Project Management - Agile

PER SCHOLAS
CUSTOMIZED TRAINING

Overview

- Traditional App Dev Methodology
- Problems with Traditional approach
- A change is needed
- What is Agile
- Agile Manifesto
- Principles behind the Agile Manifesto
- Frameworks supporting Agile
- Your Turn
- Scrum Framework

Traditional App Dev Methodology

Traditional, often called 'Waterfall', framework was generally a sequential set of steps. It required the majority of the work each step to be completed BEFORE any of the next step 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

Traditional App Dev Methodology (cont.)

4. **Testing** - the starting point of testing has varied but now generally 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, like User Acceptance and load testing.

Problems with Traditional approach

- Assumes all requirements are known up front and nothing will change before the deployment of the application
- Can take several months or even 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 placed at risk of delays. Also, any value from the project is not realized until after implementation.

A change is 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
www.youtube.com/watch?v=Z9QbYZh1YXY

What is Agile?

Agile software development describes an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customer(s)/end user(s). It advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and 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's 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 Manifesto

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 (Cont.)

We follow these principles <continued>:

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 application development methods, the risks it creates for costs and schedule
- We discussed why Agile is needed to help mitigate many of those risks and where the approach came from.
- We reviewed tenets of the Agile Manifesto
- We review the core Principles behind the Agile approach
- We identified several Agile Frameworks

Your Turn

Which are valid Risks/Problems with the traditional application development approach? (select all that apply)

- ❑ Assumes all requirements are known up front and nothing will change before deployment
- ❑ Can take several months or even years from start to implementation
- ❑ Test cases can be developed as requirements are being completed
- ❑ Requirements 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*

PER SCHOLAS
CUSTOMIZED TRAINING

Your Turn

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, early delivery

Frameworks supporting the Agile principles include (select all that apply):

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

PER SCHOLAS
CUSTOMIZED TRAINING

Scrum Framework

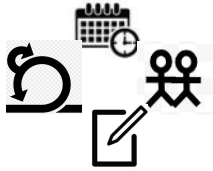
PER SCHOLAS
CUSTOMIZED TRAINING

Overview

- SCRUM - What is it?
- Scrum Theory
- The Roles of SCRUM
- The Artifacts of SCRUM
- Deeper look at User Stories
- The Artifacts of SCRUM – Part II
- Definition of DONE
- Exercise
- Sprinting for Success
- SCRUM Event
- The Sprint Goal
- Measuring Success
- Your turn

SCRUM - What is it?

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

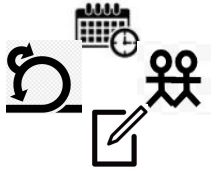


As we walk through the parts of Scrum, you may find the Scrum Summary sheet helpful. Find it in the Documents section of your LMS course, in the 'Agile Scrum' folder.

File name **"Scrum-Cheat-Sheet.pdf"**

Scrum Theory

Scrum is founded on 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.



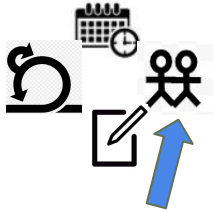
Three Pillars of Scrum:

- **Transparency** - Significant aspects of the process must be visible to those responsible for the outcome.
- **Inspection** - access to artifacts and progress reports
- **Adaptation** - Working smart for your team / products

Note: Our goal is present true Agile/Scrum as it would be in an ideal world. Real world implementations adapt to their organizations.

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

The Roles of SCRUM



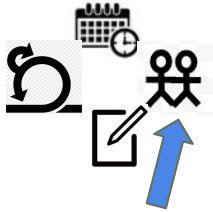
Scrum relies on a self-organizing, cross-functional team. The scrum team is self-organizing in that there is no overall team leader who decides which person will do which task or how a problem will be solved. Those are issues that are decided by the team, as a whole.

- **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 charged with removing roadblocks and addressing impediments for the team.

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

The Roles of SCRUM (Cont.)

- **Development Team** - The coders, testers and anyone else working the user stories. They are charged with doing 'whatever it takes' to get the work done.

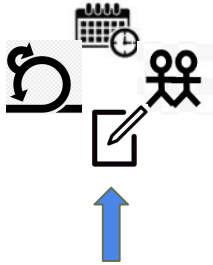


  - **Ad hoc members** - Occasionally special skill sets may be needed, and outside experts may be brought in short term for a sprint. They engage to complete a unit of work that team does not have the skillset to accomplish themselves. Examples may be a Database Administrator or a security expert.

  - **Stakeholders** - Everyone else that may have an interest in the development of the application. They may attend the Sprint Reviews.

The Artifacts of SCRUM

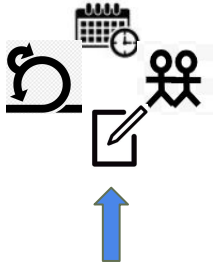
In Scrum there are 3 main artifacts that have to be produced.



- **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

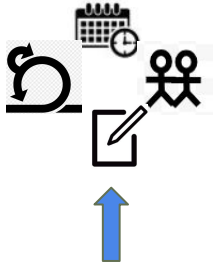
SCRUM – Backup Items

In SCRUM, items included in the Product Backlog (PBI's) and in the Sprint Backlog (SBI's) are general organized 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, what they want and why.
- **Task** - Smaller work items used to break a story into smaller units

Deeper look at User Stories



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

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

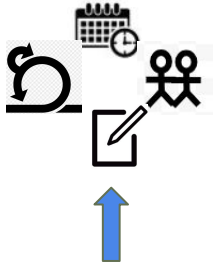
Example:

> ***As a*** *customer,* ***I want to*** *check my account balances,* ***so that*** *I can keep track of 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. Also, all "discussions" to clarify the requirements should be included.

Video: User Stories Overview - by Mark Snead https://www.youtube.com/watch?v=apOvF9NVguA

The Artifacts of SCRUM – Part II



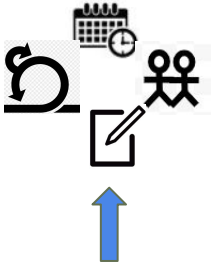
So how is all this work tracked and managed? This may start to sound complex, but its really pretty simple. Requirements are listed as user stories or high-level epics. These collectively are called the "**product backlog**". Taken together this backlog captures the business need/want. The Product Owner owns and 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, break it into smaller user stories as needed and even into tasks, if for instance, multiple folks are needed to help complete a user story. This work committed to the sprint becomes the **Sprint Backlog**.

All the work completed in the Sprint and accepted as DONE by the Product Owner is referred to as the **Increment.**

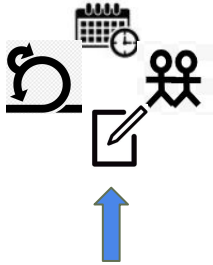
Definition of DONE

When is an increment “Done”?



- Scrum Team members and stakholders must have a shared understanding of what it means for work to be complete (Pillar of transparency)
- Scrum Master guides the Development Team in knowing how many Product Backlog items it can select during a Sprint Planning
- Development Team of the Scrum Team must define a definition of “done” 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

Exercise

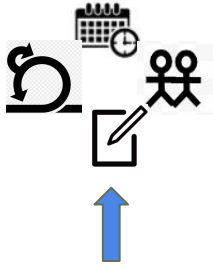


- Return to the Requirements your team gathered in the SDLC Elicitation exercise.
- Gather your team back together and divide up the requirements by any means you like.
- Each of you attempt to write 5-6 User stories based on your assigned requirements. Focus on writing the:
  - ***As a*** *<role>*, ***I want to*** *<action>*, ***so that*** *<reason>*.

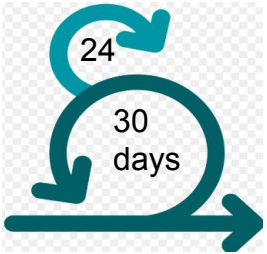
  Don't worry about any other aspect of the User Story at this time.
- Be sure to write it in a Text document (Word, Notepad, Google Doc, etc). You will be loading them into a management tool later.
- Take 20-25 minutes to do this.

Sprinting for Success

Remember that we stated that scrum is iterative? Each iteration is called a **sprint.**



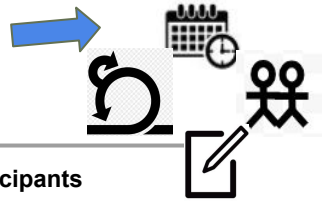
- A Sprint lasts only for a specifically defined period of time. Generally sprint durations are 2 weeks, 3 weeks or a month.
- The work done in a sprint is designed to produce a single ***potentially deployable*** feature of the application.
- The scrum team committed to the sprint should be wholly focused 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 meet to status each other. If the team plans for daily status meetings (Stand-up meetings every 24 hours) and up to 30 days duration then that can be annotated as:



PER SCHOLAS
CUSTOMIZED TRAINING

SCRUM Events

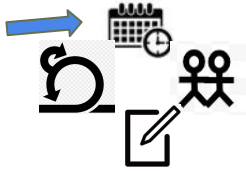
The SCRUM framework has multiple key events or meetings. The main **event** is the **SPRINT**. Each other event occurs only once during a SPRINT except for the daily Stand-up. Each has a specific purpose and expected outcome. Some have restricted attendance.



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.	Scrum master, Product Owner, Dev Team
Daily Stand-Up meeting	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; Scrum master engaged where needed	Scrum master (optional), Dev Team, Product Owner (optional)
Sprint Review	Demonstrate what was produced during the sprint by the dev team and to seek Product Owner acceptance	Product Owner decides which user stories are Done, which return to backlog,	Scrum master, Product Owner, Dev Team, Stakeholders
Retrospective meeting	Analysis and inspection of the Sprint that has just ended	Improved process through self-assessment and adjustments	Scrum master, Product Owner, Dev Team

The Sprint Goal

In the Sprint Planning Meeting, after the Development Team forecasts the Product Backlog items it will deliver in the Sprint, the Scrum Team crafts a Sprint Goal. The Sprint Goal is

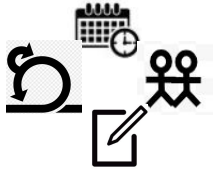


- an objective that will be met within the Sprint through the implementation of the Product Backlog items
- it provides guidance to the Development Team on why it is building the Increment

Why have a goal?

- causes the Development Team to work together rather than 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

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.

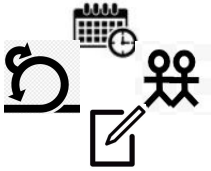
During the Sprint Planning meeting, the team needs a way to estimate the level of work effort required to complete each user story. Some teams just estimate the number of "person workdays" it will take to complete the work. **Story Points** enables a way to estimate ranges of effort. The table below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

The Scrum master 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.

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

Measuring Success (Cont.)

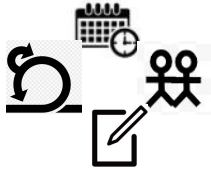
Another key measurement tool is the **Burndown Chart**. This chart shows the amount of work committed to during the sprint and tracks the teams progress towards completion. Each day, before the Standup meeting, team members should update their % complete on each user story/task assigned to them.

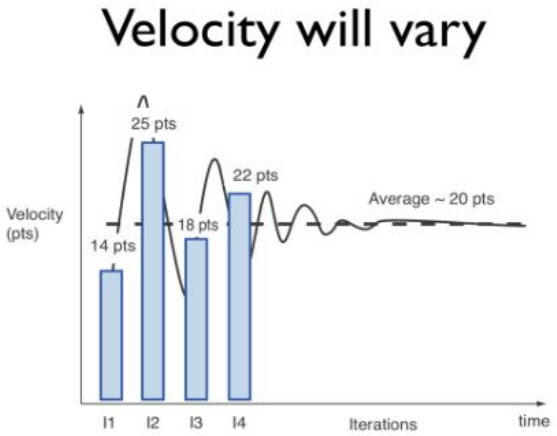


So, for example, a team commits to 100 story points in the Sprint Planning meeting. The burndown chart will show the 'Expected' burn down rate and also the actual, based on the overall % of work completed. At the end of the sprint, the value should always be 0. This is a very valuable tool for everyone to see if the work is proceeding on schedule.

Measuring Success (cont.)

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 gage the average number of story point completed in each sprint.





Velocity is the average of points completed over multiple sprints. Several things must be taken into account when estimating future sprints work loads, such as:

- Vacation/Holiday days of team members
- Maturity of the team
- Addition or loss of members

Plan for success when planning a sprint. Don't overload the team. Velocity must be sustainable.

Questions

PER SCHOLAS
CUSTOMIZED TRAINING

Your turn

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 type exercise. Add additional information like 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, then conduct one or more Stand Up sessions.

Exercise detail: https://docs.google.com/document/d/1A2aOCOZb6n6wbRL9I3IPB0qYT0TlWIOp

Your turn

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 be only 4 members.

   True or False

Your turn

5. The Burndown chart of a successful sprint should show what value after the last day of the sprint?

A. **1**
B. **0**
C. **Total Story Points for the Sprint**
D. **100%**

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**