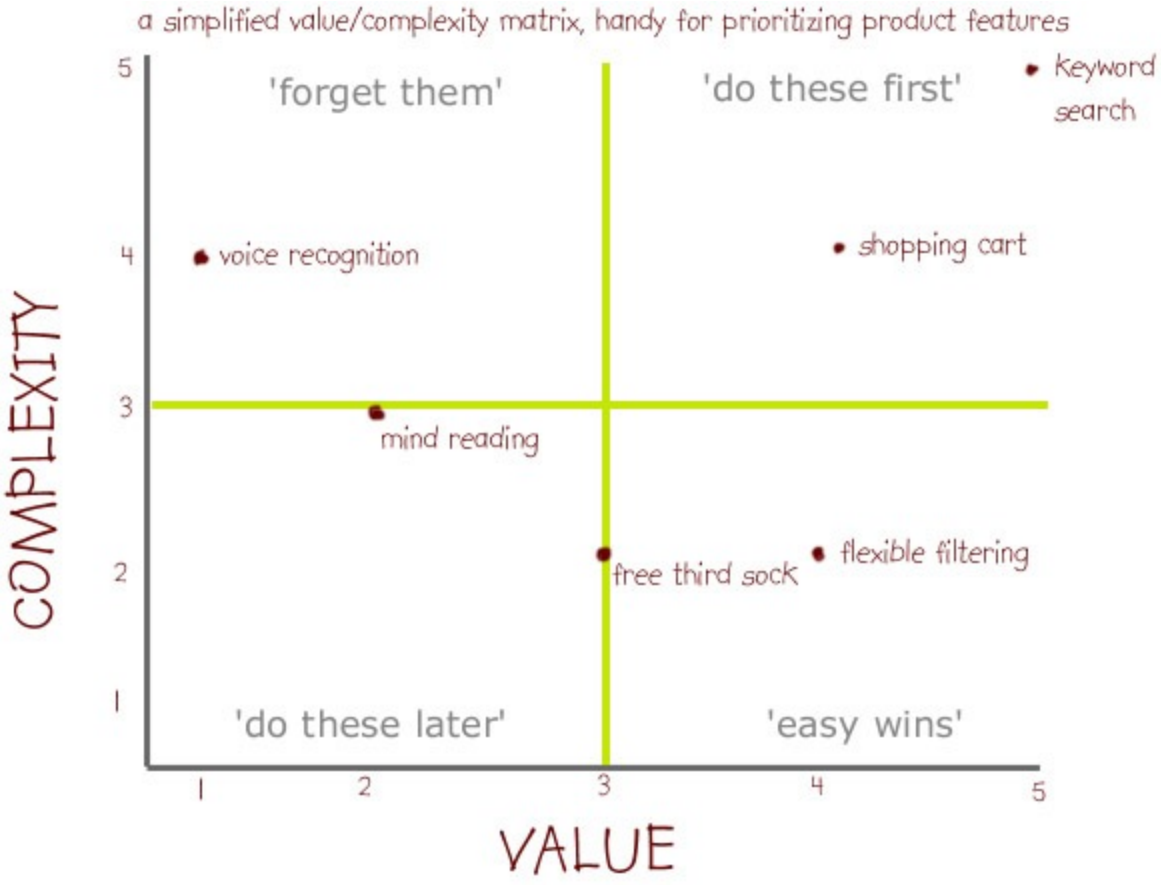
Diminishing Returns



Diminishing Returns



Value/ complexity Matrix

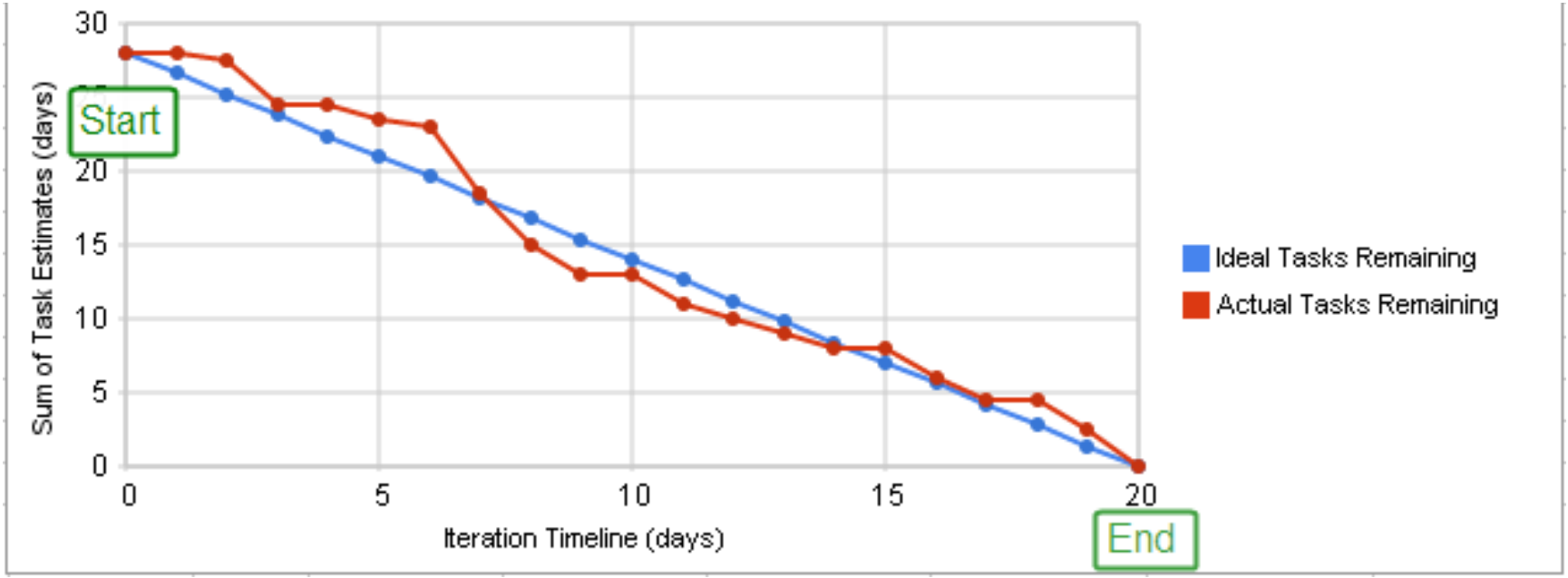


Scrum Board

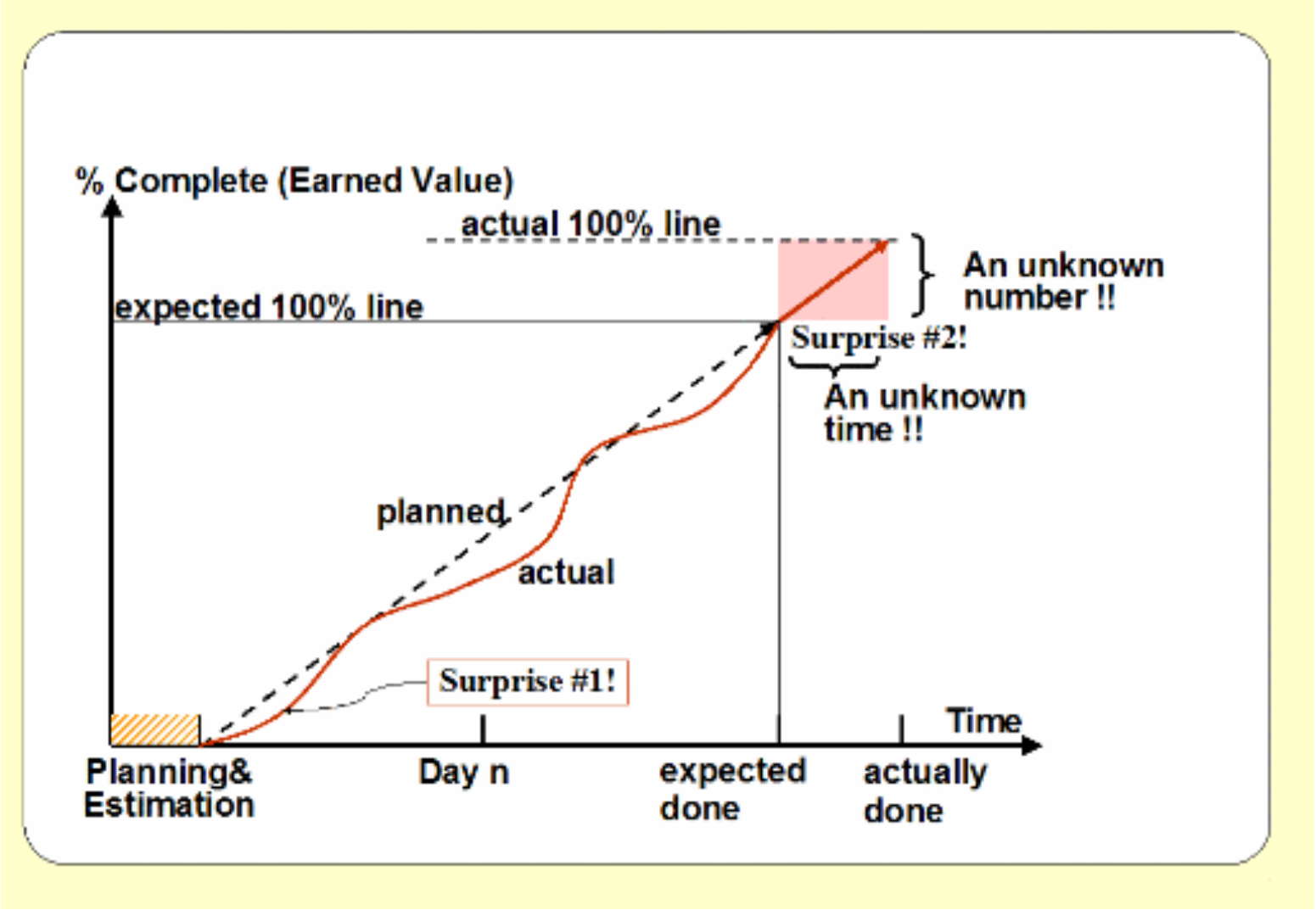
Feature Name	Scheduled	Active	Blocked	Closed
Feature 1	*Task 1.4*	*Task 1.2*	*Task 1. 3*	*Task 1.1*
Feature 2	*Task 2.3*	*Task 2.1* *Task 2.2*		
Feature 3	*Task 3.3*	*Task 3.1* *Task 3.2* *Task 3.4*		


RED TEAM
NOT STARTED
IN PROGRESS
FINISHED
DAILY SCRUM 9:30
UAT and Production Tickets
Unplanned Items and Legacy Issues
Support old PDT's
Taxipost Bus check on weight
BUG
BLOCKED
DONE
ON HOLD
Parking lot

Burn-down Chart focuses on effort



Burn-up Chart focuses on value



Daily Scrum/Stand Up…

… is less than 15 minutes and answers:

What did you accomplish?

What do you plan to accomplish?

What is keeping you from being successful?

Sprint Demonstration

- Team meets with Stakeholders to demo the solution
- Product manager describes the features completed
- Exceptions to success criteria called out
- Stakeholders make comments on the product
- Work items are added to the Backlog

Velocity

- How much of the product backlog effort can a team handle in one sprint?
  - Should become more accurate over time
  - Should rise and then level off
  - Combine with effort estimation to create more accurate timelines

For Estimation and Planning

SCRUM says…	Other poplar agile practices say…
The Product Owner is solely responsible for the backlog prioritization	*The backlog should be relatively sized using Nebulous Units of Time (NUT)*
The Team has final say in estimating and determining what they can accomplish during a sprint	*Planning Poker is an effective way of producing estimates*
During the sprint, the Team is left alone and produces the best software possible	*Big Visible charts and other information radiators such as physical task boards and 3.5 cards provide excellent return on investment.*
Work done in previous Sprints helps determine what can be done in an upcoming Sprint (Velocity)	*Beware of Parkinson's Law*
Burn-down and Burn-up charts communicate sprint and project progress	

Exercise

Planning & Estimating

- For the given project:
  - Size feature with Planning Poker (10 min)
  - Prioritize your features (5 min)

Agile Scrum Immersion

Development

Agile Developers

- Build the most important thing first
- Build things completely
- Talk to their customer(s)
- Test their code early and constantly
- Are aligned with the business goals

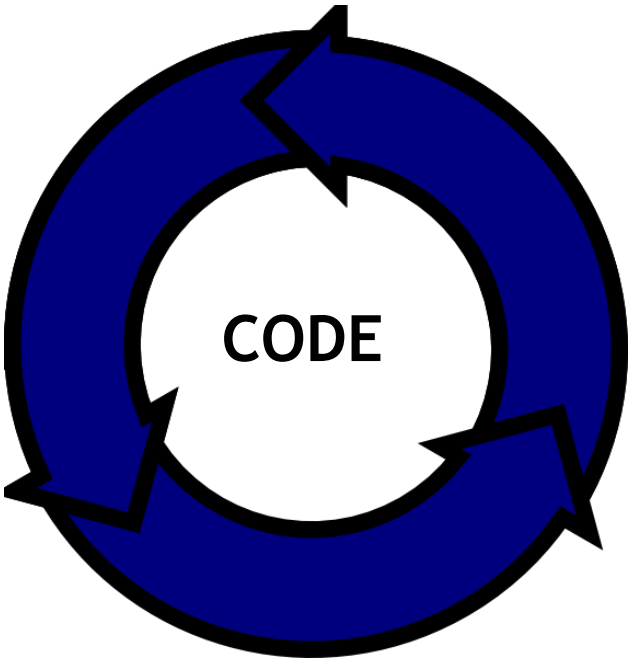
Building it Wrong

- Too many costly defects
- Poor maintainability
- Over-engineering
- Lack of confidence in the code

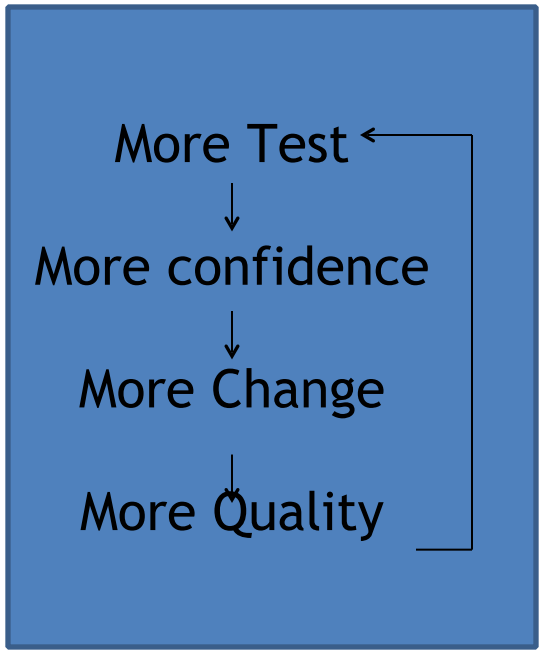
“I don’t want to touch that. It’ll take forever, and I don’t know what else will break if I do! I’ll just rewrite it.”

The Goal of TDD

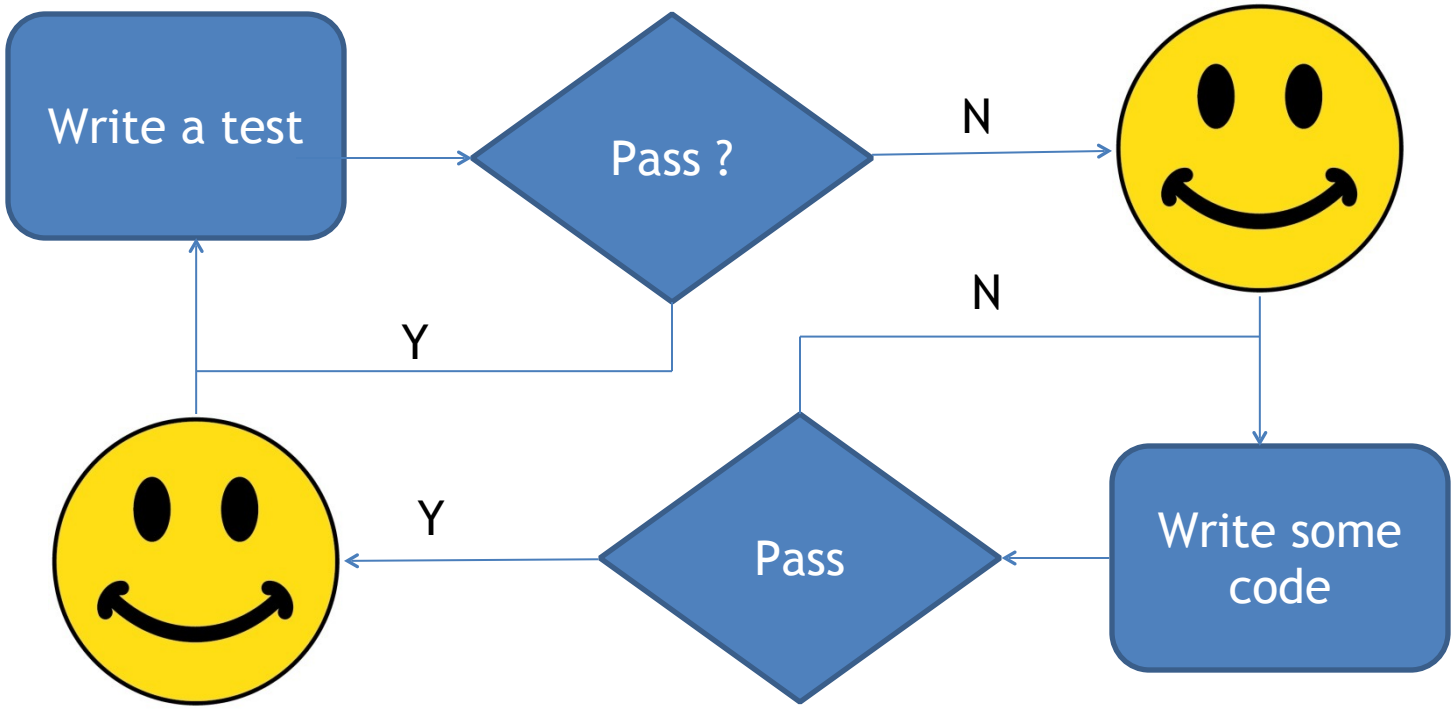
Test Harness



Virtuous cycle



Test Driven Development



Case Study

Empirical studies compared 4 projects, at Microsoft and IBM, that used TDD with similar projects that did not use TDD.

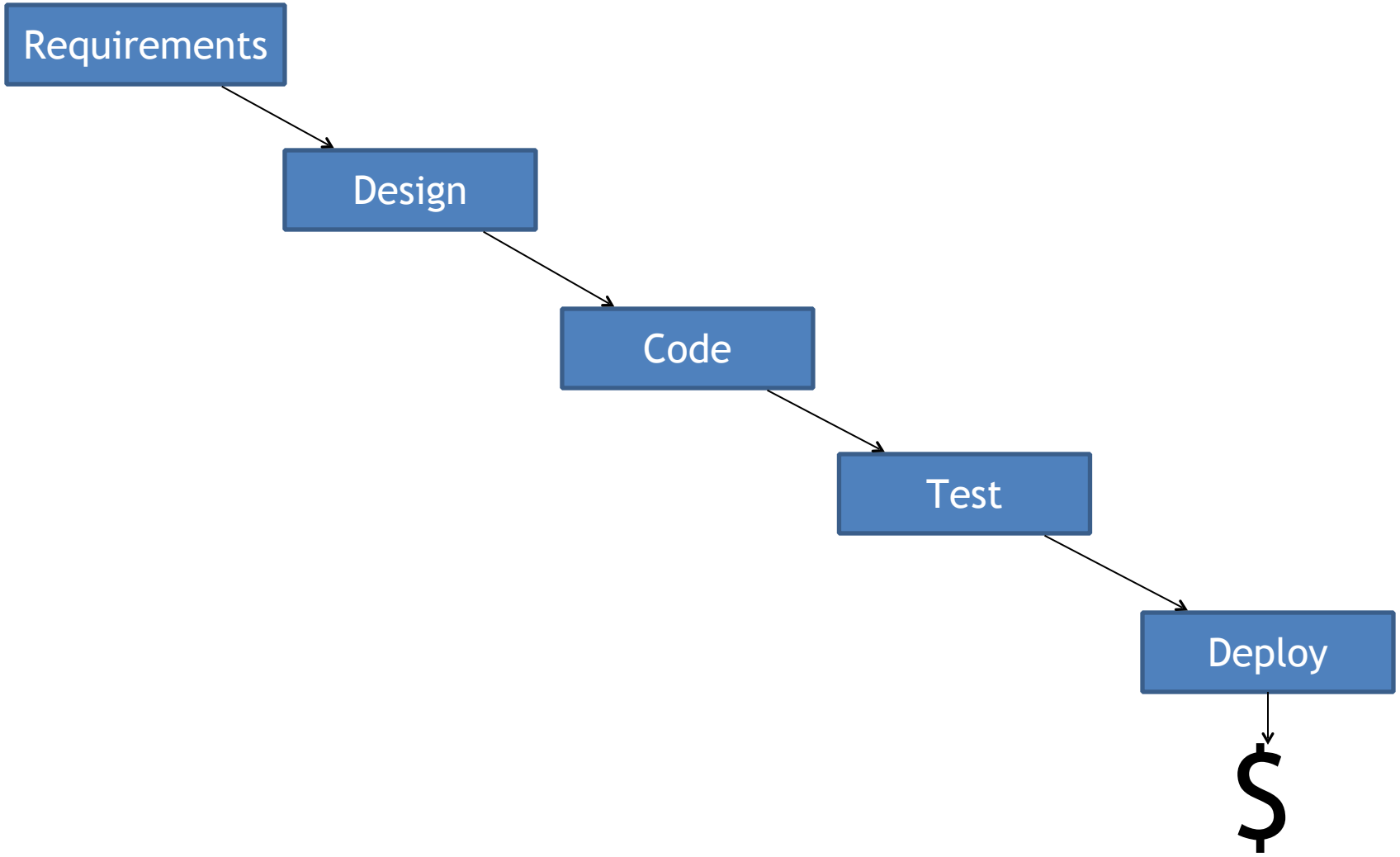
The findings were similar to other case studies

- Defect density decreased 40 % - 90 %

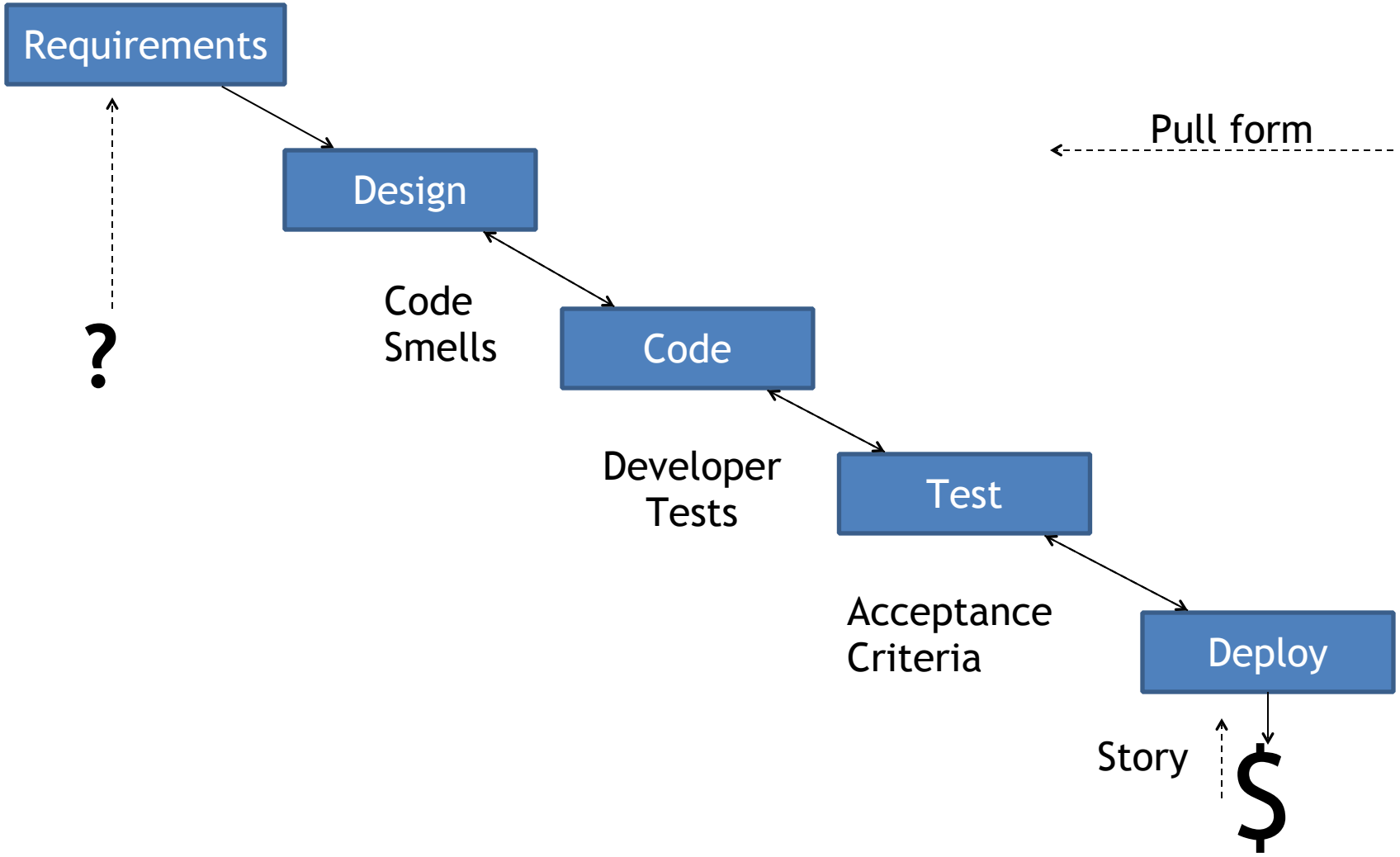
- Development time increased by 15 – 35 %

  (though teams agreed that this was offset by reduced maintenance cost)

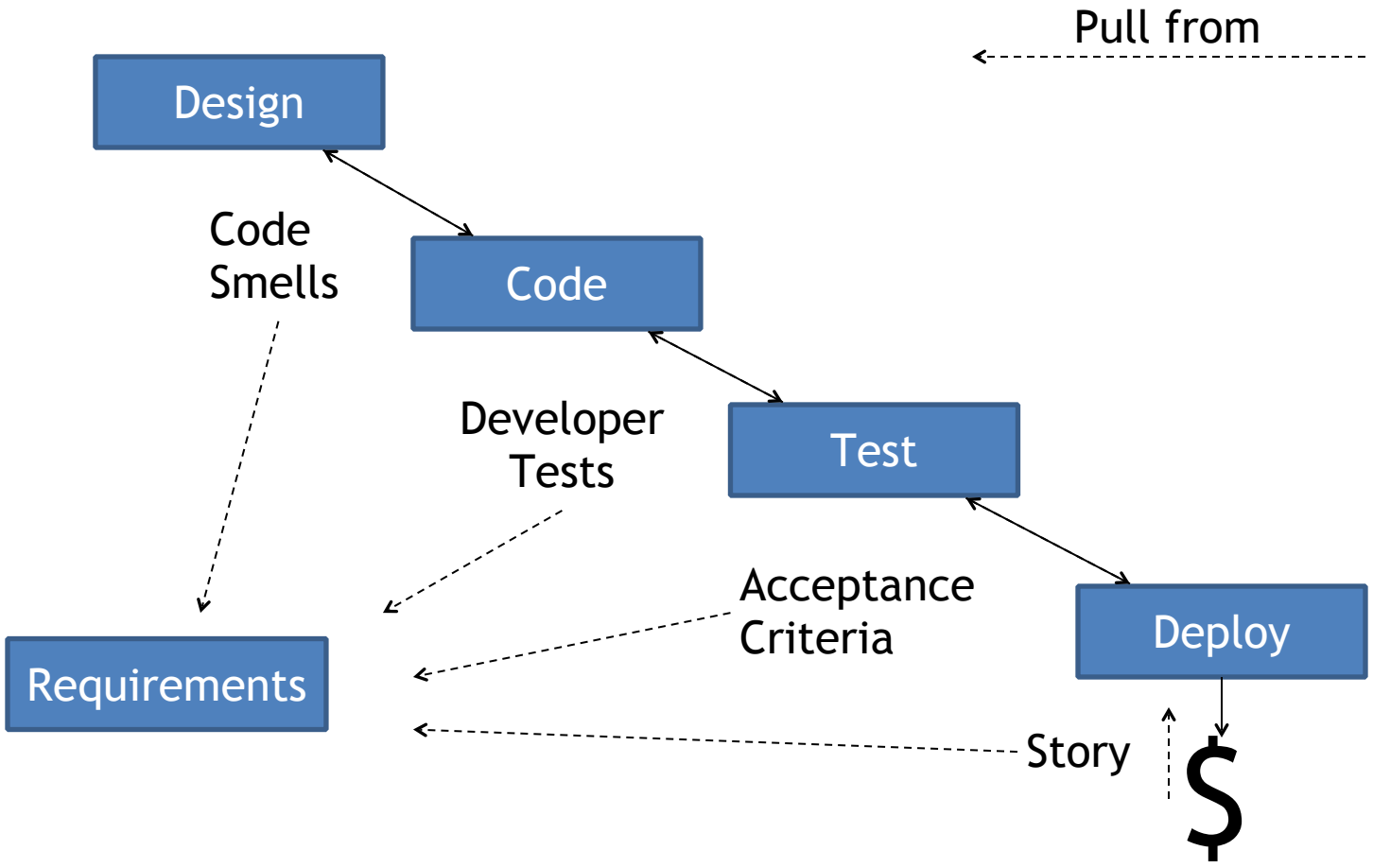
Push Model



Pull Model


Requirements
Design
Code
Test
Deploy
Pull form
?
Code
Smells
Developer
Tests
Acceptance
Criteria
Story
$

Pull Model


Pull from
Design
Code
Smells
Code
Developer
Tests
Test
Acceptance
Criteria
Deploy
Requirements
Story
$

Continuous Integration

- Maintain a single Source Repository
- Automate the build
- Make your build Self – Testing
- Everyone commits Every Day
- Every commit should Build the mainline on an integration machine
- Keep the Build Fast
- Test a Clone of the Production Environment
- Met it Easy for Anyone to Get the Latest Executable
- Everyone can see what’s happening
- Automate Deployment

For Development…

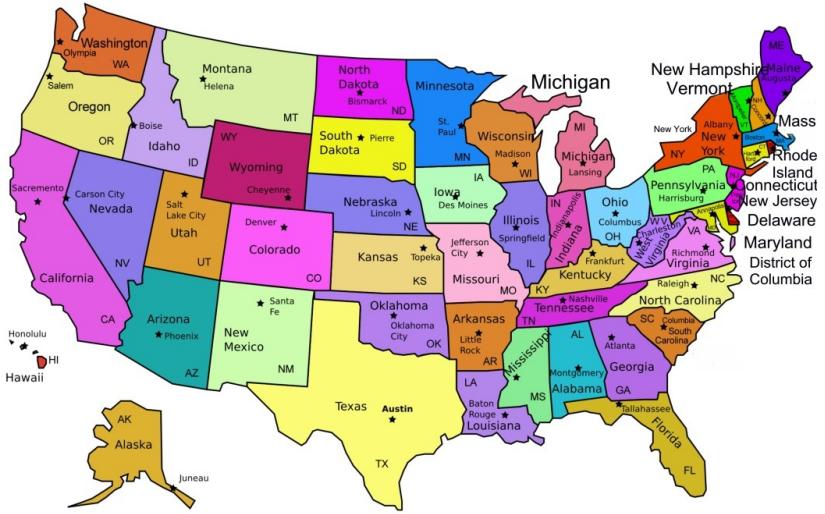
SCRUM says…	Other poplar agile practices say…
Sprint should end with a 'potentially shippable product increment'	*Take the time to build robust and maintainable systems.*
Nothing on how to achieve this	*Writing unit tests from acceptance criteria helps to ensure that development is meeting the business needs*
	TDD helps raise the quality of the code and the design so that the product increment can truly be 'shippable'
	Continuous integration provides instant feedback when a build or test fails.
	Pair Program.

Agile Scrum Immersion

Miscellany

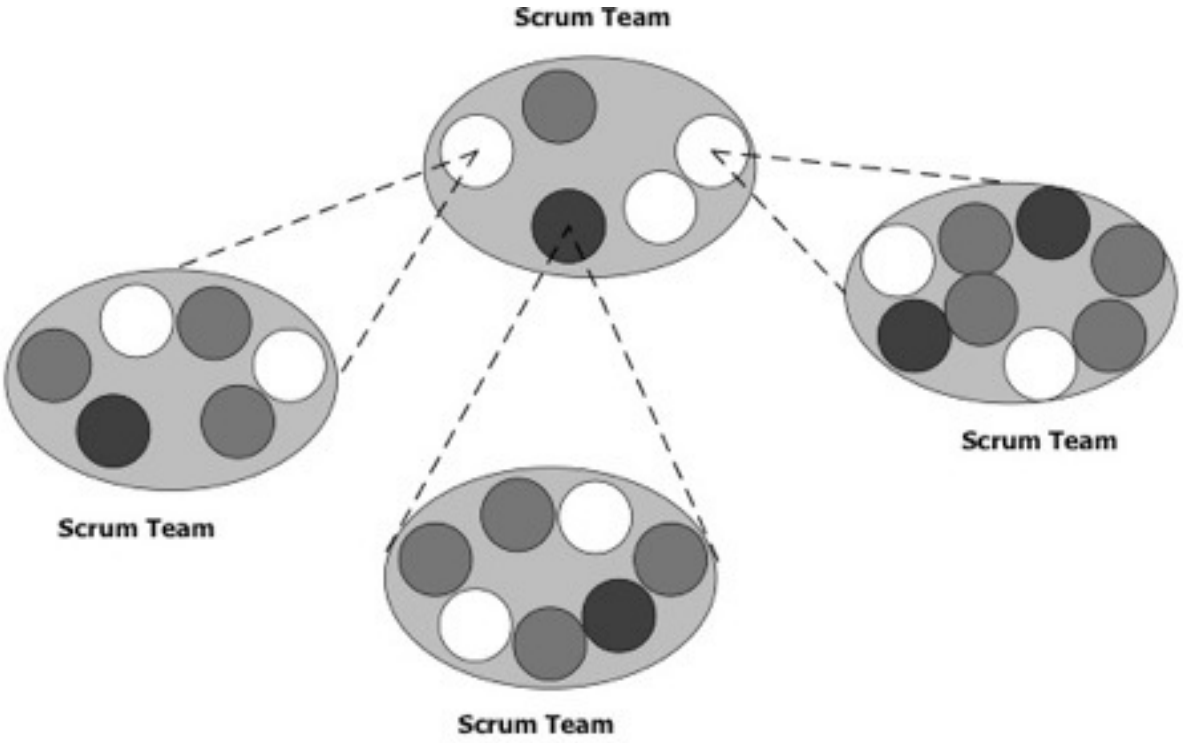
Distributed Scrum

- Technology can help simulate co-location
  - Conference Phone
  - Instant Messaging
  - Vide
  - Wiki
  - Planning Apps (37Signals Campfire)



Large Projects and Scrum

- Divide based on functionality
- Scrums of Scrums



Process Anti-Patterns

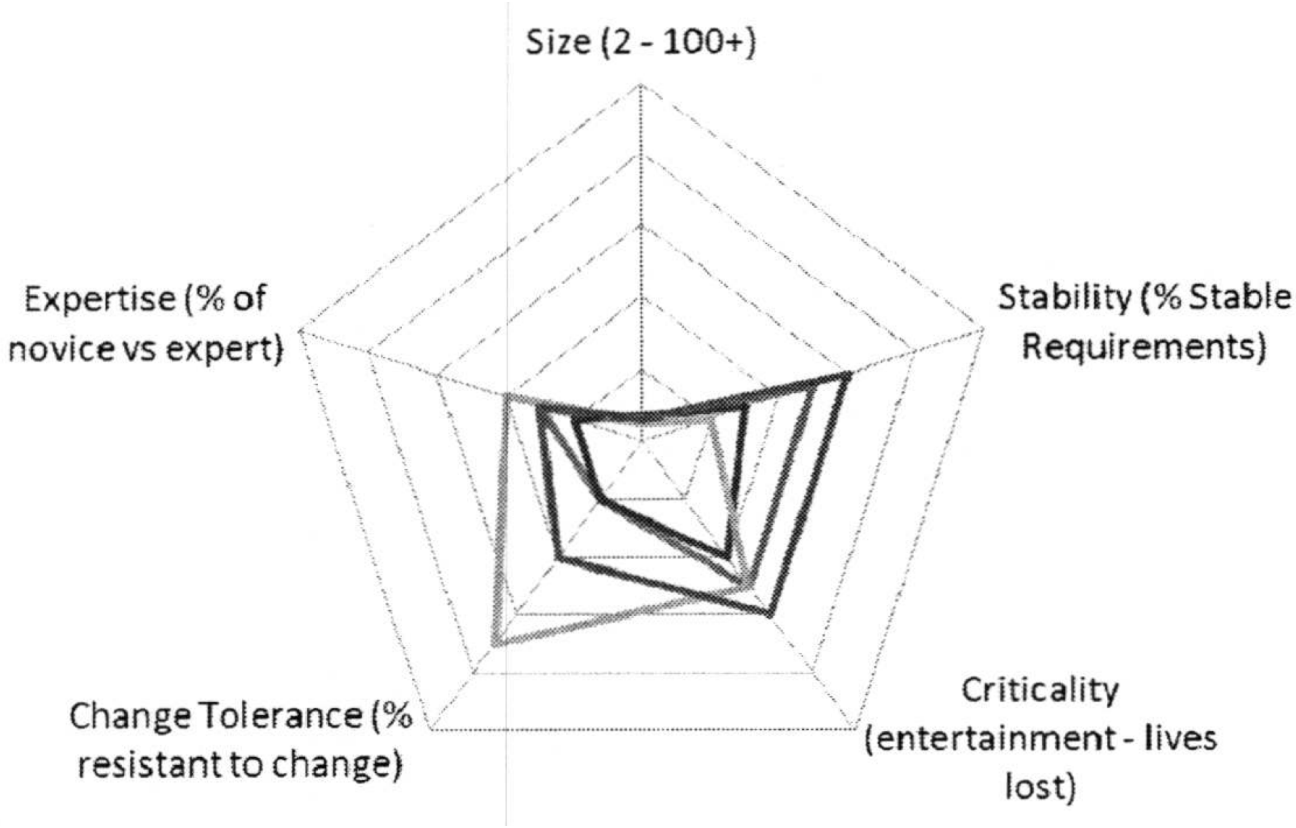
- Cargo Cults
- Tipping the scales
- Cookie Cutter

Cargo cults

Tipping the Scales

Cookie Cutter

Agile suitability



Top 10 Ways to Fail at Agile

1. Ineffective use of the retrospective
2. Inability to get everyone in the planning meetings and demos
3. Unavailable product owner, or too many product owners who can't
4. Not pulling testing into definition of "Done"
5. Obtaining only "checkbook commitments" from executive management
6. Teams lacking authority and decision making ability
7. Not having an onsite evangelist for remote locations
8. Failure to pay attention to the infrastructure required
9. A culture that does not support learning
10. Reverting to form

Things I hate about Agile…

- Product Owner keeps dropping by
- I have to show that I got work done very single day
- I am expected to demonstrate imperfect, unfinished software
- Senior management looks at my burn down chart every day
- Developers keep bugging me with questions
- Barely any down time
- Everybody else gets an opinion about my task estimates
- I am expected to work on tasks that aren't in my job description
- My project could get stopped early.


Your Agile