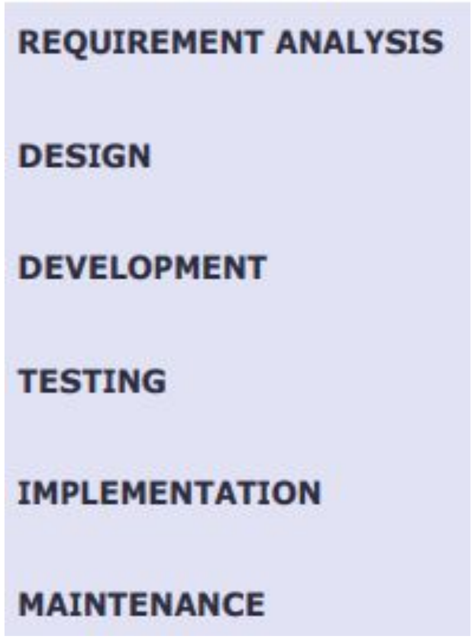
Project Management - Agile

Overview

Traditional App Dev Methodology

The Traditional methodology, often called the ‘Waterfall’ framework, was generally a sequential set of steps. It required the majority of the work for each step to be completed **before** any of the next steps could begin.

1. **Requirement Analysis** - Document all aspects of the business, system, and user requirements as the initial step in the project.
2. **Design** - Based on the completed set of requirements, design the system(s), databases, integrations, and user interfaces.
3. **Development** - Create the code for the application.



REQUIREMENT ANALYSIS

DESIGN

DEVELOPMENT

TESTING

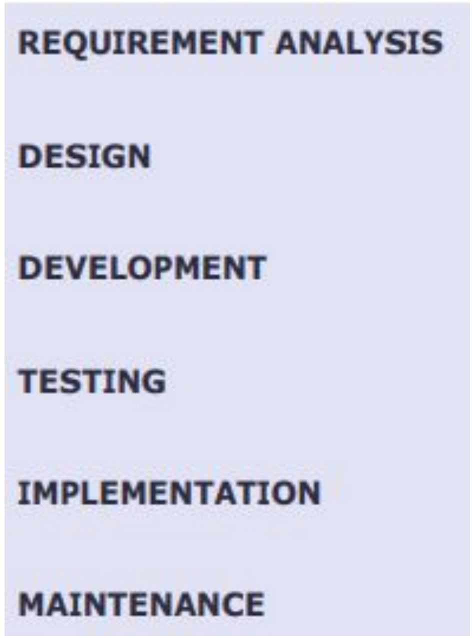
IMPLEMENTATION

MAINTENANCE

Traditional App Dev Methodology (cont.)

4. **Testing** - The starting point of testing has varied, but now general test case creation begins with requirements, or in some cases, as late as with code development. Execution begins after code is delivered.
5. **Implementation** - Code deployment/release to the user base in production. This is where the code “goes live” in production.
6. **Maintenance** - Post deployment support of the application, which may include bug fixes, enhancements, upgrades (operating system, DB, services, servers, etc.).

Additional activities may be included, depending on local practices such as user acceptance and load testing.



REQUIREMENT ANALYSIS

DESIGN

DEVELOPMENT

TESTING

IMPLEMENTATION

MAINTENANCE

Traditional Approach Problems

- ❑ Assumes all requirements are known up front and nothing will change before the deployment of the application.
- ❑ Can take several months or years from start to implementation, even when the project goes as planned.
- ❑ If requirements change, the schedule can be significantly impacted, and it would be costly to the project.
- ❑ If testing uncovers design or requirement errors, the cost and schedule impacts can be crippling to the project.

This approach has proven to be very expensive, and schedules are continually at risk of delay. Also, any value from the project is not realized until after implementation.

Changes Needed

Application development needs to be more accepting of changes, faster to realize return on the investment, and less at risk when problems are found.

In recent years, the focus turned to **iterative** development approaches that design, develop, and deploy smaller units or features of the application. Testing needs to be integrated in all aspects of the project. The business also needs to be engaged throughout the process. Several approaches were trialed, and from this movement, Agile was born.

Video: Agile Principles Overview Video - by Mark Snead

Agile

- Agile describes an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing, cross-functional teams and their customers/end users. Agile advocates adaptive planning, evolutionary development, early delivery, and continual improvement, it encourages rapid and flexible response to change. (Wikipedia)
- Agile development was articulated in recent decades by a consortium, originally known as The Object Mentor Group (they have evolved into the Agile Alliance). What is important is that in the years since agile development practices began to take the lead, this group hammered out its core values and wrote them into a document known as The Agile Manifesto.
- Agile is a set of values and principles.

Agile Manifesto

The Agile Manifesto is only 68 words, and very simply says that we can develop better software by valuing the items on the left side of the list more than the item on the right side of the list.

The Agile Manifesto says that: 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/

Principles Behind the Agile Manifesto

We follow these 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.

Principles Behind the Agile Manifesto (continued)

We follow these principles:

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
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.

Frameworks Supporting Agile

Several frameworks support the Agile principles:

- ❑ **Kanban.**
- ❑ **Scrum.**
- ❑ **Extreme Programming (XP).**
- ❑ **Crystal.**
- ❑ **Dynamic Systems Development Method (DSDM).**
- ❑ **Lean Software Development (LSD).**
- ❑ **Feature-Driven Development (FDD).**

We will look more closely at **“Scrum”** next.

What Did You Learn?

- ❏ We reviewed some of the problems related to traditional app dev methods and the risks it creates for cost and schedule.
- ❏ We discussed why Agile is needed to help mitigate many of those risks, and where the Agile approach originated.
- ❏ We reviewed tenets of the Agile Manifesto.
- ❏ We reviewed the core principles behind the Agile approach.
- ❏ We identified several Agile Frameworks.

Knowledge Check

Which are valid risks/problems with the traditional app dev approach? (Select all that apply.)

- ❏ Assumes all requirements are known up front and nothing will change before deployment.
- ❏ Can take several months or years from start to implementation.
- ❏ Test cases can be developed as requirements are being completed.
- ❏ Requirement changes can be significant and costly.

Connect the items considered of higher value to the appropriate item of lower value:

Individuals and interactions -		*- following a plan*
Working software -	over	*- processes and tools*
Customer collaboration -		*- contract negotiation*
Responding to change -		*-comprehensive documentation*

Knowledge Check

Which phrases describe Agile? (Select all that apply.)

1. Solutions evolve through the collaborative effort of self-organizing and cross-functional teams.
2. It promotes a linear approach to development with clearly defined phases in a specific order.
3. It advocates adaptive planning, evolutionary development, and early delivery.

Which frameworks support the Agile principles? (Select all that apply):

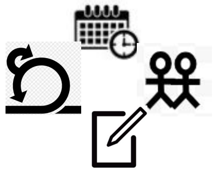
- ❑ **Kanban**
- ❑ **Waterfall**
- ❑ **Scrum**
- ❑ **Lean Software Development**

Scrum Framework

Overview

Scrum - What is it?

- Scrum
  - It is a framework, not a methodology, which is intended to be adaptable to the team's needs.
  - It is iterative, focused on small deliverable features of the application.
  - It is aligned with the Agile principles.
  - It is self-managed by the team.
  - It is one of multiple Agile compliant frameworks.



As we walk through the parts of Scrum, you may find the Scrum Summary Sheet helpful. Find it in the Documents section of your LMS course, in the 'Agile Scrum' folder.

Filename: "**Scrum-Cheat-Sheet**"

Scrum Theory

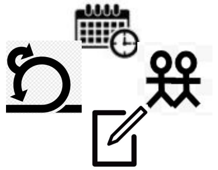
Scrum is founded on an empirical process control theory, or empiricism. Empiricism asserts that knowledge comes from experience and making decisions based on what is known. Scrum employs an iterative, incremental approach to optimize predictability and control risk.

Three Pillars of Scrum:

- ❑ **Transparency** - Significant aspects of the process must be visible to those responsible for the outcome.
- ❑ **Inspection** - Inspection in this context is not an inspection by an inspector or an auditor but an inspection by everyone on the Scrum Team. The inspection can be done for the product, processes, people aspects, practices, and continuous improvements. and Access to artifacts and progress reports.
- ❑ **Adaptation** - Adaptation in this context is about continuous improvement, the ability to adapt based on the results of the inspection. Working smart for your team/products.

The Scrum Guide - *https://www.scrumguides.org/scrum-guide.html*

Scrum Values

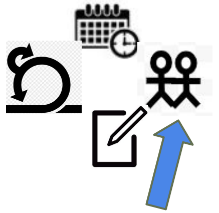


The Scrum team commits to achieving its goals and supporting each other. Their primary focus is on the work of the Sprint to make the best possible progress toward the goals. The Scrum team and its stakeholders are open about the work, the challenges, and respect each other to be capable, independent people. Team members have the courage to do the right thing and to work on tough problems.

- **Commitment**
- **Focus**
- **Openness**
- **Respect**
- **Courage**

Scrum values give direction to the team with regard to their work, actions, and behavior. The decisions that are made, the steps taken, and the way Scrum is used should reinforce these values, not diminish or undermine them.

The Roles of Scrum or Scrum Team



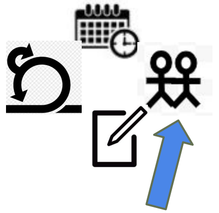
Scrum relies on a self-organizing, cross-functional team. The Scrum team is self-organizing, in that there is no overall team leader who decides which person will do which task or how a problem will be solved. Those are issues that are decided by the team as a whole.

- ❑ **ScrumMaster** - Generally thought of as a coach for the team, helping team members use the Scrum process to perform at the highest level. Facilitates meetings and is charged with removing roadblocks and addressing impediments.

- ❑ **Product Owner** - Owns the product backlog, represents the business, and “grooms” the user stories by prioritizing and writing acceptance criteria.

The Roles of Scrum (continued)

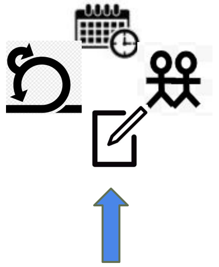
- **Development Team** - Creating a plan for the Sprint, and the Sprint Backlog.
- The coders, testers, and anyone else working the user stories who are charged with doing whatever it takes to get the work done:
  - **Ad hoc members** - Occasionally, special skill sets may be needed, and outside experts may be brought in short-term for a sprint. They engage to complete a unit of work that the team does not have the skill set in place to accomplish (e.g., database administrator or a security expert).
  - **Stakeholders** - Anyone that may have an interest in the development of the application. They may attend the Sprint Reviews.



The Artifacts of Scrum

In Scrum, there are 3 main artifacts that have to be produced:

- **Product Backlog** - List of Epics and User Stories waiting to be worked in future sprints.
- **Sprint Backlog** - The user stories committed to be worked in a sprint.
- **Increment** - The body of work accepted by the product owner at the end of a sprint.



Scrum – Product Backlog Items

In Scrum, items included in the Product Backlog (PBI's) and in the Sprint Backlog (SBI's) are generally categorized in 3 types:

- **Epic** - a big chunk of work that has one common objective. It could be a feature, customer request, or business requirement. In the backlog, it is a placeholder for a required feature with few lines of description.
- **User Story** - captures a description of a software feature from an end-user perspective. It describes the type of user, and what they want and why.
- **Task** - smaller work items used to break a story into smaller units.

Scrum User Stories

User Stories describe the type of user, and what they want and why. The format:

> ***As a*** *<role>,* ***I want to*** *<action>,* ***so that*** *<reason>.*

Example:

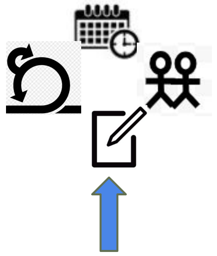
> ***As a*** *customer,* ***I want to*** *check my account balances* ***so that***
> *I can keep track of my account activity.*

A user story may also contain acceptance criteria, priority information, and status information. Generally, they should be kept lean and focused on getting the team the information they need to complete the work. All "discussions" for clarifying the requirements should be included.

Video: User Stories Overview - by Mark Snead

The Artifacts of Scrum – Part II

Sprint Backlog and Increment:



How is all of this work tracked and managed? Requirements are listed as user stories or high-level epics, which are collectively called the "**product backlog.**" Taken together, this backlog captures the business need/want. The product owner is responsible for this list.

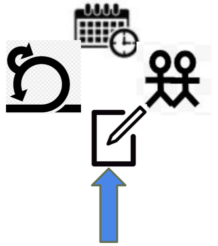
From the Product Backlog, the team selects the work they will commit to complete in the next sprint (Sprint Planning meeting). They analyze the work and break it into smaller user stories as needed, and even into tasks. For example, multiple people may be needed to help complete a user story. This work committed to the sprint becomes the **Sprint Backlog**.

All if the work completed in the Sprint and accepted as DONE by the product owner is referred to as the **Increment.**

Definition of DONE

When is an increment "DONE"?

- ❑ Scrum team members and stakeholders have a shared understanding of what it means for work to be complete (Pillar of Transparency).
- ❑ ScrumMaster guides the development team in knowing how many product backlog items can be selected during a Sprint Planning.
- ❑ Development team of the Scrum team defines a definition of DONE that is appropriate for the product.
- ❑ As Scrum teams mature, it is expected that their definitions of DONE will expand to include more stringent criteria for higher quality.



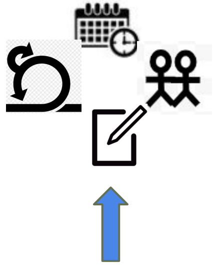
Hands-On Activity

- ❑ Gather your team back together, and return to the requirements that you gathered in the SDLC Elicitation exercise.
- ❑ Divide and assign the requirements among the team by any means you like.
- ❑ Each team member should attempt to write 5-6 user stories based on the assigned requirements. Focus on writing the:
  - ➢ ***As a** <role>, **I want to** <action>, **so that** <reason>.*

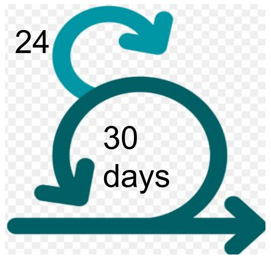
  Do not worry about any other aspect of the user story at this time.
- ❑ Be sure to write it in a text document (Word, Notepad, Google Doc, etc). You will be loading them into a management tool later.
- ❑ You are allowed 20-25 minutes for this exercise.

Sprinting for Success

Remember: Scrum is iterative. Each iteration is called a **Sprint:**

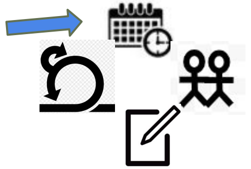


- ❑ Sprints are the heartbeat of Scrum, where ideas are turned into value.
- ❑ A sprint lasts only for a specifically defined period of time. Generally, sprint durations are 2 weeks, 3 weeks, or a month.
- ❑ The work done in a sprint is designed to produce a single ***potentially deployable*** feature of the application.
- ❑ The Scrum team committed to the sprint should be wholly focused on the work and not pulled away to work on other assignments.
- ❑ The team self-manages and commits to the duration, as well as how often they will meet for status meetings. If the team plans for daily status meetings (stand-up meetings every 24 hours) and up to 30 days duration, it can be annotated as:



The Sprint Goal

In the Sprint Planning Meeting, after the Dev Team forecasts the Product Backlog items it will deliver in the Sprint, the Scrum team crafts a Sprint Goal:



- ❑ An objective that will be met within the Sprint through the implementation of the Product Backlog items.
- ❑ Provides guidance to the Dev Team on why it is building the Increment.

Why have a Sprint Goal?

- ❑ It causes the Dev Team to work together rather than work on separate initiatives.
- ❑ A Sprint would be cancelled if the Sprint Goal becomes obsolete.
- ❑ During the Sprint, no changes are made that would endanger the Sprint Goal.

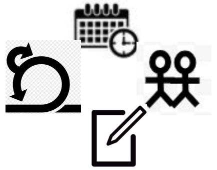
Scrum Events

The Scrum framework has multiple key events or meetings. The main **event** is the **Sprint**. Each other event occurs only once during a Sprint, except for the daily stand-up meeting. Each has a specific purpose and an expected outcome. Some have restricted attendance.

Event	Purpose	Outcome	Participants
Sprint Planning meeting	Select user stories from product backlog to commit for delivery during the current sprint.	Committed scope of work, team members aligned to each user story/task.	ScrumMaster, Product Owner, Dev Team.
Daily Stand-Up meeting / Daily Scrum	Each dev team member will status the team on what was completed yesterday, what will be done today, and call out any impediments.	Impediments identified, ScrumMaster engaged where needed.	ScrumMaster (optional), Dev Team, Product Owner (optional).
Sprint Review	Demonstrate what was produced during the sprint by the dev team and to seek product owner's acceptance.	Product owner decides which user stories are DONE, which to return to backlog,	ScrumMaster, Product Owner, Dev Team, Stakeholders.
Retrospective meeting	Analysis and inspection of the Sprint that has just ended. it's conducted after the Sprint is finished,	Improved process through self-assessment and adjustments.	ScrumMaster, Product Owner, Dev Team.

Measuring Success

Once the Scrum team matures by completing a few sprints, there are a few key measurement tools of which you need to be aware:



- During the Sprint Planning meeting, the team needs a way to estimate the level of work effort required to complete each user story. Some teams estimate the number of "person work days" it will take to complete the work. **Story Points** enable a way to estimate ranges of effort. The table below is an example that a team may use:

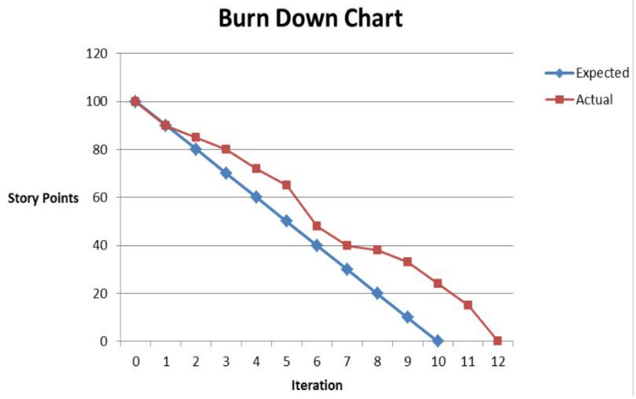
Story Points	**Workdays**
1	1 -2 days
2	3 - 5 days
3	6 - 14 days

- The ScrumMaster will guide the team in setting and using story points as they deem appropriate. This is a tool for the team to estimate their work, so the team should review and "true up" the process in the retrospective meetings.

NOTE: There are a range of methods used to measure story points. The Fibonacci Sequence and T-Shirt sizes are the most widely used. Each scrum team decides which method to use. Story Point estimation here.

Measuring Success (continued)

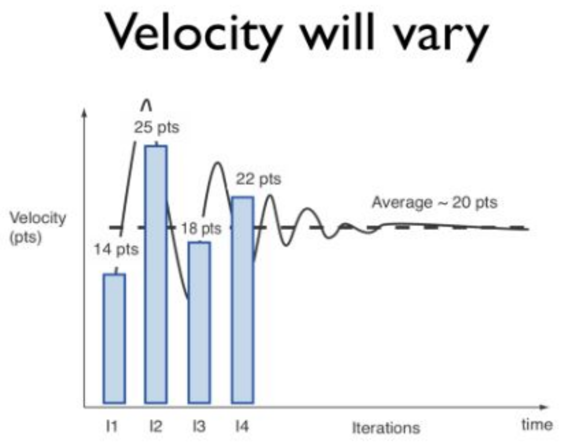
Another key measurement tool is the **Burndown Chart**. This chart shows the amount of work committed to during the sprint, and tracks the team's progress towards completion. Each day before the stand-up meeting, team members should update their percentage complete on each user story/task assigned.



For example, a team commits to 100 story points in the Sprint Planning meeting. The burndown chart will show the *expected* burndown rate and the *actual* burndown rate, based on the overall percentage of work completed. At the end of the sprint, the value should always be 0. This is a very valuable tool, and will allow everyone to see if the work is proceeding on schedule.

Measuring Success (continued)

One last measurement we want to cover is **Velocity**. As a team matures and gets a few sprints under their belt, they can look back at those sprints and review the amount of work completed in each sprint. When measuring those efforts in story points, they can gauge the average number of story points completed in each sprint.



Velocity is the sum of story points completed over one sprint. The average velocity is taken across multiple sprints, as seen in the diagram. Several things must be taken into account when estimating future sprints work loads, such as:

- Vacation days/Holidays of team members.
- Maturity of the team.
- Additions or losses of members.

Plan for success when planning a sprint. Do not overload the team. Velocity must be sustainable, which is why average velocity is important.

Questions

Hands-On Activity

The instructor will provide you access to the JIRA instance. You will also get a demonstration of the tool.

jira.perscholas.org:8080

Create user stories in your assigned JIRA project from the ones your team wrote in the prior exercise. This should be mainly a copy/paste exercise. Add additional information, such as Priority and Story Points as directed.

The goal is to experience working with a Scrum tool. Each team will create a Sprint, assign user stories to it, and then conduct one or more Stand-Up sessions.

NOTE: Exercise details will be provided by your instructor.

Knowledge Check

1. The Sprint Review is the first meeting in the sprint cycle.

   True or False

2. Stakeholders are encouraged to attend the stand-Up meetings.

   True or False

3. The Product Owner owns and manages the product backlog.

   True or False

4. A dev team of a Scrum team must include only four members.

   True or False

TEKsystems
Own change

Knowledge Check

5. What value after the last day of the sprint should the Burndown Chart of a successful sprint show?

   A. 1
   B. 0
   C. Total story points for the sprint
   D. 100 percent

6. A Sprint team's velocity is equal to:

   A. The number of user stories completed per sprint.
   B. The number of days in an average sprint.
   C. The number of story points in an average sprint.
   D. The increment size.