

# Agile: Theory Meets Practice

Understanding Agile Values & Principles: Using Them Practically in Any Situation

Workshop Presented by Kimberle Seale

# Contents

Objective: .....	3
Overview: .....	3
Key Values and Principles .....	4
Agile Values: .....	4
12 Agile Principles: .....	5
Individuals and Interactions over Process and Tools .....	6
Exercise: Describe the “Day in the Life” of your style.....	6
Agile Methodology.....	7
Benefits: .....	7
Agile Methods .....	8
Meet the Team .....	9
Exercise: Select a Team Name & Assign Your Roles:.....	9
Timeline Focus .....	10
Exercise: Plan Your Time .....	11
User Stories .....	12
Product Backlog .....	14
Exercise: Brainstorm your Product Backlog .....	14
Exercise: Create Mockups and Acceptance Tests .....	15
Personas.....	16
Exercise: Create 2 personas for your product.....	16
Estimating .....	17
Implement Your Product.....	18
Exercise: Using the work you have already done, implement your product.....	18
Best Practices .....	19
Daily Scrum.....	19
Template .....	20
Retrospective .....	20
Glossary:.....	21

## Objective:

To understand the fundamentals of Agile so that you can apply and practice them on any technology project.

## Overview:

Notes:

---

---

---

---

---

---

# Key Values and Principles

## Agile Values:

### **1. Individuals and Interactions Over Processes and Tools**

“Individuals and interactions over processes and tools.” Valuing people more highly than processes or tools is easy to understand because it is the people who respond to business needs and drive the development process. If the process or the tools drive development, the team is less responsive to change and less likely to meet customer needs.

### **2. Working Software Over Comprehensive Documentation**

Historically, enormous amounts of time were spent on documenting the product for development and ultimate delivery. Technical specifications, technical requirements, technical prospectus, interface design documents, test plans, documentation plans, and approvals required for each. The list was extensive and was a cause for the long delays in development. Agile does not eliminate documentation, but it streamlines it in a form that gives the developer what is needed to do the work without getting bogged down in minutiae. Agile documents requirements as user stories, which are sufficient for a software developer to begin the task of building a new function.

### **3. Customer Collaboration Over Contract Negotiation**

Negotiation is the period when the customer and the product manager work out the details of a delivery, with points along the way where the details may be renegotiated. Collaboration is a different creature entirely. The Agile Manifesto describes a customer who is engaged and collaborates throughout the development process. This makes it far easier for development to meet their needs of the customer.

### **4. Responding to Change Over Following a Plan**

Traditional software development regarded change as an expense, so it was to be avoided. The intention was to develop detailed, elaborate plans, with a defined set of features and with everything, generally, having as high a priority as everything else, and with a large number of many dependencies on delivering in a certain order so that the team can work on the next piece of the puzzle.

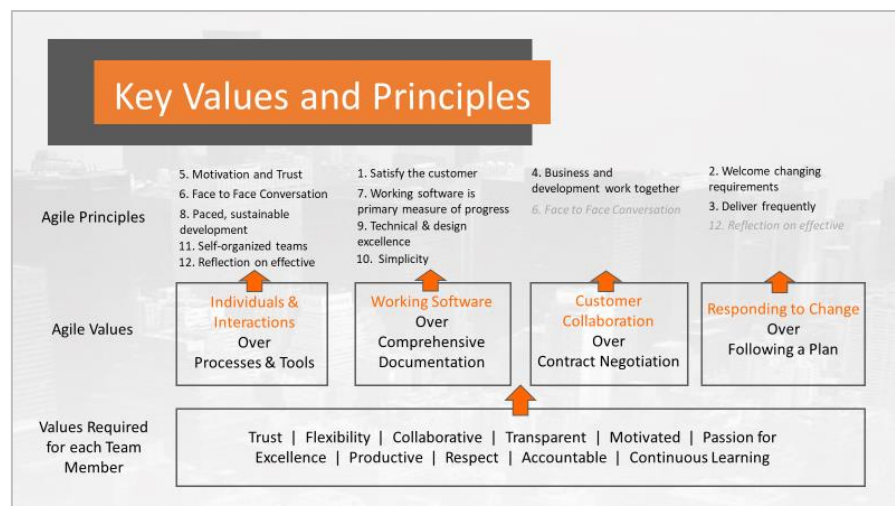
## 12 Agile Principles:

The Twelve Principles are the guiding principles for the methodologies that are included under the title “The Agile Movement.” They describe a culture in which change is welcome, and the customer is the focus of the work. They also demonstrate the movement’s intent as described by Alistair Cockburn, one of the signatories to the Agile Manifesto, which is to bring development into alignment with business needs.

The twelve principles of agile development include:

1. **Customer satisfaction through early and continuous software delivery**
2. **Accommodate changing requirements throughout the development process**
3. **Frequent delivery of working software**
4. **Collaboration between the business stakeholders and developers throughout the project**
5. **Support, trust, and motivate the people involved**
6. **Enable face-to-face interactions**
7. **Working software is the primary measure of progress**
8. **Agile processes to support a consistent development pace**
9. **Attention to technical detail and design enhances agility**
10. **Simplicity**
11. **Self-organizing teams encourage great architectures, requirements, and designs**
12. **Regular reflections on how to become more effective**

The intention of Agile is to align development with business needs, and the success of Agile is apparent. Agile projects are customer focused and encourage customer guidance and participation. As a result, Agile has grown to be an overarching view of software development throughout the software industry and an industry all by itself.



## Individuals and Interactions over Process and Tools

Exercise: Describe the “Day in the Life” of your style. What would be your motto or theme song?

---

---

---

---

---

---

---

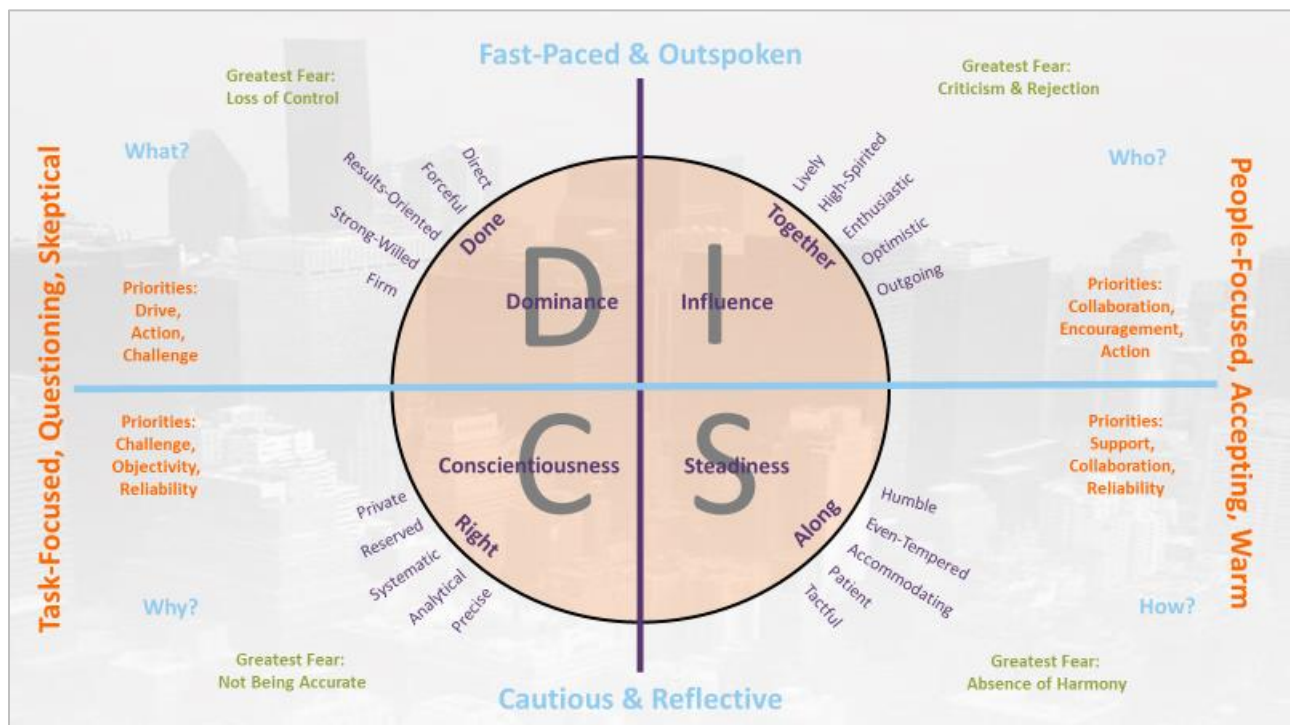
---

---

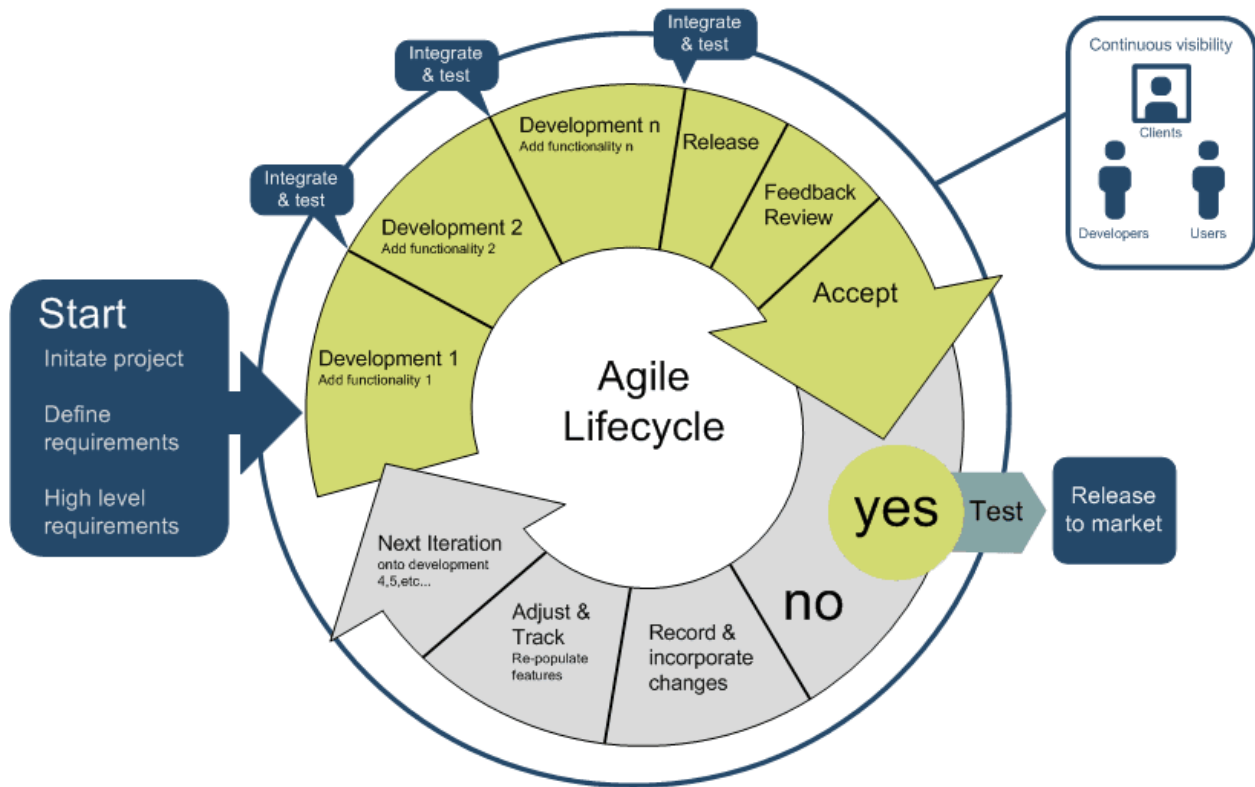
---

Below is a tool you can use to figure out your teammate’s personality style. Determine these questions about them:

1. Are they Fact-Paced & Outspoken OR Cautious & Reflective?
2. Are they People-Focused, Accepting and Warm OR Task-Focused, Questioning & Skeptical?
3. Based on those two answers, what is their DiSC? You then have a good starting point for what their priorities, greatest fears and motivators are.



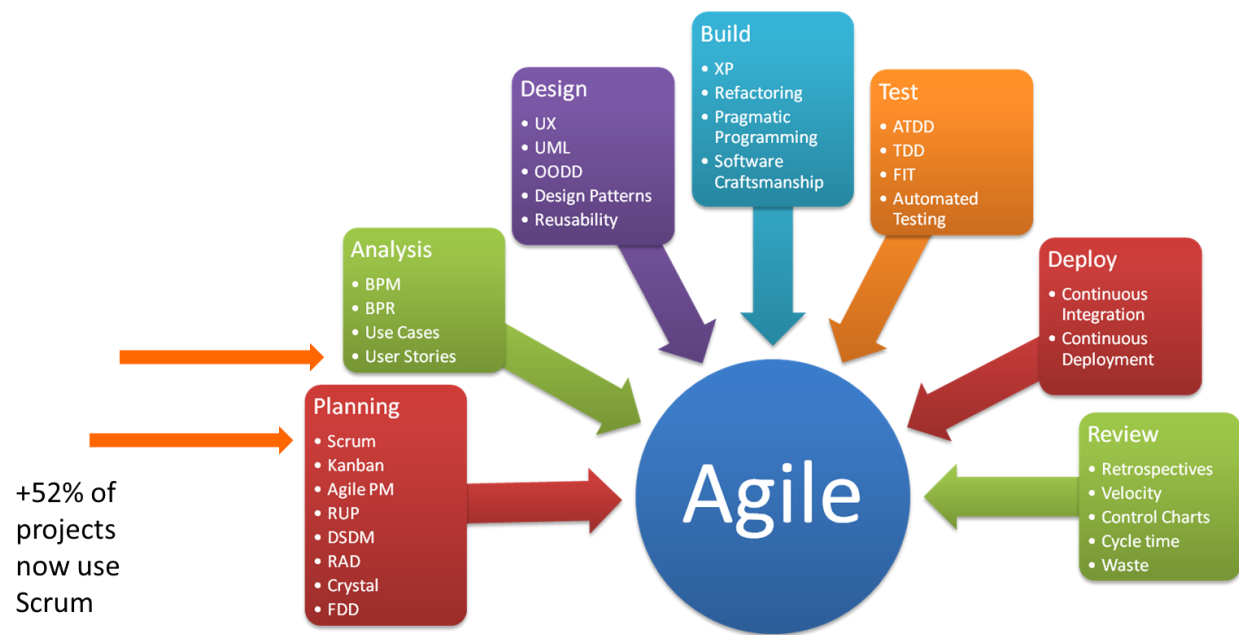
# Agile Methodology



## Benefits:

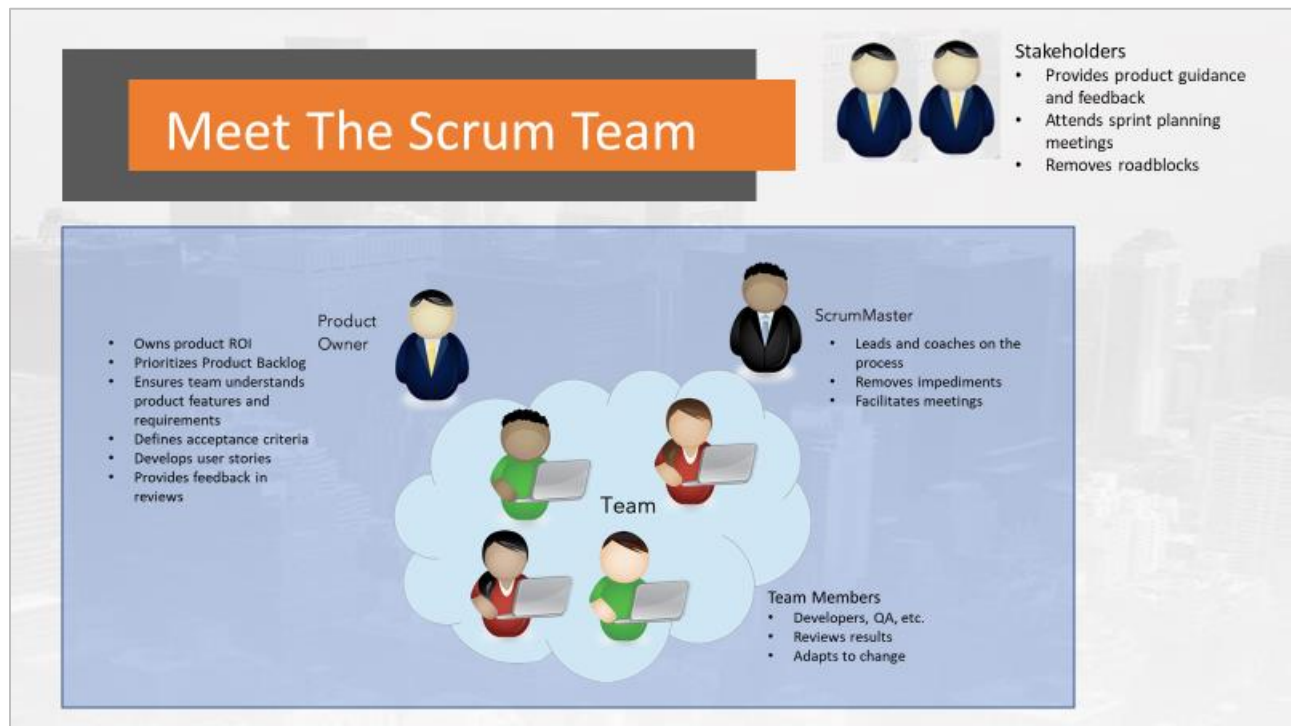
- Stakeholder engagement – higher satisfaction
- Transparency
- Predictable costs and schedule
- Allows for change
- Frequent and early deliver
- High quality
- Business and Customer focus

# Agile Methods





# Meet the Team



Exercise: Select a Team Name & Assign Your Roles:

Hint: leverage your DiSC profiles.

**Team Name:**

**Scrum Master (1) :**

**Product Owner (1):**

**Stakeholder (1):**

**Developers (1+):**

**QA (1+):**

## Timeline Focus



## Product Backlog



### Plan Sprints:

Sprint 1: What we make on Tuesday  
Sprint 2: What we make on Wednesday  
Sprint 3: What we make on Thursday

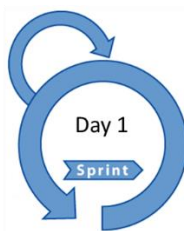
## Sprint Backlog

Sprint 1:

**Epic: Desert**  
Pumpkin Pie  
Chocolate Pie

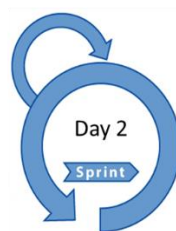
## Sprint Backlog

Sprint 1:  
**Epic: Desert**  
Pumpkin Pie  
Chocolate Pie



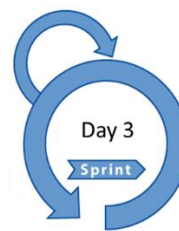
## Sprint Backlog

Sprint 2:  
**Epic: Dinner**  
Mash Potatoes  
Mac N Cheese



## Sprint Backlog

Sprint 3:  
**Epic: Dinner**  
Turkey  
Gravy



### Exercise: Plan Your Time

This afternoon you will be implementing one sprint. Plan out your time for the following:

- **Number of sprints = 1**
- **Must include: Sprint planning, the work you need to complete in the Sprint, Daily Scrum (see Best Practices), and Retrospective**

Time	Activity

# User Stories

**AS A** product owner, **I WANT** to be able to effectively write and discuss user stories with the developers, **SO THAT** they can quickly and clearly understand what they need to create in the system.

Why do it?

---

---

---

---

---

What makes a good story?

---

---

---

---

---

Story Types

---

---

---

---

---

---

---

---

3 Basic Formats

---

---

---

---

---

---

---

---

## When Stories Don't Work

---

---

---

---

---

---

---

## Elements of a Good Story

- Required: Story Text
- Optional: (complete depending on complexity)
  - Description
  - Examples, Images, Tables, and Diagrams
    - A picture is worth a 1000 words! Use diagrams first before you write your stories to clearly understand the flow.
  - References and Links
  - Notes and Comments
  - Exclusions: Detailing what's not covered by the story
  - Acceptance Tests

User Story Training

## Examples: Descriptions

**Story Text:** As an account rep, when I am on the phone with a debtor and they dispute the debt, I want to be able to initiate the dispute so the review clock starts and so that the account is removed from collections.

**Description:** (As described in the Epic document with the full listing of the UCs and diagrams (Disputes))

The account rep should have the ability to flag a debt as disputed. This should be via the account screen.

The following is a high-level view of the actions performed by the account rep:

1. Account Rep performs the Dispute action code
2. Account Rep selects account being disputed via tree view
3. Account Rep selects party who is disputing
4. Account Rep enters Dispute information
5. System saves the dispute information and sets the Disputed flag on every account in the set
6. System raises the Dispute Recorded event which moves account set into the Disputed stage

When the account rep goes to enter the dispute details, they should capture the following details:

- Dispute received - verbal, written
- Dispute reason
- Dispute reported by - 60 character string
- Dispute reported date - default to today's date
- Dispute reported amount

38

# Product Backlog

Exercise: Brainstorm your and create your Product Backlog

- Name your Product
- Create your Epics
- Write your Stories
- Prioritize for your Minimum Viable Product (MVP)

Vision: A local grocery wants an online app for their customers to be able to use their favorite menus throughout the week, create a shopping list from their menus and send that shopping list to the Grocery for them to bag and deliver

Core Features:

- Copy or create any recipe into their favorites list
- Be able to create a set of recipes they will use for any time period (example for the week)
- The app automatically determines what items are in the recipes and creates a grocery list
- The shopper can add or remove any item the grocery has in stock
- The shopper can pay for their groceries and schedule a time and place for delivery

Notes:

---

---

---

---

---

---

---

# Design

## Exercise: Create Mockups and Acceptance Tests

- Create 1 mock ups for your product
- Write 1 Acceptance Test per team member

Notes:

---

---

---

---

---

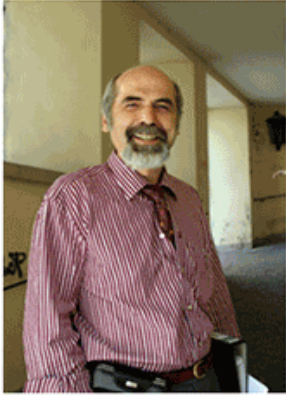
---

---

# Personas

Exercise: Create personas for your product  
Create personas for your product.

Example:



## Bill

- 52 Years Old
- University Professor of English
- Lives in Bradley Maine
- Married 28 years
- 2 Children (One in college)
- His son, Jeremy is 26 years old and has Down syndrome.

Bill and his wife both work full time. They each make 5 figure incomes that allow them to travel during the holidays with his wife and two kids.

Bill uses the web for work and home. He checks his email and administers online classes. He also looks for events and places that the whole family could visit. He is impatient with the internet because his back gets sore if he sits at the computer too long.

Bill feels fine though he has high blood pressure. He eats healthy and tries to exercise at least two or three times a week.

He uses glasses when he reads and surfs the web. He hates sites with small print because they make him feel old.

"I need to find the information fast. I have classes to teach."

"The internet is a tool to help find experiences that enrich my life as well as my kids."

### Typical Web Tasks

- Checks email
- Plans holiday trips
- Looks for local events and ways to help Jeremy.

Notes:

---

---

---

---

---

---

---



## Estimating

Notes:

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

## Implement Your Product

Exercise: Using the work you have already done, perform the following to implement your product:

- Plan your sprint backlog – what do you have left to do?
- Assign the work to your team members
- Finish up your stories – 2 per person per team
- Finish up your mock ups – 2 total per team
- Finish up your Acceptance Tests – one per person
- Finish up your User Personas – 2 per team
- Perform 1 Standups using the standard questions
- Perform 1 retrospectives
- Show your work on a Task Board (ex. Kanban board)
- Show your final product you deliver to the stakeholders

# Best Practices

## Daily Scrum

1. Facilitator: Scrum Master
2. Everyone stands
3. Takes 15m or less
4. Focus on the 3 Questions:
  - What did you do yesterday?
  - What are you working on today?
  - Do you have any roadblocks keeping you from doing your job?
5. Follow up or lengthy discussions are held post-daily scrum
6. Start on time
7. Include remote team members
8. Hold the Daily Scrum everyday at the same time
9. Focus on continuous improvement

# Template

## Retrospective

1. Score the team (1-not effective to 5-most effective) on the following criteria:

#	Question	Rank
1.	How well did the team work collaborate well together?	
2.	Did everyone play their role?	
3.	Did others chip in to get the work done even if it wasn't part of their responsibility?	
4.	Did the team display a high level of trust in each other and themselves?	
5.	Were you productive, did you complete all the work?	
6.	Was everyone motivated to work?	
7.	Were people accountable for getting their assigned work done?	
8.	How well was the daily scrum run?	
<b>Total Score (40 possible)</b>		
<b>What is the average score among all team members?</b>		

2. What improvements would you recommend your team make for the next sprint?

---

---

---

---

---

---

---

3. What questions do you have about what you have learned today?

---

---

---

---

---

---

---

---

---

---

# Glossary:

## Acceptance Testing

An acceptance test is a formal description of the behavior of a software product, generally expressed as an example or a usage scenario. A number of different notations and approaches have been proposed for such examples or scenarios. In many cases the aim is that it should be possible to automate the execution of such tests by a software tool, either ad-hoc to the development team or off the shelf.

## Automated Build

In the context of software development, build refers to the process that converts files and other assets under the developers' responsibility into a software product in its final or consumable form. The build is automated when these steps are repeatable, require no direct human intervention, and can be performed at any time with no information other than what is stored in the source code control repository.

## Backlog

A product backlog is a list of the new features, changes to existing features, bug fixes, infrastructure changes or other activities that a team may deliver in order to achieve a specific outcome.

The product backlog is the single authoritative source for things that a team works on. That means that nothing gets done that isn't on the product backlog. Conversely, the presence of a product backlog item on a product backlog does not guarantee that it will be delivered. It represents an option the team has for delivering a specific outcome rather than a commitment.

It should be cheap and fast to add a product backlog item to the product backlog, and it should be equally as easy to remove a product backlog item that does not result in direct progress to achieving the desired outcome or enable progress toward the outcome.

Product backlog items take a variety of formats, with user stories being the most common. The team using the product backlog determines the format they chose to use and look to the backlog items as reminders of the aspects of a solution they may work on.

The product backlog is often referred to as simply the backlog. This entry clarifies the term as product backlog to avoid confusion with sprint backlogs, which are a related, but different concept.

## Backlog Grooming

Backlog grooming is when the product owner and some, or all, of the rest of the team refine the backlog on a regular basis to ensure the backlog contains the appropriate items, that they are prioritized, and that the items at the top of the backlog are ready for delivery.

## Burndown Chart

Burndown charts and burnup charts track the amount of output (in terms of hours, story points, or backlog items) a team has completed across an iteration or a project.

## Continuous Deployment

Continuous deployment aims to reduce the time elapsed between writing a line of code and making that code available to users in production. To achieve continuous deployment, the team relies on infrastructure that automates and instruments the various steps leading up to deployment, so that after each integration successfully meeting these release criteria, the live application is updated with new code.

## Continuous Integration

Continuous Integration is the practice of merging code changes into a shared repository several times a day in order to release a product version at any moment. This requires an integration procedure which is reproducible and automated.

## Daily Meeting

The daily meeting is one of the most commonly practiced Agile techniques and presents opportunity for a team to get together on a regular basis to coordinate their activities.

## Definition of Done

The definition of done is an agreed upon list of the activities deemed necessary to get a product increment, usually represented by a user story, to a done state by the end of a sprint.

## Epic

An epic is a large user story.

## Estimation

In software development, an "estimate" is the evaluation of the effort necessary to carry out a given development task; this is most often expressed in terms of duration.

## Given-When-Then

The Given-When-Then formula is a template intended to guide the writing of acceptance tests for a User Story: (Given) some context, (When) some action is carried out, (Then) a particular set of observable consequences should obtain.

## Incremental Development

In an Agile context, Incremental Development is when each successive version of a product is usable, and each builds upon the previous version by adding user-visible functionality.

## Iteration

An iteration is a timebox during which development takes place. The duration may vary from project to project and is usually fixed.

## Iterative Development

Agile projects are iterative insofar as they intentionally allow for "repeating" software development activities, and for potentially "revisiting" the same work products (the phrase "planned rework" is sometimes used; refactoring is a good example).

## Kanban Board

A Kanban Board is a visual workflow tool consisting of multiple columns. Each column represents a different stage in the workflow process.

## Minimum Viable Product (MVP)

A Minimum Viable Product is, as Eric Ries said, the "version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort."

## Minimum Viable Product (MVP)

A Minimum Viable Product is, as Eric Ries said, the "version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort."

## Planning Poker

An approach to estimation used by Agile teams. Each team member "plays" a card bearing a numerical value corresponding to a point estimation for a user story.

## Points

Agile teams generally prefer to express estimates in units other than the time-honored "man-hours." Possibly the most widespread unit is "story points."

## Refactoring

Refactoring consists of improving the internal structure of an existing program's source code, while preserving its external behavior.

## Retrospective

The team meets regularly to reflect on the most significant events that occurred since the previous such meeting, and identify opportunities for improvement.

## Scrum

Scrum is a process framework used to manage product development and other knowledge work.

## Scrum Master

The scrum master is responsible for ensuring the team lives agile values and principles and follows the practices that the team agreed they would use.

## Scrum of Scrums

A technique to scale Scrum up to large groups (over a dozen people), consisting of dividing the groups into Agile teams of 5-10.

## Sprint Backlog

A sprint backlog is the subset of product backlog that a team targets to deliver during a sprint in order to accomplish the sprint goal and make progress toward a desired outcome.

## Sprint Planning

Sprint planning is an event that occurs at the beginning of a sprint where the team determines the product backlog items they will work on during that sprint.

## Team

A "team" in the Agile sense is a small group of people, assigned to the same project or effort, nearly all of them on a full-time basis.

## Three Questions

The daily meeting is structured around some variant of the following three questions: What have you completed? What will you do next? What is getting in your way?

## Timebox

A timebox is a previously agreed period of time during which a person or a team works steadily towards completion of some goal.

## Unit Testing

A unit test is a short program fragment written and maintained by the developers on the product team, which exercises some narrow part of the product's source code and checks the results.

## Usability Testing

Usability testing is an empirical, exploratory technique to answer questions such as "how would an end user respond to our software under realistic conditions?"

## User Stories

In consultation with the customer or product owner, the team divides up the work to be done into functional increments called "user stories."

## Velocity

At the end of each iteration, the team adds up effort estimates associated with user stories that were completed during that iteration. This total is called velocity.