ECE4574 – Large-Scale SW Development for Engineering Systems
Lecture 2 – Scrum

Creed Jones, PhD

Course Updates

- Quiz 1 is next Wednesday, August 30
  - Will cover the syllabus and lectures 1 – 3; Ten questions, multiple choice & true/false
  - It will be open between 7 PM and 1 AM (Eastern)
  - You can take it any time; once you start you will have 20 minutes to finish
  - No help from others, open notes, you can use my lecture slides and so on

- Project teams
  - Email me if you have a team chosen!
  - Final selection will be completed on August 31

- HW1 is posted
  - Due Friday, September 22 at 11:59 PM, via Canvas

Past Project Topics

- DermaCheckAI (skin cancer)
- Password Manager
- Road Damage Predictor
- Budget Tracker
- Recipz
- Streamlined Streaming
- Mosaic Image Generator
- Cargo Conditions Monitoring System
- RaceLine Optimizer
- King-MME (checkers)
- EasyCAD
- Dynamic Equipment Scheduler
- COVID All in One
- Hospital Management System
- SafeWaze
- COVID-safe appointment scheduling
- Mobile Fitness App
- Spectrogram Audio Visualizer
- Enhance: Health and Fitness Analytics
- Patient Manager
- Get Fit
- Interactive Hospital Management
- Hospital Bed Allocator
- Health & Fitness App

Topics for Today

- Reminder on Software Methodology
- Scrum Methodology
  - Roles
  - Artifacts
  - Backlog
  - Meetings
- Requirements
  - General Requirements
  - User Stories

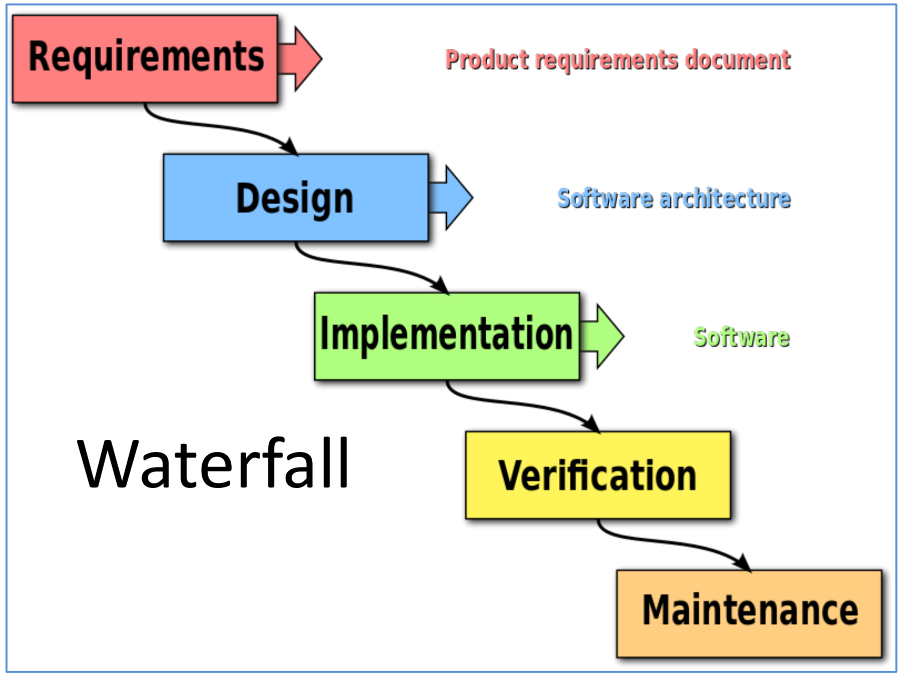
Software Development Methodology – not how to write a program but how to work with others to create software

For a software project, the methodology is the process we use to:

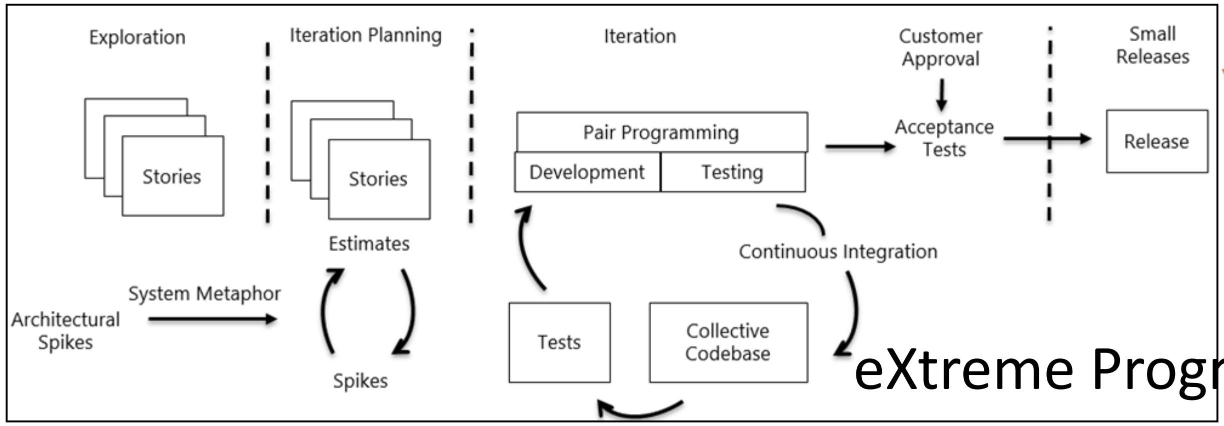
- Understand the requirements for the software
- Split up the work into phases
- Divide the tasks among individuals or teams
- Communicate with each other
- Check on our progress
- Handle changes

VIRGINIA TECH

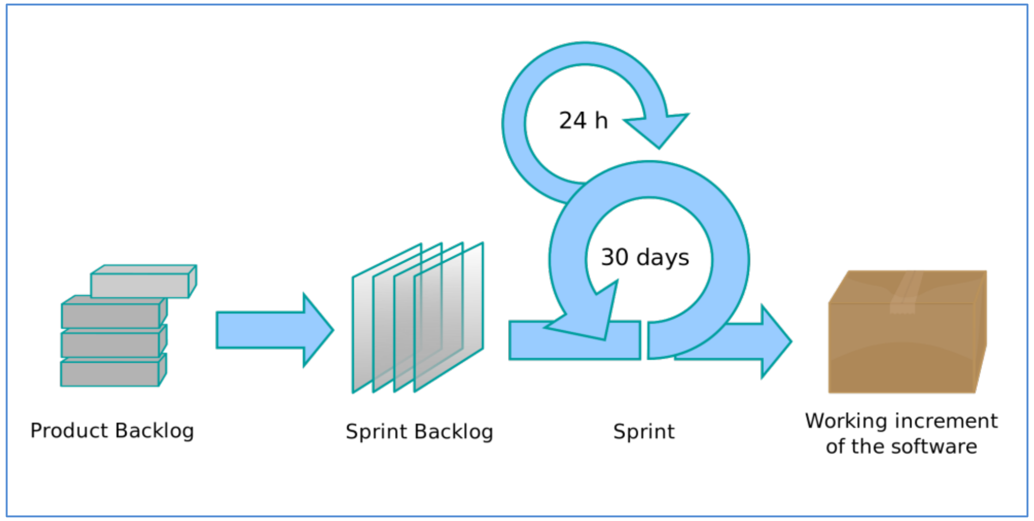
ece BRADLEY DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING



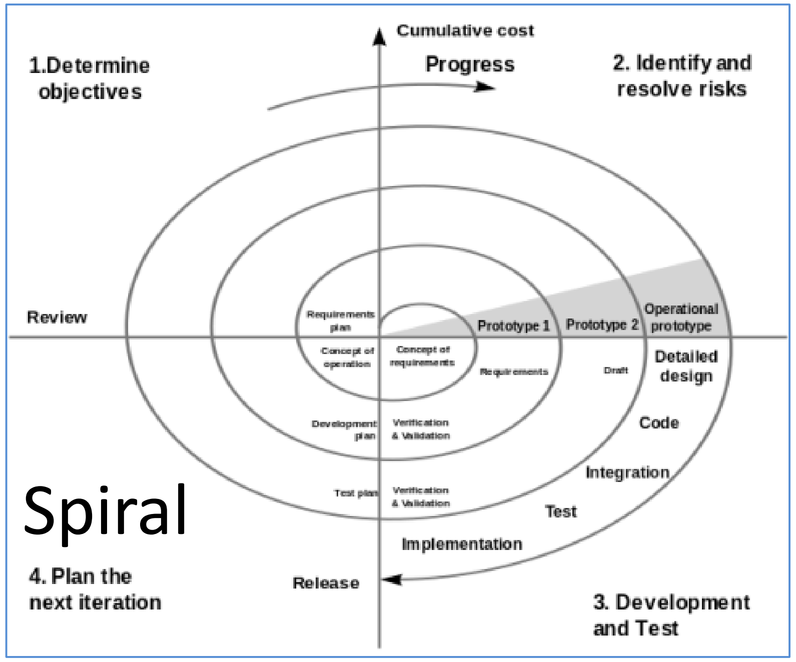
Waterfall



eXtreme Programming

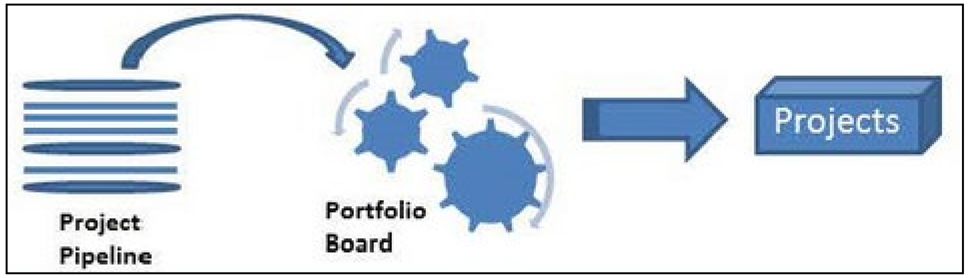


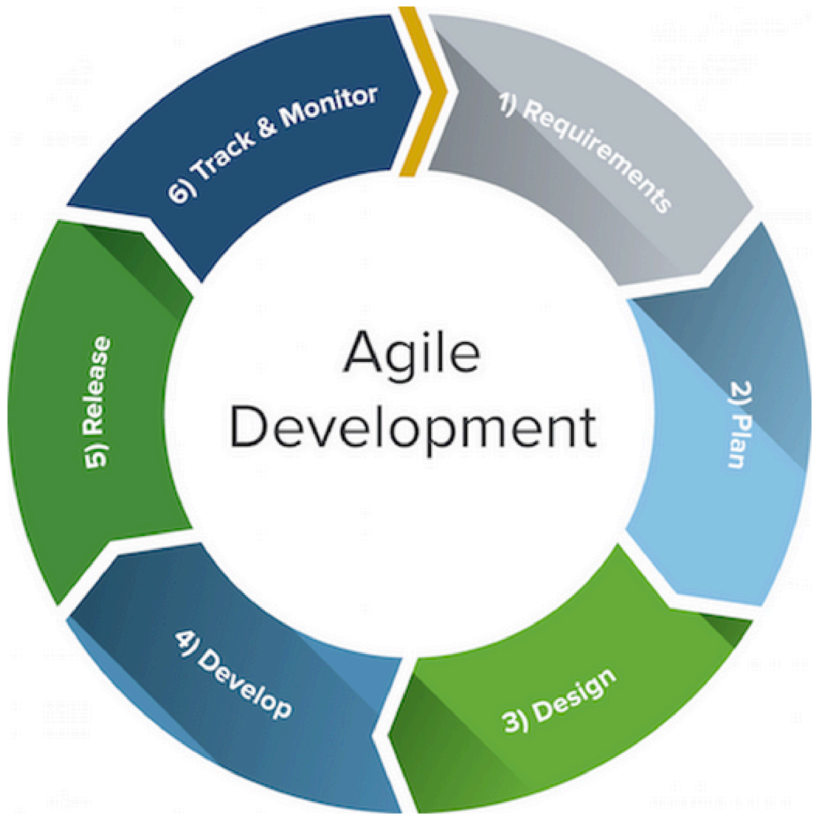
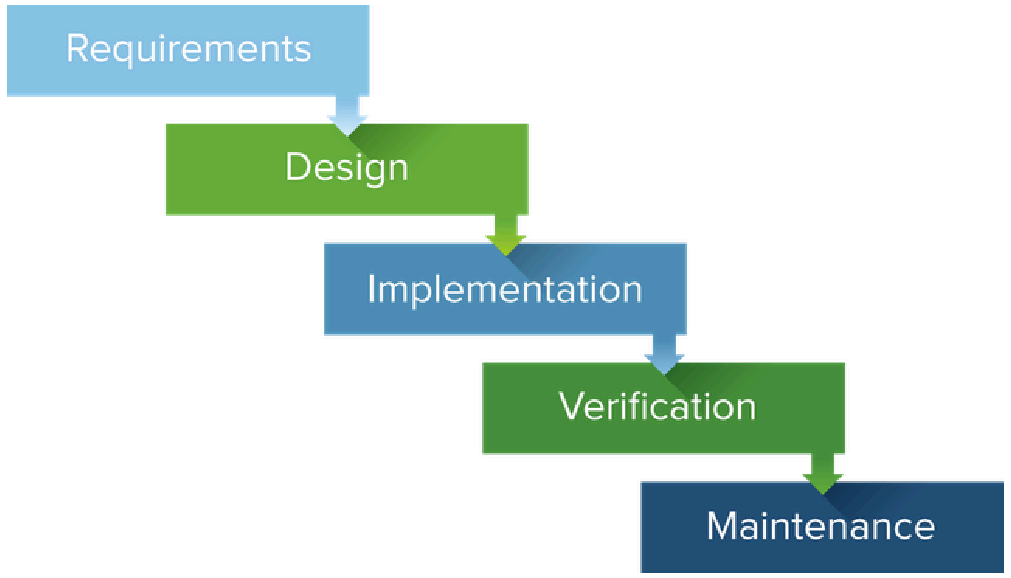
Scrum



Spiral

Kanban



Agile	Waterfall
 Agile Development 1) Requirements 2) Plan 3) Design 4) Develop 5) Release 6) Track & Monitor	 Requirements Design Implementation Verification Maintenance
• Continuous cycles • Small, high-functioning, collaborative teams • Multiple methodologies • Flexible/continuous evolution • Customer involvement	• Sequential/linear stages • Upfront planning and in-depth documentation • Contract negotiation • Best for simple, unchanging projects • Close project manager involvement

Specific Agile Methods

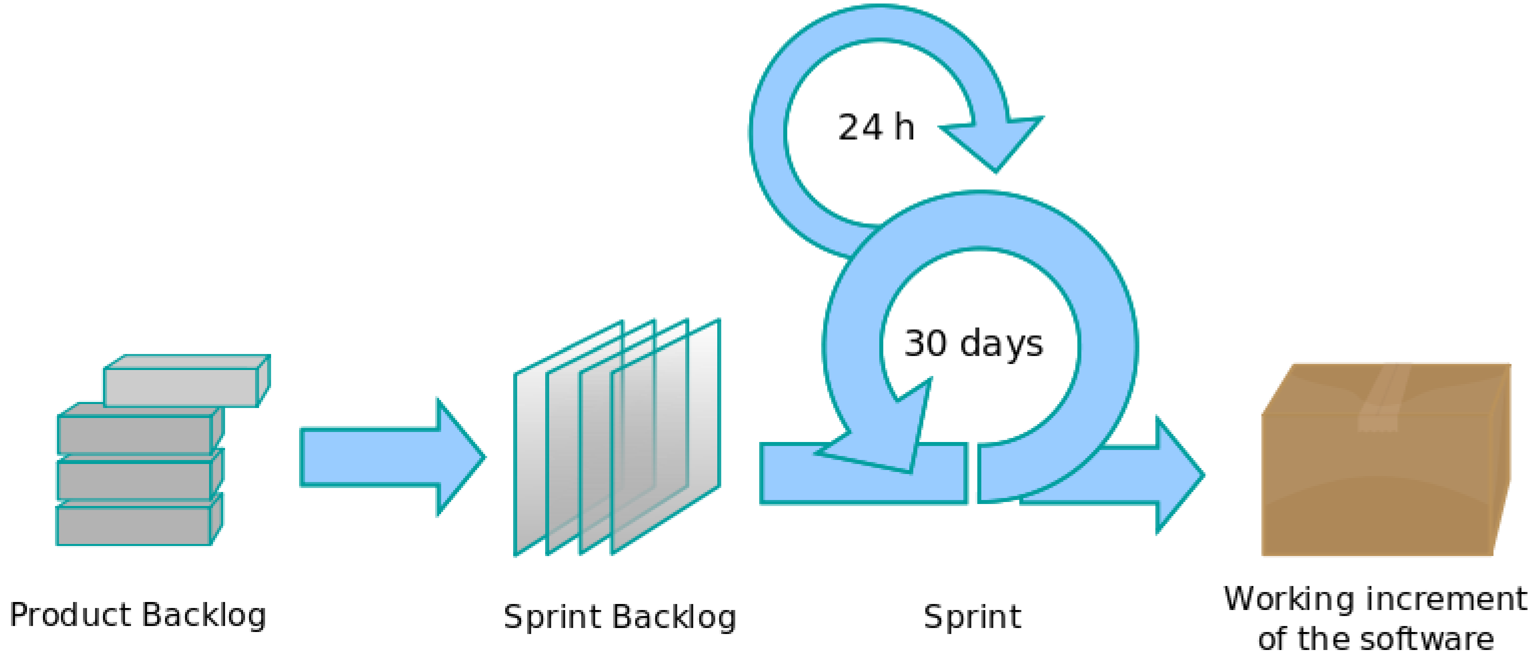
- Two well recognized and complimentary agile methodologies are Scrum and eXtreme Programming (or XP)
- Scrum is focused more on the project management aspects of software development
- eXtreme Programming is more focused on the hands on practice of development

Scrum is a method of working as a team; it's very popular in SW development today

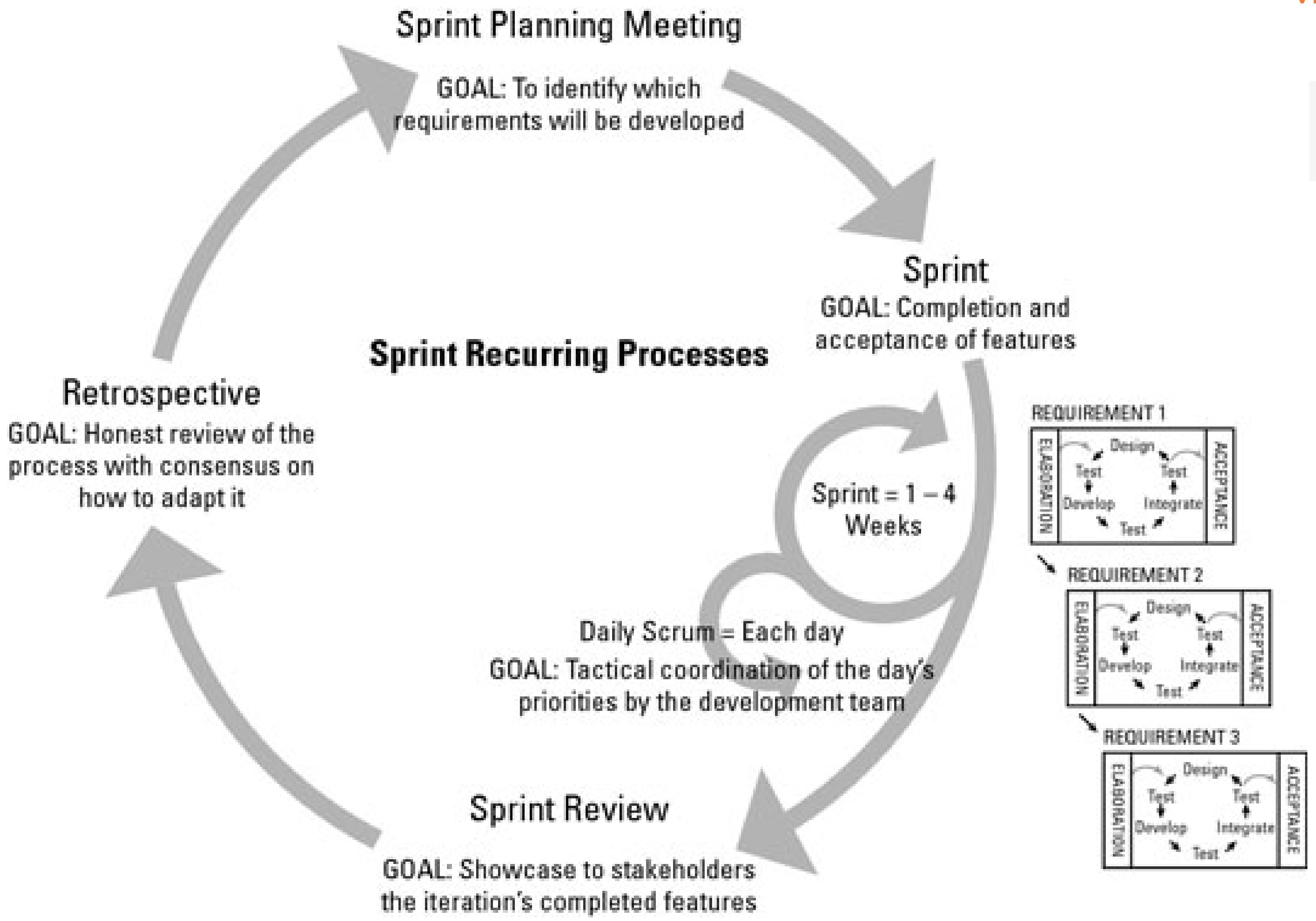
- Scrum is an iterative, communication-rich way for a small to medium sized software team to work together, and to adapt quickly to changes in requirements, team or understanding

- Resources for you:
  - https://www.scrum.org/
  - Goldstein, "Scrum Shortcuts Without Cutting Corners"
    - https://virginiatech.on.worldcat.org/oclc/854858358
  - https://www.atlassian.com/agile/scrum

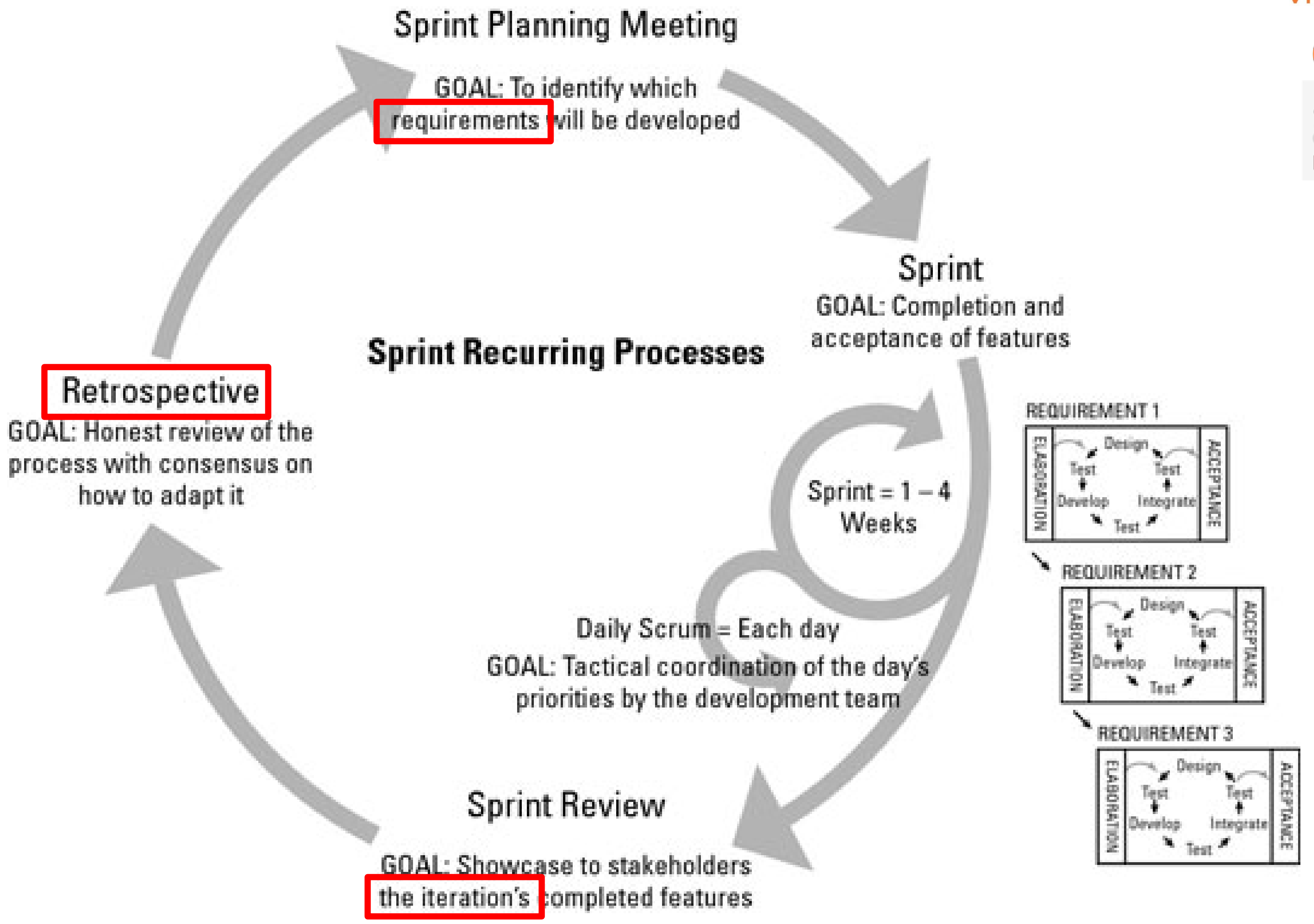
A scrum development project consists of several *sprints* within which certain requirements are considered, met and tested



A Sprint


Sprint Planning Meeting
GOAL: To identify which requirements will be developed
Sprint
GOAL: Completion and acceptance of features
Sprint Recurring Processes
Retrospective
GOAL: Honest review of the process with consensus on how to adapt it
Sprint = 1 – 4 Weeks
Daily Scrum = Each day
GOAL: Tactical coordination of the day's priorities by the development team
Sprint Review
GOAL: Showcase to stakeholders the iteration's completed features
REQUIREMENT 1
REQUIREMENT 2
REQUIREMENT 3
ELABORATION
Design
Test
Develop
Integrate
ACCEPTANCE

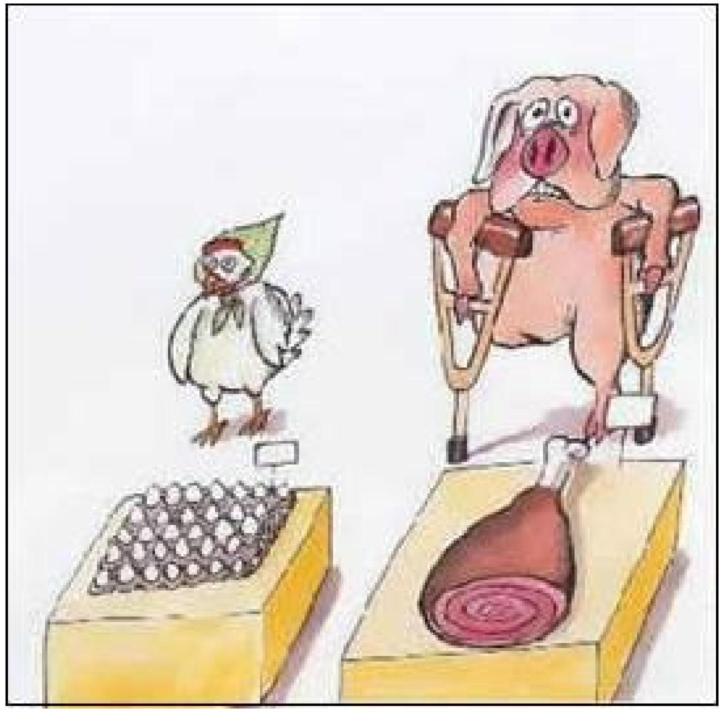
A Sprint and its outputs



Sprint Planning Meeting
GOAL: To identify which requirements will be developed
Sprint
GOAL: Completion and acceptance of features
Sprint Recurring Processes
Retrospective
GOAL: Honest review of the process with consensus on how to adapt it
Sprint = 1 – 4 Weeks
Daily Scrum = Each day
GOAL: Tactical coordination of the day's priorities by the development team
Sprint Review
GOAL: Showcase to stakeholders the iteration's completed features
REQUIREMENT 1
REQUIREMENT 2
REQUIREMENT 3
ELABORATION
Design
Test
Develop
Integrate
ACCEPTANCE

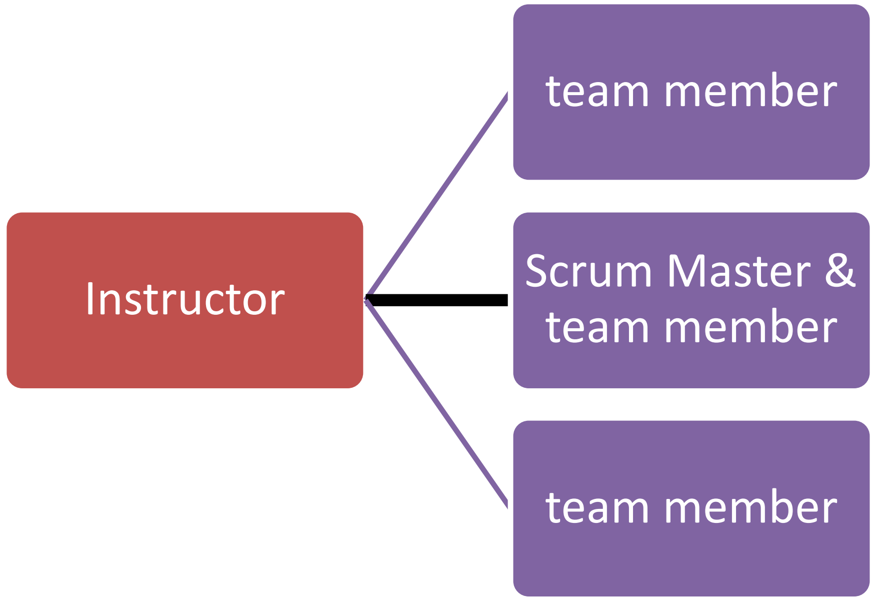
There are several roles in a scrum effort – some are *chicken* roles (partial commitment) and others are *pig* roles (total commitment)

- Product Owner – Pig
  - Represents the stakeholders and is the voice of the customer
- Development Team – Pig
  - Responsible for shipping working software
- Scrum Master – Pig
  - Facilitator responsible for removing impediments to the team and being a buffer between the team and any outside distractions
- Stakeholders – Chicken
  - The people who enable the project and anticipate benefit from the project
- Managers – Chicken
  - People who control the work environment



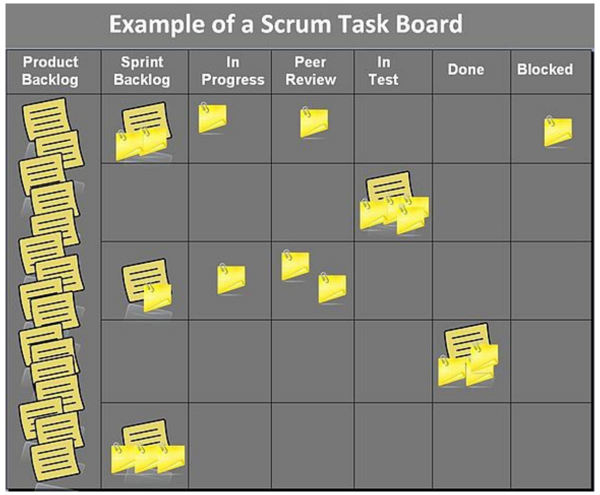
For our course projects, several of these roles will be filled by the team members or the instructor

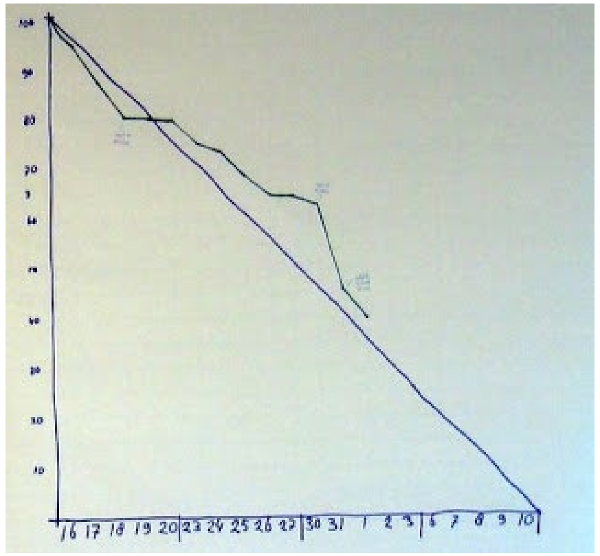
- Product Owner – Pig – **Instructor (sort of)**
  - Represents the stakeholders and is the voice of the customer
- Development Team – Pig – **Team members**
  - Responsible for shipping working software
- Scrum Master – Pig – **Selected team member**
  - Facilitator responsible for removing impediments to the team and being a buffer between the team and any outside distractions
- Stakeholders – Chicken – **Instructor**
  - The people who enable the project and anticipate benefit from the project
- Managers – Chicken – **N/A**
  - People who control the work environment



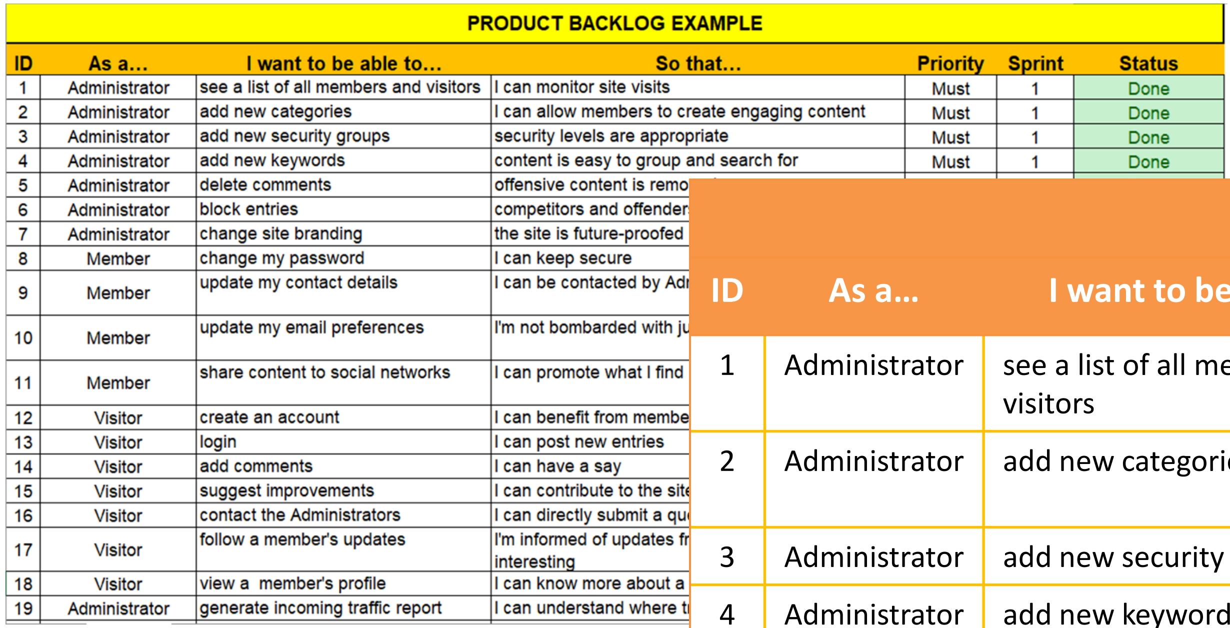
Scrum Artifacts are the records of the work – whether hard-copy or electronic

- **Product Backlog** – master list of ordered product items including stories, features, bugs to be fixed, non-functional requirements any work item that the development team may be responsible for implementing
- **Sprint Backlog** – The product backlog items for the current sprint which the product owner and development team have agreed on
- **Increment** – The potentially shippable software which includes the most recently completed sprint backlog items and all previously completed sprint items
- **Burn down chart** – A chart of incomplete sprint items versus sprint time (optional in small projects)





The Sprint Backlog is extracted from the product backlog, considering risk, team availability and the need to have something working at sprint end



PRODUCT BACKLOG EXAMPLE

ID	As a...	I want to be able to...	So that...	Priority	Sprint	Status
1	Administrator	see a list of all members and visitors	I can monitor site visits	Must	1	Done
2	Administrator	add new categories	I can allow members to create engaging content	Must	1	Done
3	Administrator	add new security groups	security levels are appropriate	Must	1	Done
4	Administrator	add new keywords	content is easy to group and search for	Must	1	Done
5	Administrator	delete comments	offensive content is remo			
6	Administrator	block entries	competitors and offender			
7	Administrator	change site branding	the site is future-proofed			
8	Member	change my password	I can keep secure			
9	Member	update my contact details	I can be contacted by Adr			
10	Member	update my email preferences	I'm not bombarded with ju			
11	Member	share content to social networks	I can promote what I find			
12	Visitor	create an account	I can benefit from membe			
13	Visitor	login	I can post new entries			
14	Visitor	add comments	I can have a say			
15	Visitor	suggest improvements	I can contribute to the site			
16	Visitor	contact the Administrators	I can directly submit a qu			
17	Visitor	follow a member's updates	I'm informed of updates fr interesting			
18	Visitor	view a member's profile	I can know more about a			
19	Administrator	generate incoming traffic report	I can understand where t			

Sprint 1 Backlog

ID	As a...	I want to be able to...	So that...	Priority	Time	Who?	Status
1	Administrator	see a list of all members and visitors	I can monitor site visits	Must	5	Bob	
2	Administrator	add new categories	I can allow members to create engaging content	Must	3	Julie	
3	Administrator	add new security groups	security levels are appropriate	Must	5	Bob	
4	Administrator	add new keywords	Content is easy to group and search for	Must	2	Ronith	
5	Administrator	delete comments	offensive content is removed	Must	3	Bob	
6	Administrator	block entries	bad actors cannot submit content	Must	2	Pat	
7	Member	change site branding	the site is future-proofed in case the brand changes	Could	1	Pat	
8		change my password	Personal security	Must	3	Ronith	

A typical Backlog Item has four major parts

- The Requirement or Feature
  - Can be the title from a User Story
- Estimated Effort
  - In ideal hours or days of effort
  - If estimate is greater than 16 hours decompose it into subtasks
- Associated Priority
- Status (Open/Done)

Before each sprint, we do *Sprint Planning*

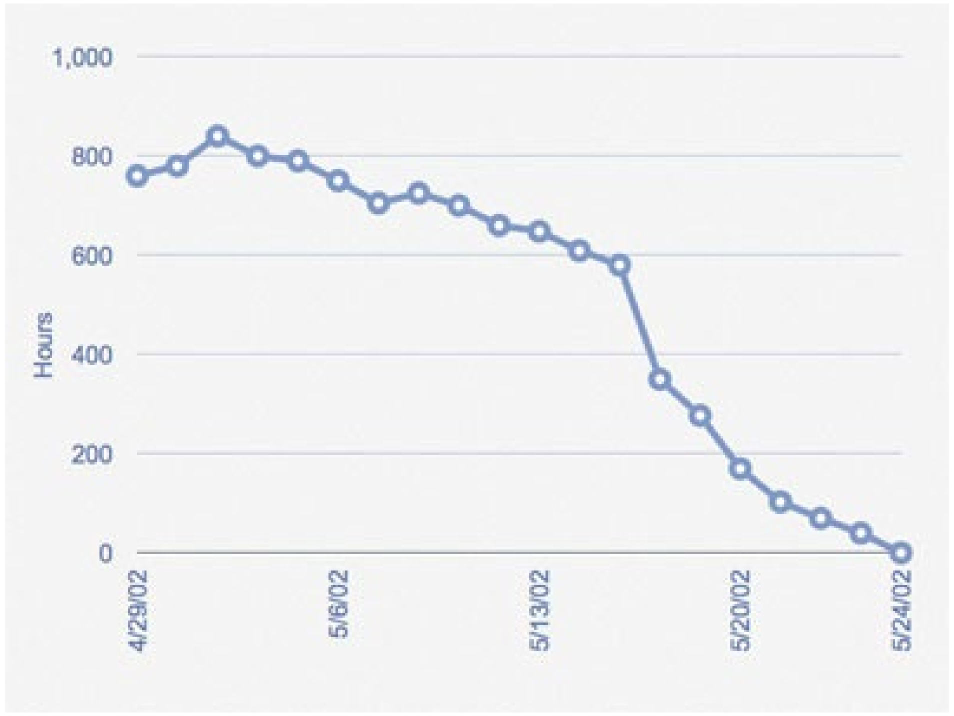
- Composed of two parts
- Part 1 – Select Product Backlog Items
  - Product Owner Selects Items, working with team
    - sometimes delegates this to the team, using stated priorities
  - Dev Team Decides Scope for Sprint
    - Planning Poker – Wide Dephi Technique
- Part 2 – Sprint Backlog
  - Only the Dev Team is involved
  - Output is the Sprint Backlog with tasks, responsibilities and time estimates

The Sprint Backlog has more detail than the product backlog; associated requirement, smaller tasks and information on resource (who), time and possibly status

Requirement	Task	Who	Days	Status
Members can read profiles of other members to explore possible connections	Verify membership before read	B Smith	1	In dev test
	Open profiles temporarily	A Lincoln	2.5	Coding
	Code to record possible connections	B Smith	2	Not started
	Revise UI to accommodate	T Ford	0.5	Not started
Members’ billing information shall be checked prior to any transaction	Connect to financial clearing house	A Lincoln	2.5	Not started
	Block all transactions if billing fails	A Lincoln	1	Not started
	Messaging to user in case of failure	T Ford	1	Not statted

Burndown Chart is a plot of items remaining (typically used in larger projects – we won’t need these in this course project)

- Can be used for Product and/or Sprint



Daily Scrum (or scrum meeting) is held <u>standing up</u> to keep it short and to the point

- Three questions and an update
  1. What have you done since last daily scrum
  2. What are you going to do before the next one
  3. What impediments are you facing
  - Update the Sprint Backlog on estimated hours for tasks

At the end of the sprint, we have a Sprint Review session

- Team presents what it accomplished during the sprint
- Typically takes the form of a demo of new features or underlying architecture
- Informal
  - 2-hour prep time rule
  - No slides
- Whole team participates
- Invite the world

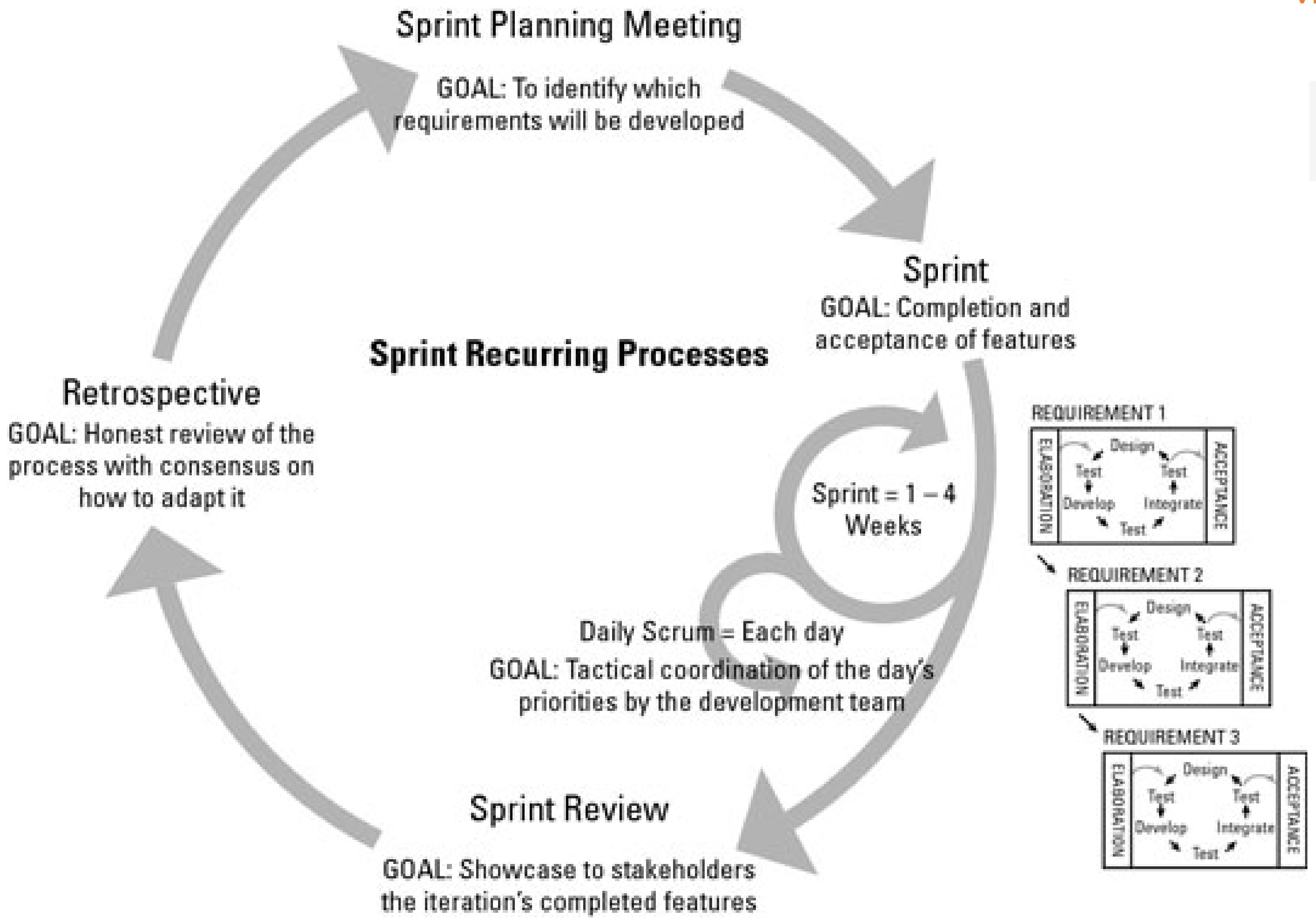
- Credit to Mountain Goat Software, LLC

Sprint Retrospective

- Separate from the Sprint review
  - focused on the process, not the project
- Take a look at what is and is not working
- Typically 15–30 minutes
- Done after every sprint
- Whole team participates
  - ScrumMaster
  - Product owner
  - Team
  - Possibly customers and others

- Credit to Mountain Goat Software, LLC

A Sprint


Sprint Planning Meeting
GOAL: To identify which requirements will be developed
Sprint
GOAL: Completion and acceptance of features
Sprint Recurring Processes
Retrospective
GOAL: Honest review of the process with consensus on how to adapt it
Sprint = 1 – 4 Weeks
Daily Scrum = Each day
GOAL: Tactical coordination of the day's priorities by the development team
Sprint Review
GOAL: Showcase to stakeholders the iteration's completed features
REQUIREMENT 1
REQUIREMENT 2
REQUIREMENT 3
ELABORATION
Design
Test
Develop
Integrate
ACCEPTANCE

REQUIREMENTS

Software Requirements are the stakeholder needs that the software system will satisfy

- *Stakeholders* are people with some interest in the success of the SW project
  - Users
  - Business leaders
  - Clients

- Often the precise requirements are difficult to find out and express clearly
  - Some stakeholders don't know exactly what they want
  - Some stakeholders don't know how it's being done today
  - Often they aren't aware of what can (and cannot) be done
  - We should express them <u>unambiguously</u> to guide development

Software requirements can be simple and somewhat unstructured, but they need to be clear enough that we can all agree on when they are satisfied

There are many ways to write software requirements – let’s talk about three of them:

1. Informal sentences
2. Using formal syntax
3. User stories

Requirements can be written in plain text – this is easy to read but the author must be careful to avoid ambiguity

- Upon startup, create a sample of 10,000 data points from a normal (Gaussian) distribution
- Automatically calculate the min and max values of the data
- Divide the range of the data into 20 equally spaced bins
- Plot the frequency distribution (the histogram) in these bins as a bar chart
- Provide three radio buttons to select other distributions
- When the radio button is pressed, repeat the histogram process with the newly selected probability distribution

Formal requirements are written using a grammar; here, the *Gherkin* language is used (see www.cucumber.io)

```
Feature: Allow new businesses to appear on the map

Scenario Outline: Businesses should provide required data

Given a business:
        | name | location |
        | <business> | <location> |
        When <business> signs up to the map platform
        Then it <should?> be added to the platform
        And its name <should?> appear on the map at <location>

Examples: Business name and location should be required
        | business | location | should? |
        | UNNAMED BUSINESS | NOWHERE | shouldn't |

Examples: Allow only businesses with correct names
        | business | location | should? |
        | Back to Black | 8114 2nd Street, Stockton | should |
        | UNNAMED BUSINESS | 8114 2nd Street, Stockton | shouldn't
|

Examples: Allow businesses with two or more establishments
        | business | location | should? |
        | Deep Lemon | 6750 Street South, Reno | should |
        | Deep Lemon | 289 Laurel Drive, Reno | should |
```

https://www.quora.com/What-are-some-good-examples-of-a-software-requirements-specification

User Stories are a requirements form that is often easier to capture from the user

- Another way to capture requirements
- A good resource is:
  User Stories Applied by Mike Cohn
- The idea is that stories are lightweight conversation reminders – not a contract to perform

The *Overview* is a written description of the story used for planning and as a reminder

- Conversation about the story that serve to flesh out the details of the story
- Tests that convey and document details and that can be used to determine when a story is complete
- A story represents the customer requirements, it is not comprehensive

User Story Essentials

Title of the user story

As a … <user role>

I need to … <functional behavior>

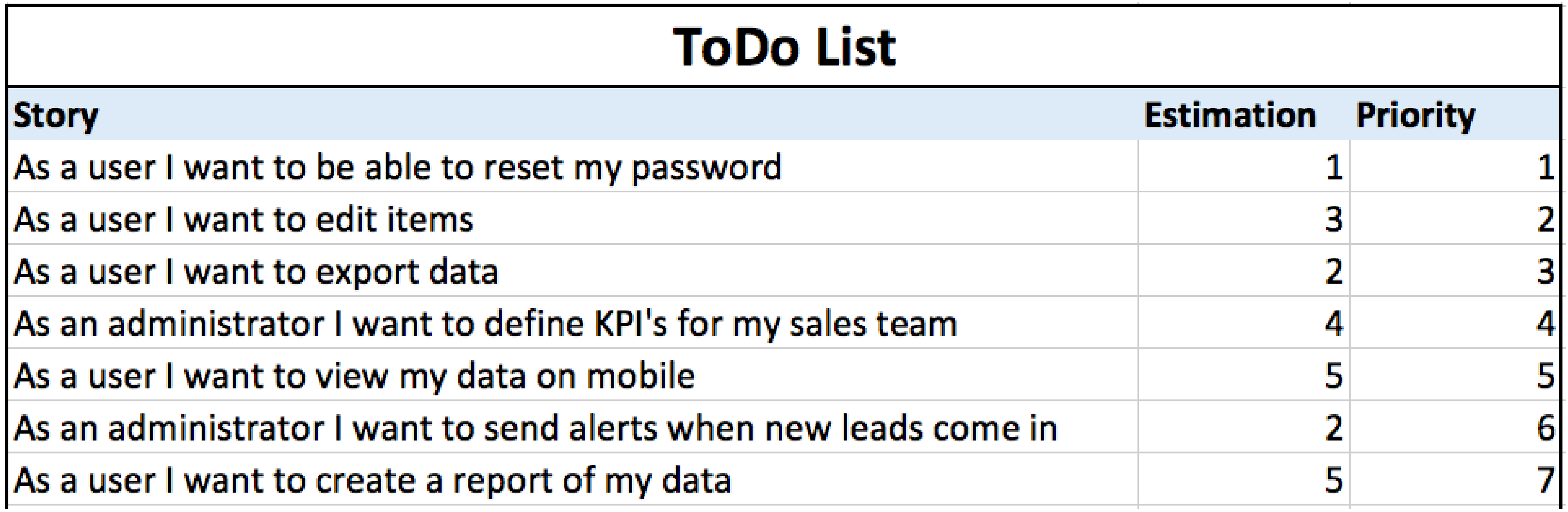
so that … <value statement>

And it will be done when … <acceptance criteria>

Variations on this form are often used

The *Title* of the user story is a short sentence or phrase capturing the essence of the desired functionality

- Used as a short-hand reference to the story
- Place holder on the Product Backlog



ToDo List

Story	Estimation	Priority
As a user I want to be able to reset my password	1	1
As a user I want to edit items	3	2
As a user I want to export data	2	3
As an administrator I want to define KPI's for my sales team	4	4
As a user I want to view my data on mobile	5	5
As an administrator I want to send alerts when new leads come in	2	6
As a user I want to create a report of my data	5	7

When developing a system it is common to use roles to model user behavior

- Roles are also useful for enforcing security or access – this is called Role Based Access Control or RBAC
- Example: Bank System Roles
  - Teller
  - Account Representative
  - Vice President
  - Security Guard

Parts of the user story

- Functional behavior is a description of the action that the user wants to perform
  - Calculation - Information preservation
  - Information transformation - Device manipulation

- Value statement is a description of why this is useful or valuable to the user
  - It should answer the question of why they want to perform the functional behavior

- Acceptance criteria are objective and verifiable statements of either what can or can not be done when this user story is implemented
  - Often adds depth to the story and provides boundaries of behavior

What does a user story look like? How long should it be?

- User stories are often recorded on a 3"X5" cards
- The story on the front
- The acceptance criteria on the back (aka tests)
- Some simply stated stories can involve huge effort – these are called *epics*
- Epics should be decomposed into "easier" stories
  - Large enough to be valuable/meaningful to the customer
  - Small enough to fit within an iteration/sprint
- How many tests? Enough for the customer to clarify the intent – as long as they add value

Try an example - Background

- A company wants to get into the job posting business on the web
- One feature that came up in discussion is being able to list matching job results from a search
- The following slide is a user story attempt to capture the essence of the job listing feature

A User Story example

- Job search results listing
- As a Job Seeker I can view information about each job that is matched by my search criteria. Show the company, title, job description, salary, and location of matching jobs. Allow me to sort by any of the columns except description. This will help me narrow my job search to relevant openings so I can get a jo faster!

FRONT

- Try it with no matching jobs – a nice message should display
- Try it with no search criteria – all jobs should be displayed
- Try it with an empty job description – columns should retain positions
- Try it with a really long job description – text wrapping within column
- Try it with a missing salary – columns should retain positions
- Try it with a six-digit salary – commas in appropriate places

BACK

Are User Stories always used to capture requirements?

- No, they are common but not mandatory

- Some people (or teams) prefer them
- Some applications are best served by user stories
  - heavy user interaction (UX-rich applications)
  - if manual processes are already well documented

- You are free to use whatever requirements form makes sense for your project

Topics for Today

- Reminder on Software Methodology
- Scrum Methodology
  - Roles
  - Artifacts
  - Backlog
  - Meetings
- Requirements
  - General Requirements
  - User Stories