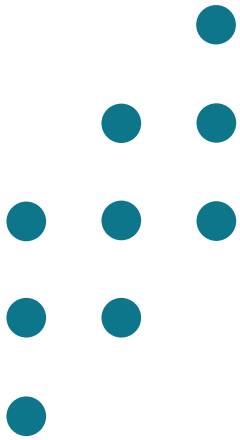
# Certified Scrum Master (CSM) Training



This workbook is designed for onsite, in-person classes. If your class is online/virtual, the workbook is optional.

Victor Bonacci
Certified Scrum Trainer
victor@rocketninesolutions.com

V4.1 (2023)

**Certifications in Scrum, the leading framework for Agile software development**


# Getting started with Scrum

## Embark on the journey of transforming your workplace with Scrum

Transitioning to Agile and Scrum requires a new mindset and overall cultural adjustments. And like all change, it doesn't come easy. But when teams and organizations fully commit to Scrum, they'll discover a new sense of flexibility, creativity, and inspiration — all of which will lead to greater results. Certified Scrum Coaches and Trainers and Scrum Alliance Registered Education Providers can help you navigate this transformative journey.

## About training

Training can mean the difference between a team that truly embraces and understands Scrum and one that is just following a management directive. Certified Scrum Trainers (CSTs) can help your team effectively make the transition.

## About coaching

Becoming more Agile with Scrum isn't something that happens overnight. To succeed, your team likely will need training and mentoring. You may need an experienced coach at your disposal to help you fully embrace Scrum.

Coaches designated with Scrum Alliance's Certified Enterprise Coach℠ (CECs) and Certified Team Coach℠ (CTCs) are experts in Scrum theory and practice. CECs and CTCs bring Scrum out of the classroom and into your world of work. Coaches help you learn how to master the new team patterns, increased collaboration, and high performance. Both CECs and CTCs understand how Scrum impacts leadership and team member responsibilities.

# Certified ScrumMaster® (CSM®) Requirements

## Requirements

- Be present for the full 2-day CSM course taught by a Certified Scrum Trainer (CST).
- After successfully completing the course, you will need to take the 50 question CSM test and answer 37 out of the 50 questions correctly within the 60-minute time limit.
- After you pass the CSM test, you will be asked to accept the CSM License Agreement and complete your Scrum Alliance membership profile.

## Test Details

- Cost of the test is included in the CSM class fee.
- Test must be completed within 90 days.
- Answers are saved as you go, so no need to worry if you lose connection.
- Test is based on the Scrum Guide (scrumguides.org) and the 2022 Learning Objectives covered in this CSM class.
- Test is “open book”.
- You will receive your score (and pass/fail) immediately upon completing the test.

## If You Don’t Pass…

- If you do not pass on your first attempt, you have one more attempt with no additional fee within the original 90-day window.
- If you do not pass on your second attempt, you may take the test again ($25 fee).

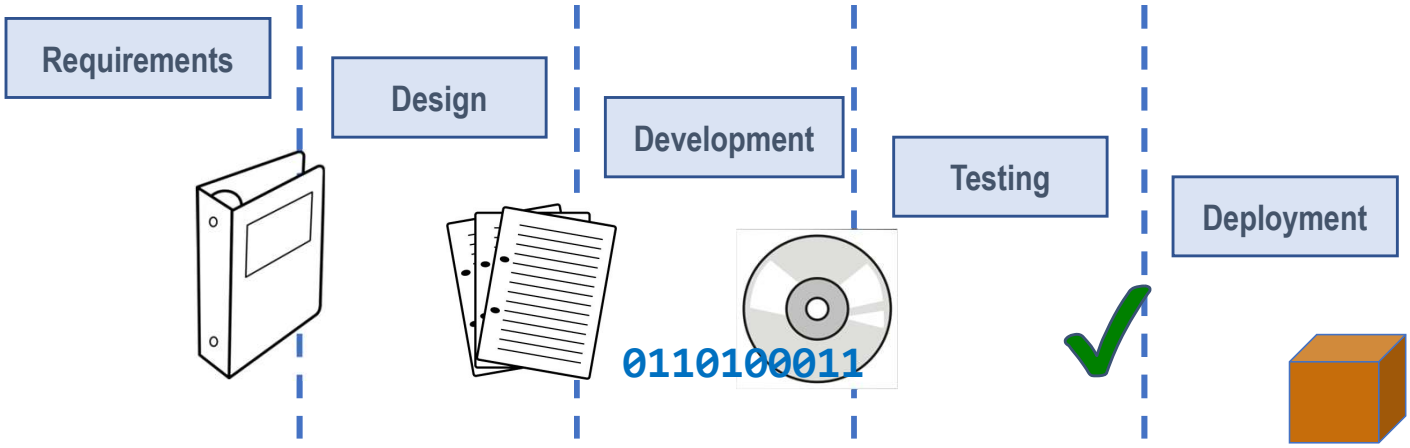
## When You Pass

- Once you pass the test, return to ScrumAlliance.org to create your profile and accept the Scrum Alliance licensing agreement (Settings > Certification Dashboard).
- Upon accepting the license, your certificate is available as a PDF.
- To renew your certification, there is a $100 fee every two years. There are no continuing education requirements for the CSM. You’ll receive an email from the Scrum Alliance when your certification is due to renew.

# Linear vs. Iterative

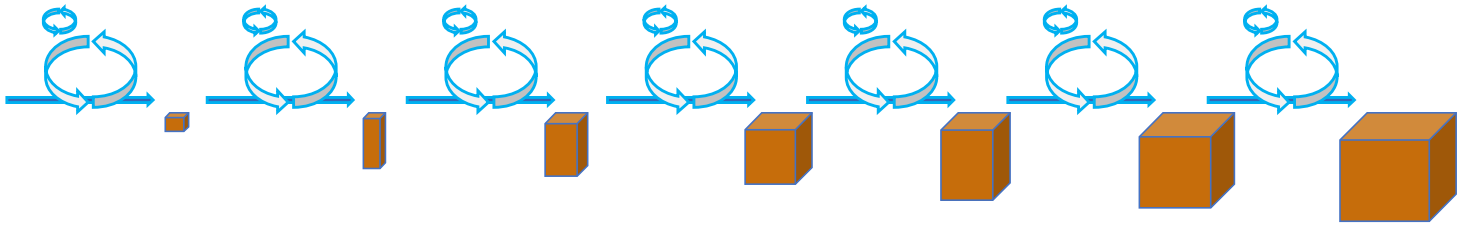
## Waterfall (Linear, fixed)

Traditional project management encouraged creating a “big plan up front” that would cascade the work from one working group to the next. Large work items were discouraged from traveling back upstream for rework or feedback. The flow of activity was linear, and the work advanced slowly.




## Agile/Scrum (Iterative, incremental)

Agile projects emphasize an iterative approach with many built-in opportunities for course-correction and empirical feedback. The flow of work though the system is much faster as the work increments are small. Teams are able to adapt quickly to change.



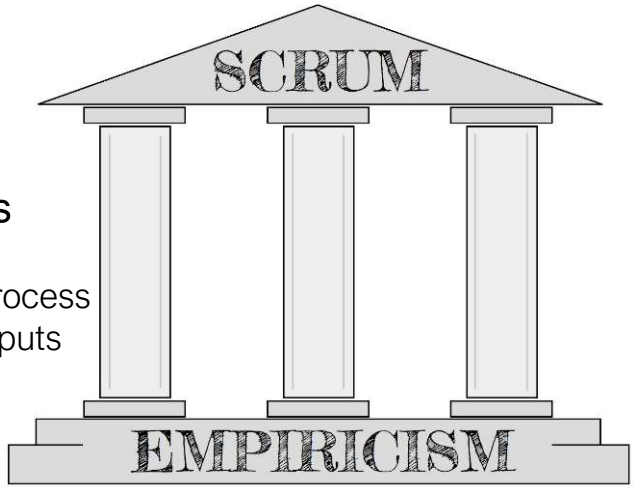
# Scrum and Empiricism

Scrum is supported by the three pillars of empirical process control.

Empirical Process *vs* Defined Process

- Variable Inputs
- Adaptable Process
- Variable Outputs
- Plan-Do-Study-Act

- Known Inputs
- Repeatable Process
- Expected Outputs
- Assumptions




## Build the Tallest Freestanding Structure:

The winning team is the one that has the tallest structure measured from the tabletop surface to the top of the marshmallow. That means the structure ***cannot*** be suspended from a higher structure, like a chair, ceiling or chandelier.

## The Entire Marshmallow must be on top:

The entire marshmallow needs to be on the top of the structure. Cutting or eating part of the marshmallow ***disqualifies*** the team.

## Use as Much or as Little of the Kit:

The team can use any amount of their 20 spaghetti sticks, and as much or as little of the string or tape.

## Break up the Spaghetti, String or Tape:

Teams are free to break the spaghetti, cut up the tape and string to create new structures.

## The Challenge Lasts 18 minutes:

Teams ***cannot*** hold on to or touch the structure when the time runs out. Those touching or supporting the structure at the end of the exercise will be ***disqualified***.

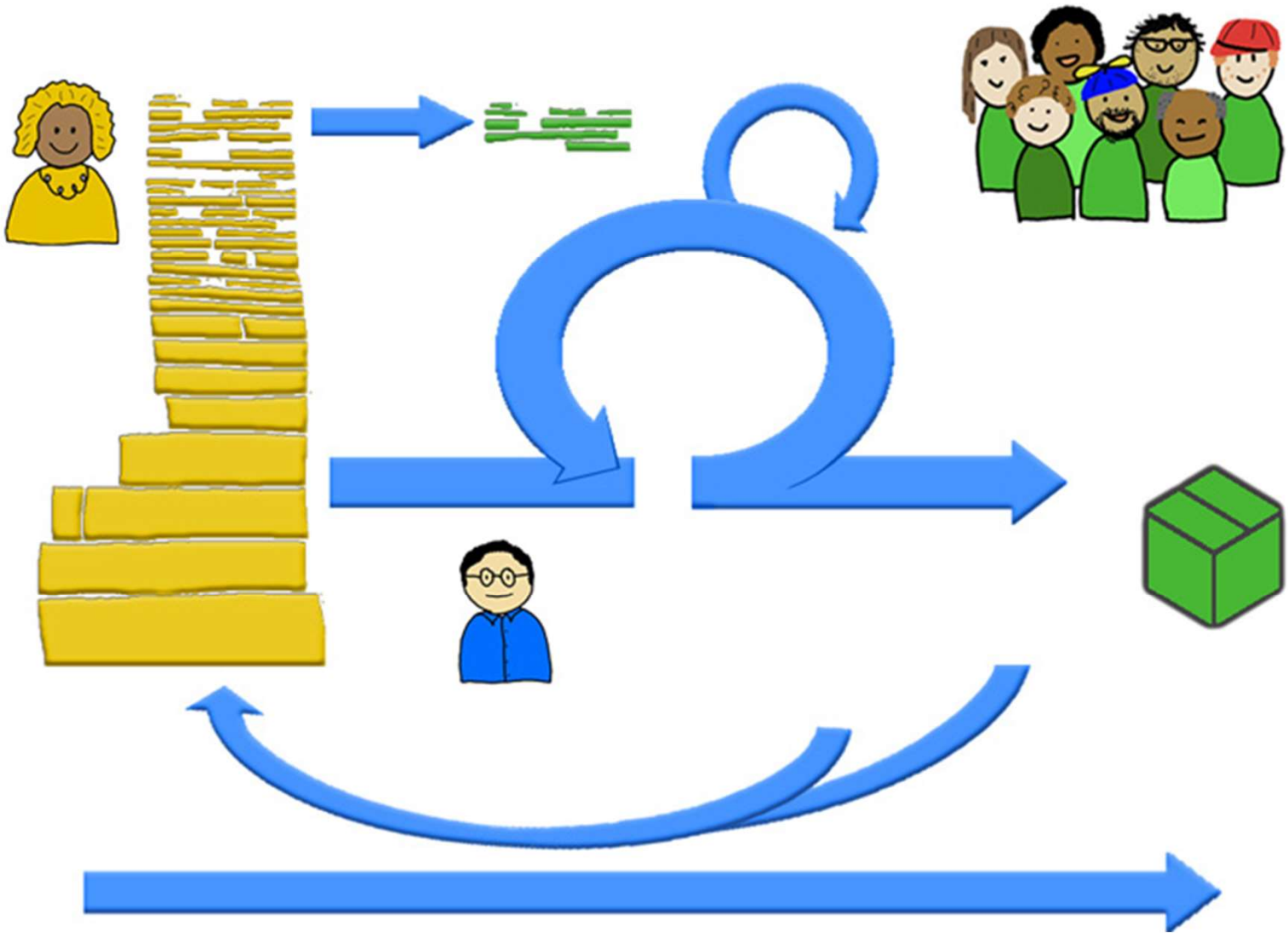
# 12 Principles of the Agile Manifesto

Instructions: As a table group, create a ***#hashtag*** for each principle below.

| | |
|---|---|
| Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. | |
| Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. | |
| Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. | |
| Business people and developers must work together daily throughout the project. | |
| Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. | |
| The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. | |
| Working software is the primary measure of progress. | |
| Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. | |
| Continuous attention to technical excellence and good design enhances agility. | |
| Simplicity -- the art of maximizing the amount of work not done -- is essential. | |
| The best architectures, requirements, and designs emerge from self-organizing teams. | |
| At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. | |

# Scrum Framework

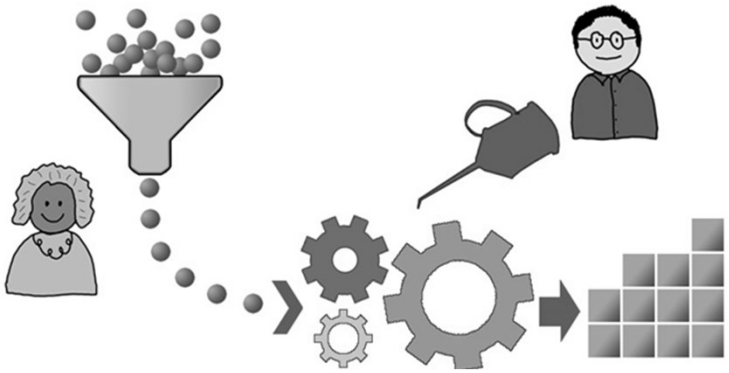
Label the following diagram:



*Scrum is _____ (to understand),*
*but it is not _____ (to implement).*

| 3 Artifacts: | 4 Meetings: | 3 Roles: |
|---|---|---|
| _____ | _____ | _____ |
| _____ | _____ | _____ |
| _____ | _____ | _____ |
| | _____ | |

# Developers



- Self _____
- Cross _____
- Dedicated
- responsible for Q_____
- follows the values
- Size = ____
- owns the HOW

# *DING!!* or *Dud…*

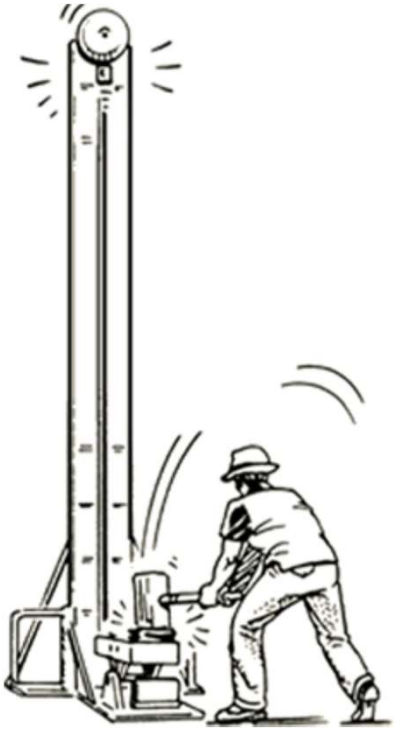
For each statement below, choose either ***Ding*** or ***Dud***:

Developers are assigned to only one team at a time.

DING!! Dud…

Developers give the Product Owner status updates at the Daily Scrum.

DING!! Dud…



Developers provide estimates to Product Backlog items.

DING!! Dud…

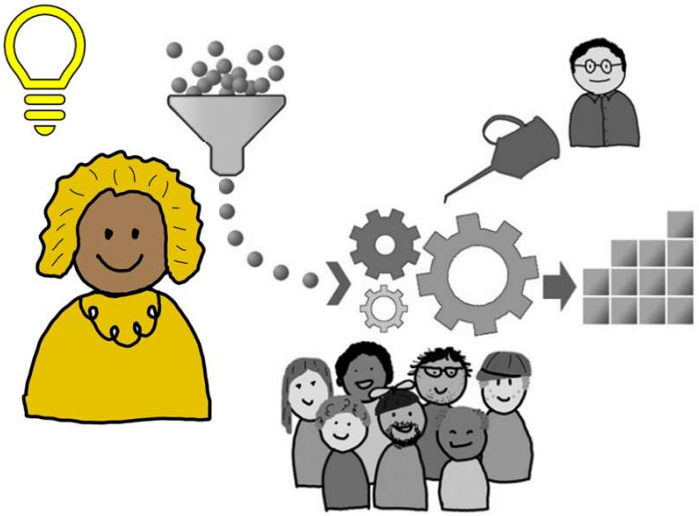
Stakeholders manage the Scrum Team's day-to-day activities.

DING!! Dud…

The Developers are responsible for delivering a high-quality product that can change over time.

DING!! Dud…

# Product Owner



- must have _____ (not be a proxy)
- must have _____ of the domain
- must be _____ to the team
- owns the WHAT

**Draw a line** from the attribute or responsibility to each of the Scrum Roles.
Each attribute / responsibility can be associated with **one or more** roles.

Scrum Master

Developers

Product Owner

Shows only completed functionality.

Participates in the Daily Scrum.

Commits to the Sprint Goal.

Acts as a servant leader.

Provides the voice-of-the-customer.

Protects the team.

Ensures value delivery.

Owns the Product Backlog.

Turns backlog items into potentially-shippable product increments.

Makes sure the team adheres to the values, principles and practices of Scrum.

Determines how long Product Backlog items will take to implement.

# Product Backlog and its Items

Product Backlog should be **DEEP**:

D _____

E _____

E _____

P _____

Examples of PBIs include:

- User Story
- Epic
- Spike
- NFR _____
- Defect (Bug)
- Experiment / Improvement

## The 3 Cs of User Stories

Write on _____ and annotate with notes, estimates, etc.

Details comes from _____ with product owner.

Acceptance tests _____ that the story was coded correctly.

## A User Story Template

As a _____, I want to _____ so that _____.

**Who** **What** **Why**

*Any story template is a suggestion, not a rule.*

Implied Order

| Title | Description | Est | Acceptance Criteria |
|---|---|---|---|
| Toast pops up | When the timer is done, the toast pops up so it doesn't get burned | 5 | • Can set more or less time<br>• When timer is done, spring releases |
| *Qwerty* | *Ehiehpep pirpihrh pirh* | *8* | *Youb ob inhdff…* |
| *lorem ipsum* | *gpodawund covfefe* | *3* | *Oiua rwt khrgqw rtwer uo…* |

# Time Boxes


Match the sprint length to the recommended maximum time-box for Sprint Planning:

| sprint length | time-box (max) |
|---|---|
| 1 week | 4 hours |
| 2 weeks | 8 hours |
| 3 weeks | 6 hours |
| 4 weeks | 2 hours |

# Defining Ready and Done

Example **Definition of Ready (DOR)**

- Story defined with ***acceptance criteria***
- Parent epic identified
- ***Sized by team (developers)***, can be Done in under 3 days
- Team determines Story is INVESTed
- ***Dependencies identified*** and accepted
- Architectural stories are completed
- SMEs/Functional ***experts identified***
- ***API Contracts reviewed*** with developer
- UX artifacts created, ***mockup attached***

Example **Definition of Done (DOD)**

- ***Unit tests created***, checked in, all passing
- Code ***checked in*** and ***builds successfully*** on integration environment
- Automated unit ***test coverage*** > 80%
- All ***acceptance criteria*** pass
- Peer ***code review*** complete
- No open defects
- Needed documentation is complete
- Product owner has reviewed and accepted story

# Sprint Planning

## Characteristics of a Sprint Backlog

- Real-time snapshot of what is being worked on to accomplish the Sprint Goal
- Development team can modify at any time
- Product Owner can remove Product Backlog Items
- Just enough detail and highly visible view of tasks
- Tool for the team to manage itself during the Sprint
- Used to turn Product Backlog Items into potentially shippable functionality

# The Sprint Goal

*Select the phrases from the right to fill in the blanks on the left.*

The Sprint Goal is an objective set for the Sprint that provides guidance to the _____ on why it is building the Increment. This goal is created during the _____ meeting. The _____ gives the Development Team some flexibility regarding the functionality implemented within the Sprint.

***Sprint Goal***

***Developers***

***Sprint Planning***

Why does the Sprint Goal not change once the Sprint begins?

_____

_____

At your table, brainstorm an example Sprint Goal:

_____

Exercise: Fill in the blanks and discuss with your tablemates:

# Sprint Planning and the Sprint Backlog

*fill in the blanks by using the words & phrases in the bottom box*

1. The _____ and the _____ are two outputs of Sprint Planning.
2. During Sprint Planning, the _____ topic is facilitated by the Product Owner, while the _____ topic is owned by the Development Team.
3. Benefits of having a Sprint Goal include _____ and _____.
4. The ideal sprint backlog _____, _____ and _____.
5. Only the _____ can remove selected product backlog items (PBIs) and thus modify the sprint backlog indirectly.
6. If the Scrum Team disregards some elements of sprint planning, the Development Team is at risk of _____ or _____.
7. One input to Sprint Planning is _____.

| | | |
|---|---|---|
| past performance of the Development Team | HOW | sprint backlog |
| alignment among the Development Team | Product Owner | sprint goal |
| a better context for the work | has just enough detail | is highly visible |
| provides a real-time snapshot of progress | overcommitting | WHAT |
| not understanding the work of the Sprint | | |

# Serving the Developers

Identify three scenarios where the Scrum Master may serve the Developers:

_____

_____

_____

What can possibly go wrong if the Product Owner or a stakeholder applies excessive time pressure on the Developers?

| consequences | | remedies |
|---|---|---|
| _____ |  | _____ |

Technical Debt _____

Tech Debt can be caused by:

_____

_____

_____

The accumulation of Tech Debt can:

# Technical Practices

List some development practices that will help Scrum Teams deliver a high-quality product Increment and reduce technical debt each Sprint:

_____ _____ _____

_____ _____ _____

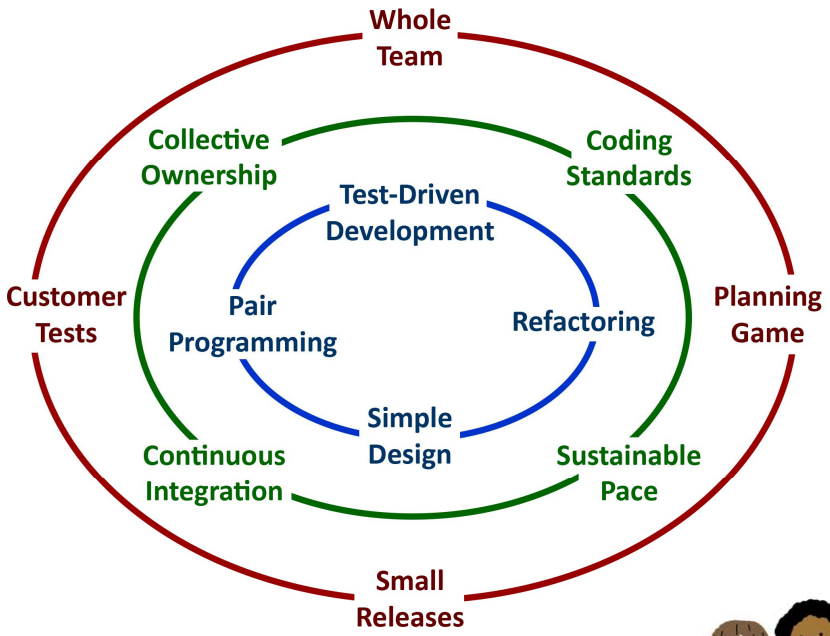
How do these practices impact the ScrumTeam's ability to deliver a potentially releasable Increment each Sprint?

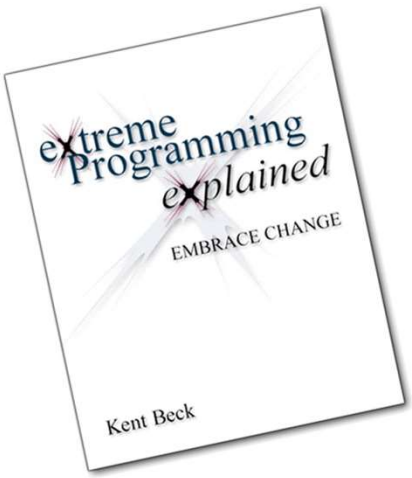
_____

_____

_____

# eXtreme Programming (XP)



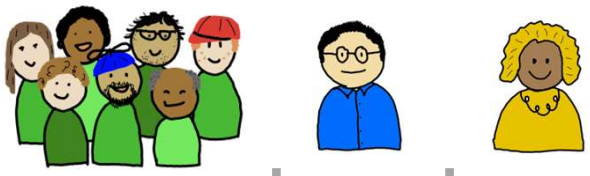

# Daily Scrum

How does the Daily Scrum differ from a status meeting?

How do the various constraints support the team?

Tactics to run the Daily Scrum within the time-box:

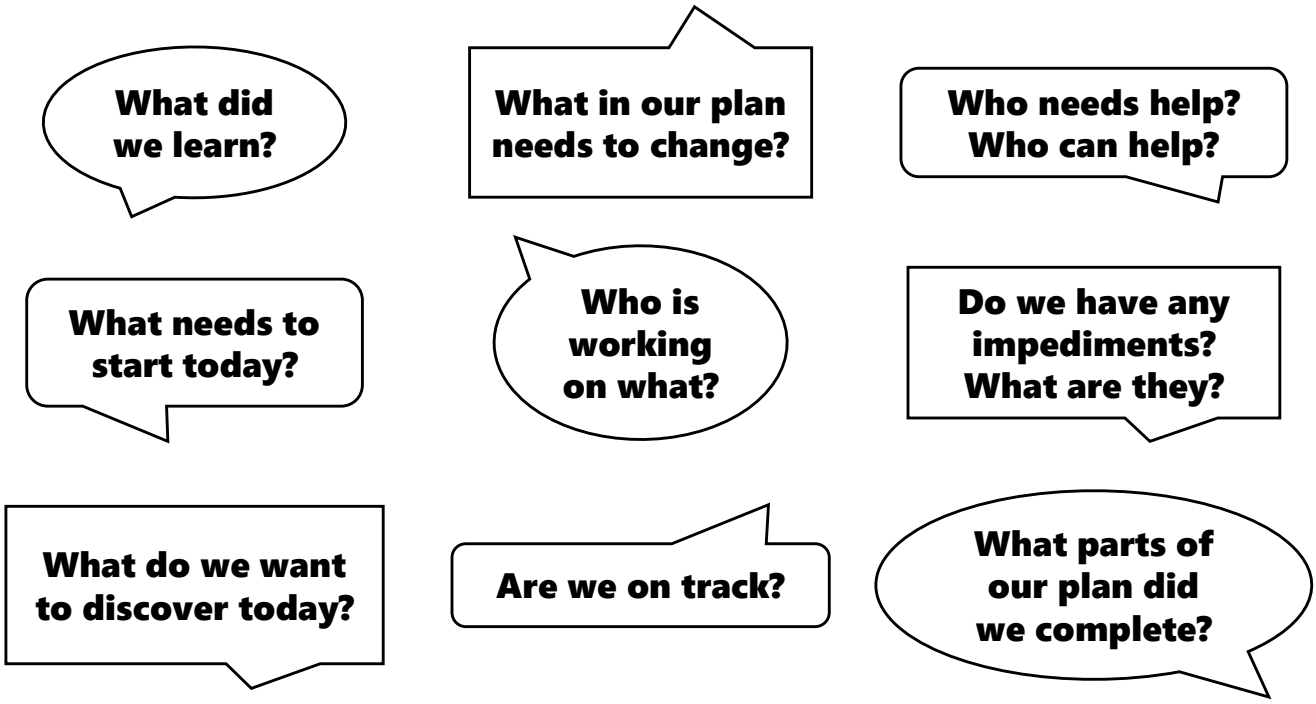
## Scrum Team Responsibilities at the Daily Scrum



| | | | |
|---|---|---|---|
| Answer the three questions | | | |
| Teach how to do a daily scrum | | | |
| Review progress toward the sprint goal | | | |
| Provide clarification | | | |
| Offer observations, not solutions | | | |
| Offer early feedback | | | |
| Update the sprint backlog | | | |
| Facilitate conversation if necessary | | | |

# Daily Scrum

Some alternatives to the standard three questions might include:

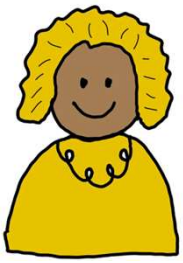



What is your favorite question to answer?
Share with your table mates.


- ❑ By the Book
- ❑ Has No Info
- ❑ Late-comer
- ❑ The Solver
- ❑ Remote Member
- ❑ Off-topic Chatty
- ❑ Too Much Detail
- ❑ ScrumMaster's Pet
- ❑ Ignores Priorities
- ❑ Won't Ask for Help
- ❑ Too Vague
- ❑ Distracted by Phone

# Responsibilities at Sprint Review

*Connect the roles to their responsibilities*

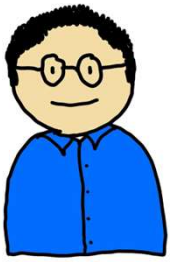


Accept or reject items (prior to meeting)

Ensure that the review happens

Demonstrate the results of the Sprint

Respond to questions

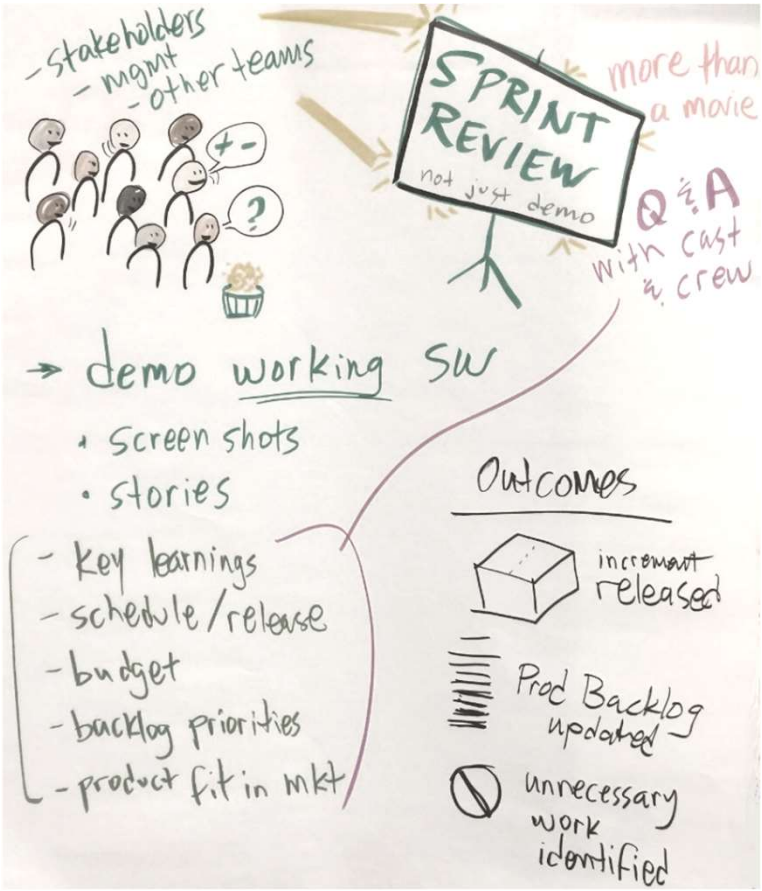


Solicit and process feedback

Maintain the time box

## Stakeholders

- Provide feedback
- Help resolve impediments



# Sprint Review & Retrospective

*What are the recommended maximum time-boxes for a **one-month** Sprint?* (circle the answer)

| Sprint Review | Retrospective |
|---|---|
| **1 hour** | **1 hour** |
| **4 hours** | **2 hours** |
| **8 hours** | **3 hours** |

How will a Scrum Team will inspect & adapt and increase transparency at each of the following Scrum events?

Daily Scrum

_____

_____

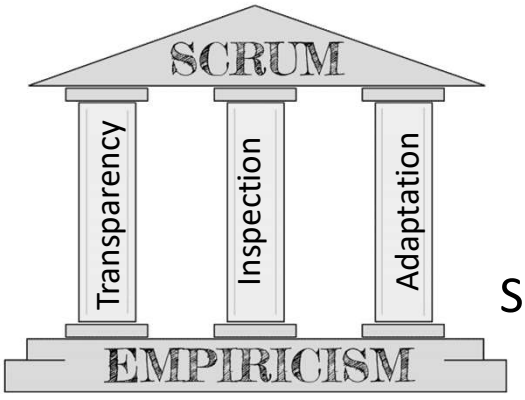
_____

Sprint Planning

_____

_____

_____




Sprint Review

_____

_____

_____

Sprint Retrospective

_____

_____

_____

# Retrospectives

***Mix up your approach***. Find hundreds of variations online:

Retro Mat – *retromat.org*

Fun Retrospectives – *funretrospectives.com*

Tasty Cupcakes (games & activities) – *tastycupcakes.org*

Lean Coffee – *leancoffee.org*

# *DING!!* or *Dud…*

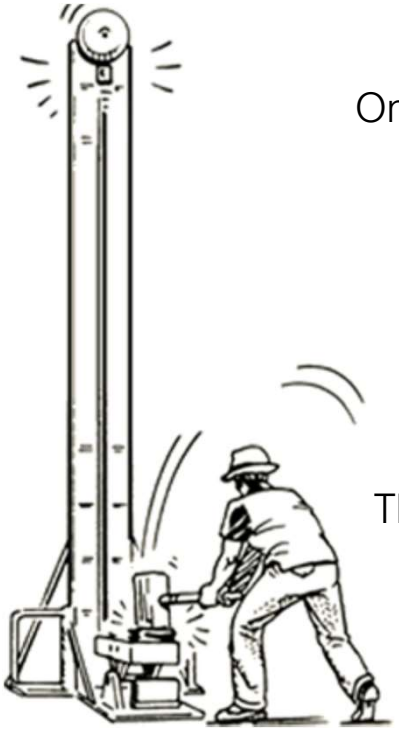
For each statement below, choose either ***Ding*** or ***Dud***:

The Daily Scrum serves the same purpose as a status meeting.

DING!! Dud…

The Scrum Team may share lessons learned at the Sprint Review.

DING!! Dud…



One outcome of the Sprint Review is an updated Sprint Backlog.

DING!! Dud…

The Product Owner does not attend the Retrospective.

DING!! Dud…

The Scrum Master determines what the Developers will discuss and improve on at the Retrospective.

DING!! Dud…

# HOW TO SPLIT A USER STORY

## 1 PREPARE THE INPUT STORY

Does the big story satisfy INVEST* (except, perhaps, small)?

NO → Combine it with another story or otherwise reformulate it to get a good, if large, starting story.

YES → Is the story size 1/10 to 1/5 of your velocity?

- You're done.
- Continue. You need to split it.

## 2 APPLY THE SPLITTING PATTERNS

start here

### WORKFLOW STEPS

Does the story describe a workflow?

- Can you take a thin slice through the workflow first and enhance it with more stories later?
- Can you split the story so you do the beginning and end of the workflow first and enhance with stories from the middle of the workflow?

### OPERATIONS

Does the story include multiple operations? (e.g. is it about "managing" or "configuring" something?)

- Can you split the operations into separate stories?

### BUSINESS RULE VARIATIONS

Does the story have a variety of business rules? (e.g. is there a domain term in the story like "flexible dates" that suggests several variations?)

- Can you split the story so you do a subset of the rules first and enhance with additional rules later?

### DEFER PERFORMANCE

Does the story get much of its complexity from satisfying non-functional requirements like performance?

- Could you split the story to just make it work first and then enhance it to satisfy the non-functional requirement?

### VARIATIONS IN DATA

Does the story do the same thing to different kinds of data?

- Can you split the story to process one kind of data first and enhance with the other kinds later?

### SIMPLE/COMPLEX

Does the story have a simple core that provides most of the value and/or learning?

- Could you split the story to do that simple core first and enhance it with later stories?

### MAJOR EFFORT

When you apply the obvious split, is whichever story you do first the most difficult?

- Could you group the later stories and defer the decision about which story comes first?

### INTERFACE VARIATIONS

Does the story have a complex interface?

- Is there a simple version you could do first?

Does the story get the same kind of data via multiple interfaces?

- Can you split the story to handle data from one interface first and enhance with the others later?

last resort

### BREAK OUT A SPIKE

Are you still baffled about how to split the story?

- Can you find a small piece you understand well enough to start?
  - Write that story first, build it, and start again at the top of this process.
- Can you define the 1-3 questions most holding you back?
  - Write a spike with those questions, do the minimum to answer them, and start again at the top of this process
  - Take a break and try again.

## 3 EVALUATE THE SPLIT

Are the new stories roughly equal in size?

NO → Try another pattern on the original story or the larger post-split stories.

YES → Is each story about 1/10 to 1/5 of your velocity?

- Do each of the stories satisfy INVEST?
  - Try another pattern.
- Are there stories you can deprioritize or delete?
  - Try another pattern. You probably have waste in each of your stories.
- Is there an obvious story to start with that gets you early value, learning, risk mitigation, etc.?
  - Try another pattern to see if you can get this.
- You're done, though you could try another pattern to see if it works better.

* INVEST - Stories should be:
Independent
Negotiable
Valuable
Estimable
Small
Testable

AGILEFORALL
www.agileforall.com

Visit http://www.richardlawrence.info/splitting-user-stories/ for more info on the story splitting patterns

Copyright © 2011-2018 Agile For All. All rights reserved.

Last updated 2/21/2018