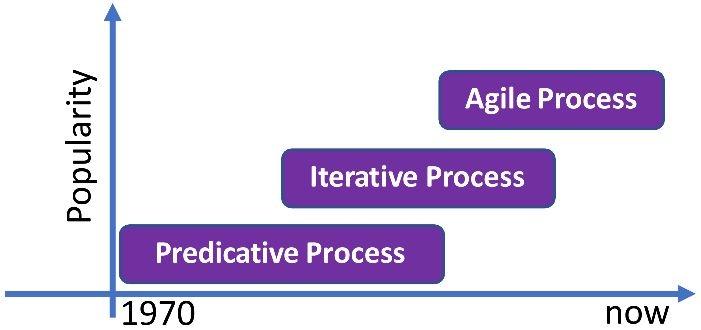
CS335,

Agile Software Development

Dr. Lanlan Gao

Software Process Evolution



Popularity

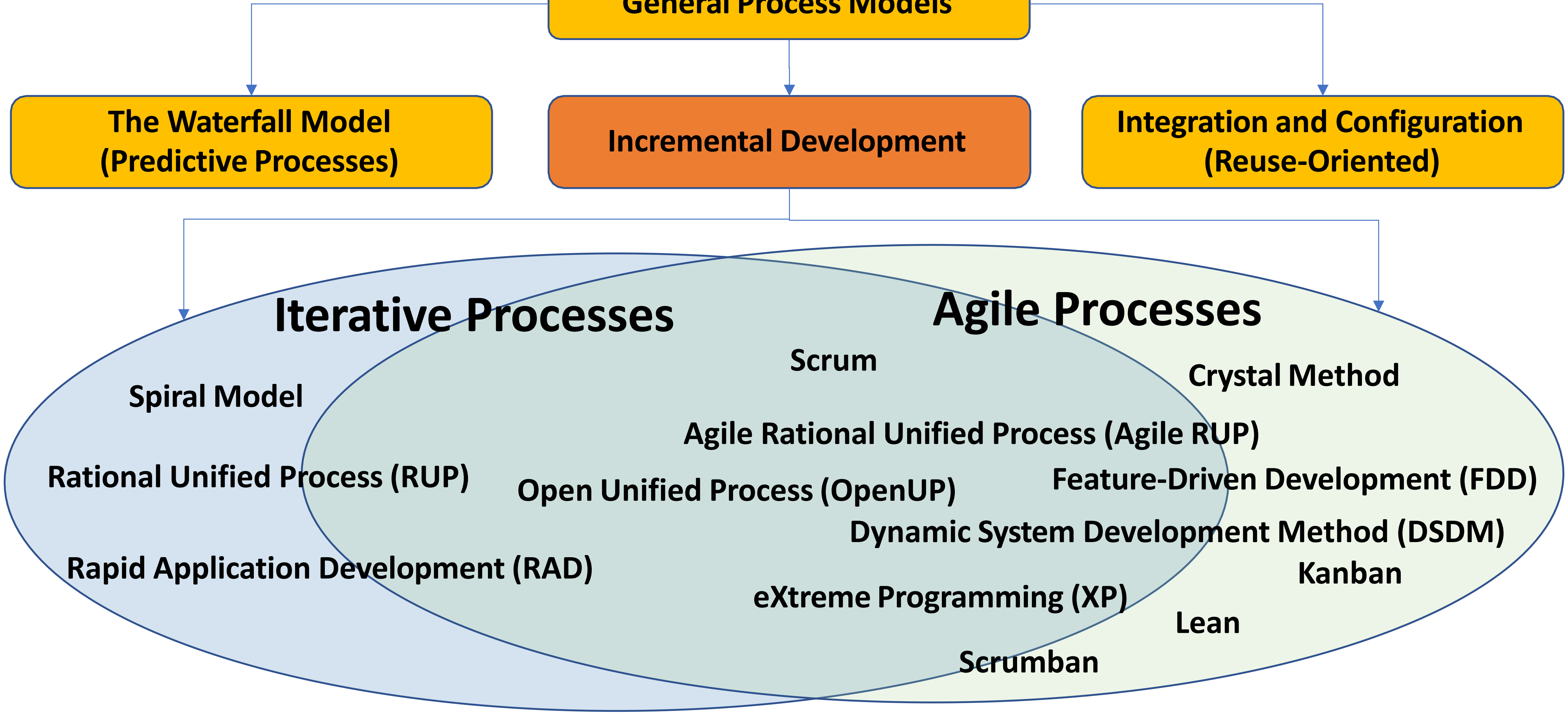
Agile Process

Iterative Process

Predicative Process

1970

now



General Process Models

The Waterfall Model (Predictive Processes)

Incremental Development

Integration and Configuration (Reuse-Oriented)

Iterative Processes

Agile Processes

Spiral Model

Rational Unified Process (RUP)

Rapid Application Development (RAD)

Scrum

Agile Rational Unified Process (Agile RUP)

Open Unified Process (OpenUP)

eXtreme Programming (XP)

Scrumban

Crystal Method

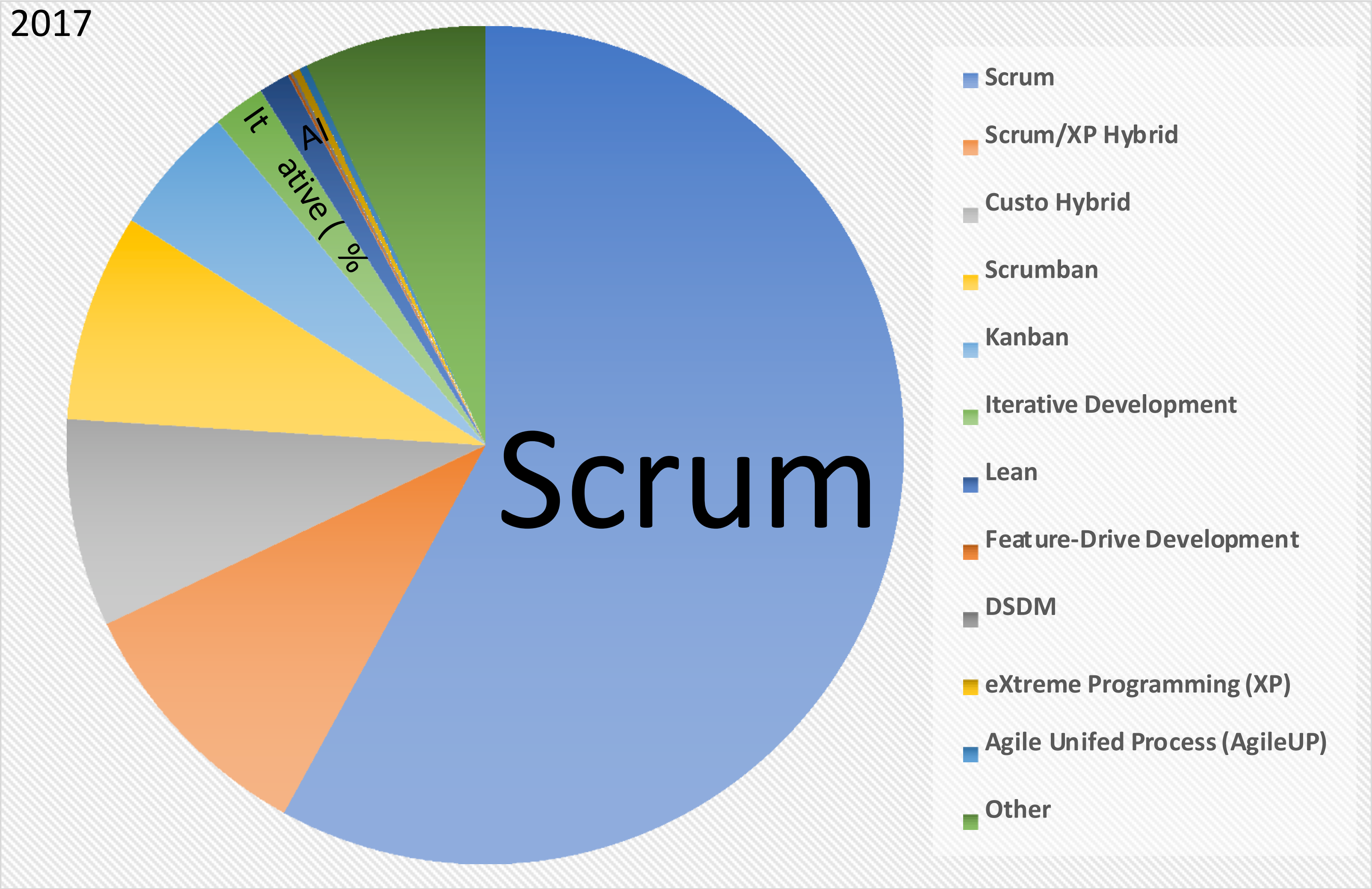
Feature-Driven Development (FDD)

Dynamic System Development Method (DSDM)

Kanban

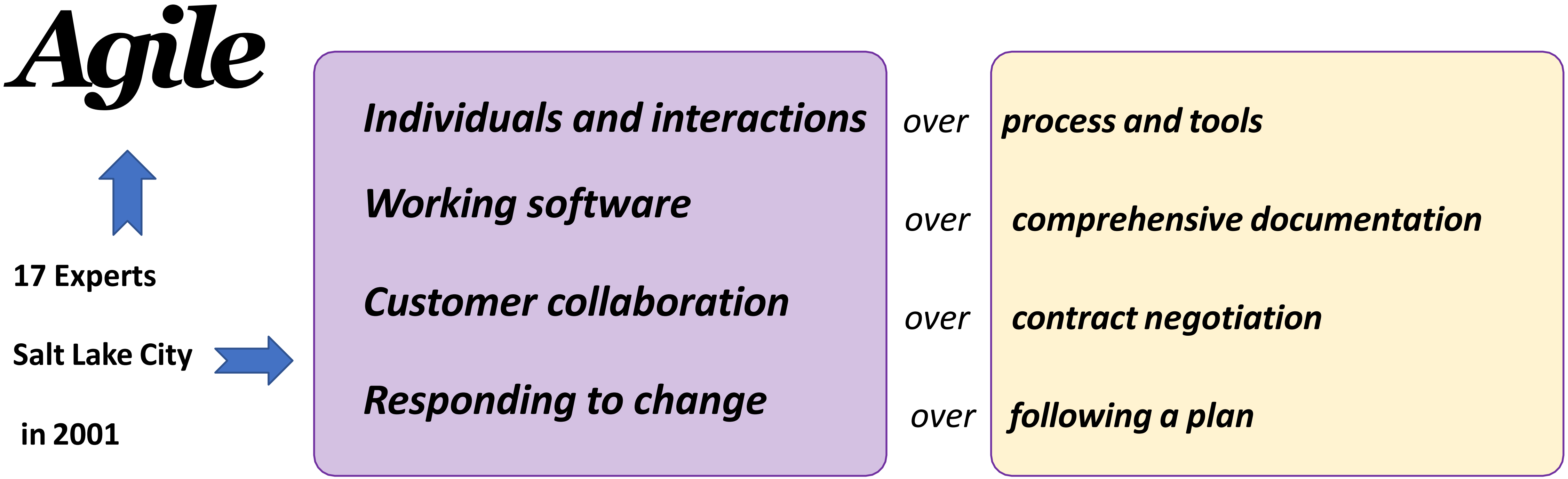
Lean

Adoption of Agile Methods



Reference: *https://www.agile247.pl/wp-content/uploads/2017/04/versionone-11th-annual-state-of-agile-report.pdf*

Manifesto for Agile Software Development



Agile

17 Experts

Salt Lake City

in 2001

Individuals and interactions *over* ***process and tools***

Working software *over* ***comprehensive documentation***

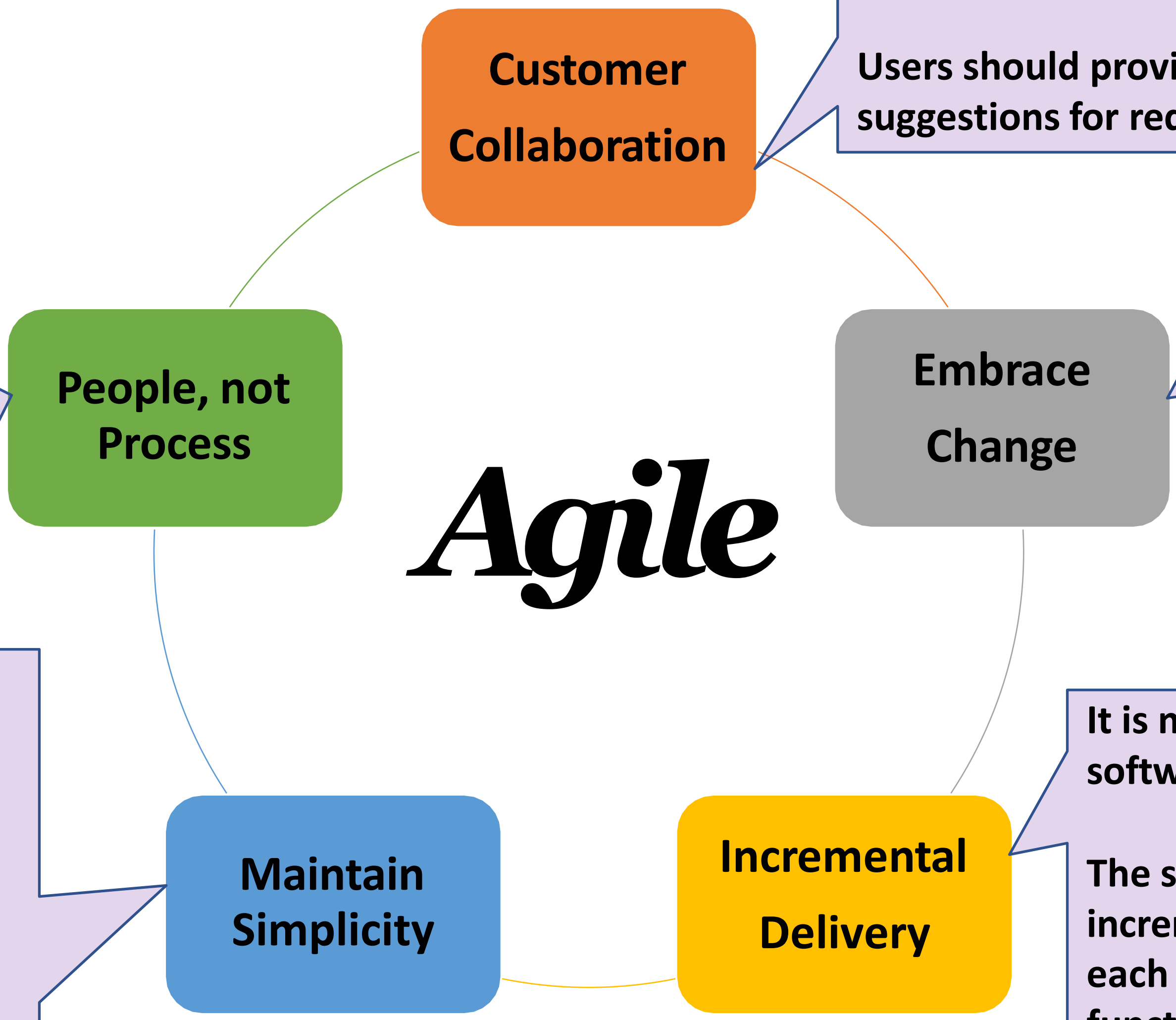
Customer collaboration *over* ***contract negotiation***

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.

***Source**: http://agilemanifesto.org*

Principles of Agile Methods



Agile

Customer Collaboration

Users should work closely with the development team.

Users should provide feedback on the system and suggestions for requirements/improvement.

Embrace Change

Changes to the requirements can happen at any time during the development.

Plans can quickly become inaccurate with the rapid change to the requirements.

Incremental Delivery

It is more important to deliver the software than following the plan.

The software should be developed incrementally and iteratively with each delivery having more functionalities included.

Maintain Simplicity

Keep everything simple, for both the software and the software process.

Focus on delivering valuable software to the customer rather than writing comprehensive documentation.

People, not Process

Focus on the people on the team. Tools and practices are second.

People should be left to develop their own ways of working, rather giving prescriptive processes.

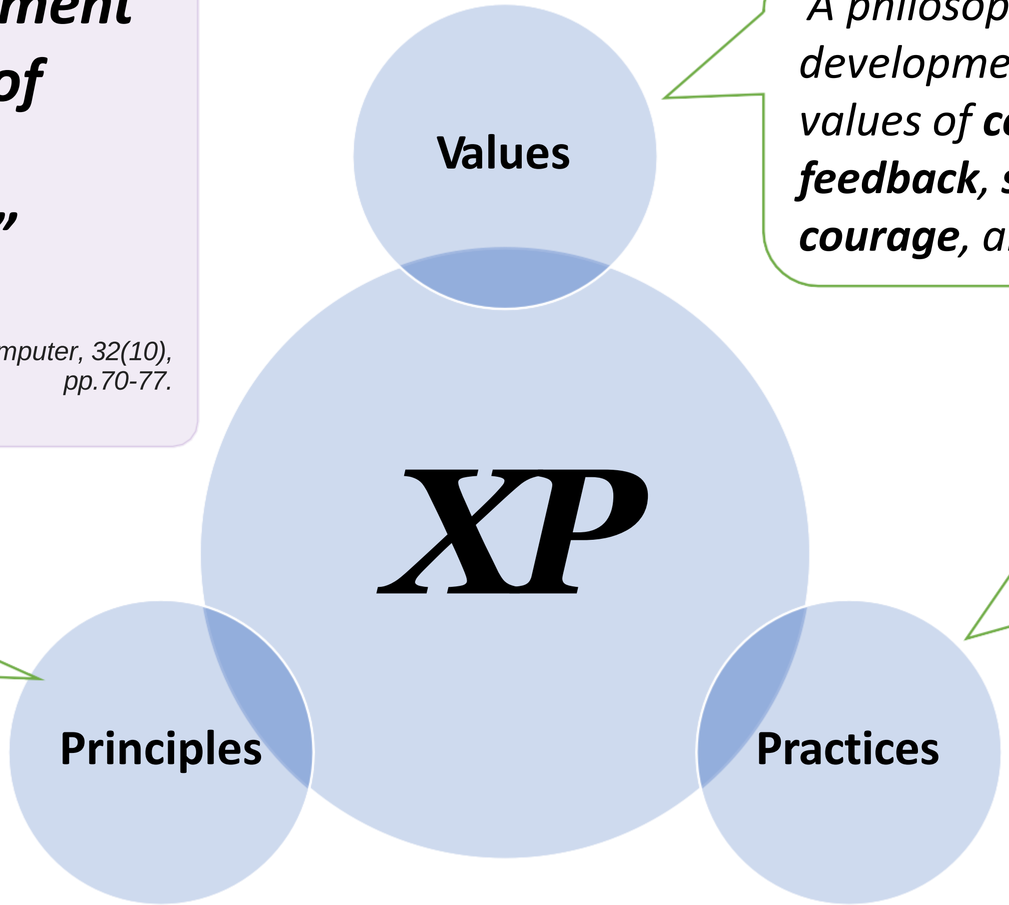
Exploit and explore skills and knowledges of team members and trust them.

***NOTE**: There are 11 principles from Agile Development (https://agilemanifesto.org/principles.html)*

XP (Extreme Programming) and **Scrum** are both well - known Agile software development methodologies, In practice, some teams may also combine elements of both XP and Scrum. For example, a team may follow the Scrum framework for project management while implementing XP’s engineering practices to improve code quality.

What is eXtreme Programming (XP)?

> ***“XP is a style of software development focusing on excellent application of programming techniques, clear communication, and teamwork…”***
>
> *-- Beck, K., 1999. Embracing change with extreme programming. Computer, 32(10), pp.70-77.*



*”A philosophy of software development based on the values of **communication**, **feedback**, **simplicity**, **courage**, and **respect**.“*

”A set of complementary principles, intellectual techniques for translating the values into practice...“

”A body of practices proven useful in improving software development.“

XP Principles (1)

1. Humanity
    - *Balance the needs of the individual with the needs of the team*
2. Economics
    - The time value of money
    - The option value of systems and teams
3. Mutual Benefit
    - *Every activity should benefit all concerned*
    - *Write automated tests that help the design and implementation better today; leave the test as the 'documentation' for future programmers*
    - *Refactor to improve simplicity, clarity and coherence*

"Software development is more valuable when it earns money sooner and spends money later"

-- Beck, K., 1999. Embracing change with extreme programming. Computer, 32(10), pp.70-77.

!!!"Extensive internal documentation of software is an example of a practice that violates mutual benefit."!!!

-- Beck, K., 1999. Embracing change with extreme programming. Computer, 32(10), pp.70-77.

XP Principles (2)

4. Self-Similarity
   - *Use the structure of one solution into a new context, even at different scales (it's a good place to start)*
5. Improvement
   - *Get an activity started right away, then refine the results over time.*

> ***"…the best (or perfect) is the enemy of the good (enough)"***
>
> ***-- a wise Italian***

6. Diversity
   - *Programmers should work together on the problem and all opinions should be valued*
7. Reflection
   4. Review and analyze why they succussed or failed

XP Principles (3)

8. Flow
   - Continuous flow of activities rather than discrete phase (small increments, continuous integration)
9. Opportunity
   - Seeing problems as opportunities for changes (personal growth, deepening relationships, and improved software)
10. Redundancy
    - *Difficult problems in software development should be solved in several different ways.*
11. Failure
    - If you have 3 ways to implement a user story, but you don’t know which to use, try all of them

XP Principles (4)

12. Quality
    - *Quality can be measured in defect, design quality, and the experience of development*
13. Baby Step
    - *Proceeds one test at a time, and integrates and test a few hours' worth of changes at a time*
14. Accepted Responsibility
    - *Responsibility cannot be assigned; it can only be accepted*

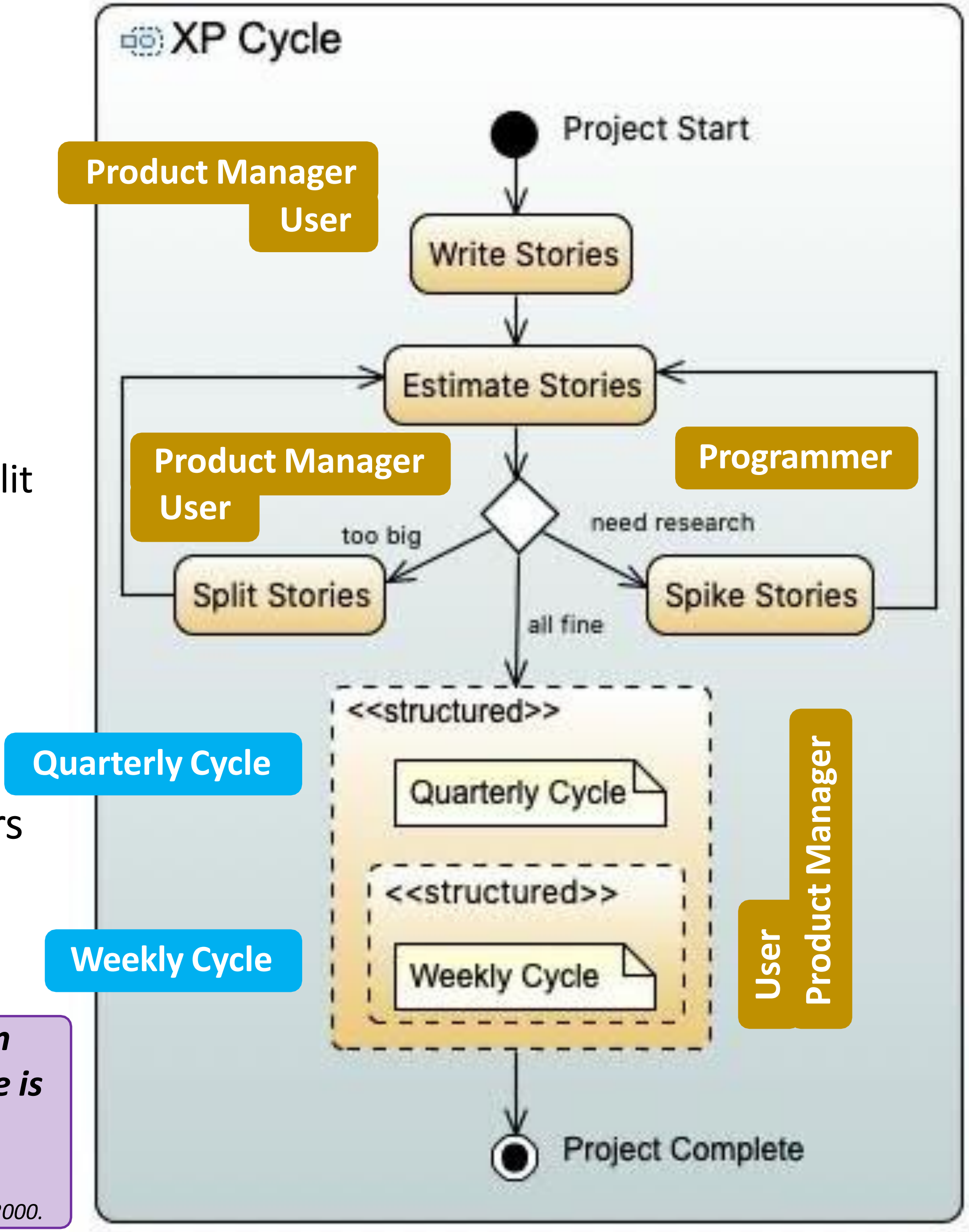
> ***"Projects don't go faster by accepting lower quality"***
>
> *-- Beck, K., 1999. Embracing change with extreme programming. Computer, 32(10), pp.70-77.*

XP Practices

XP Roles

XP Workflow

1. At the beginning of the project development, the *product manager* and the *user* write **Stories** (programmers may also be involved).
2. The *programmers* estimate the stories and tasks.
   - if a story is too big, the product manager and/or the user split the story; if the programmers don't understand the topic, initial a ***Spike***.
3. The product manager and/or the user decide on a "*theme*" (the big picture) for a Quarterly Cycle.
   - The process iterates until the project finishes.
4. The product manager, the users and/or the programmers pick a reasonable number of user stories for Weekly Cycle and add some ***Slacks***.
   - The process iterates until the end of the Quarterly Cycle

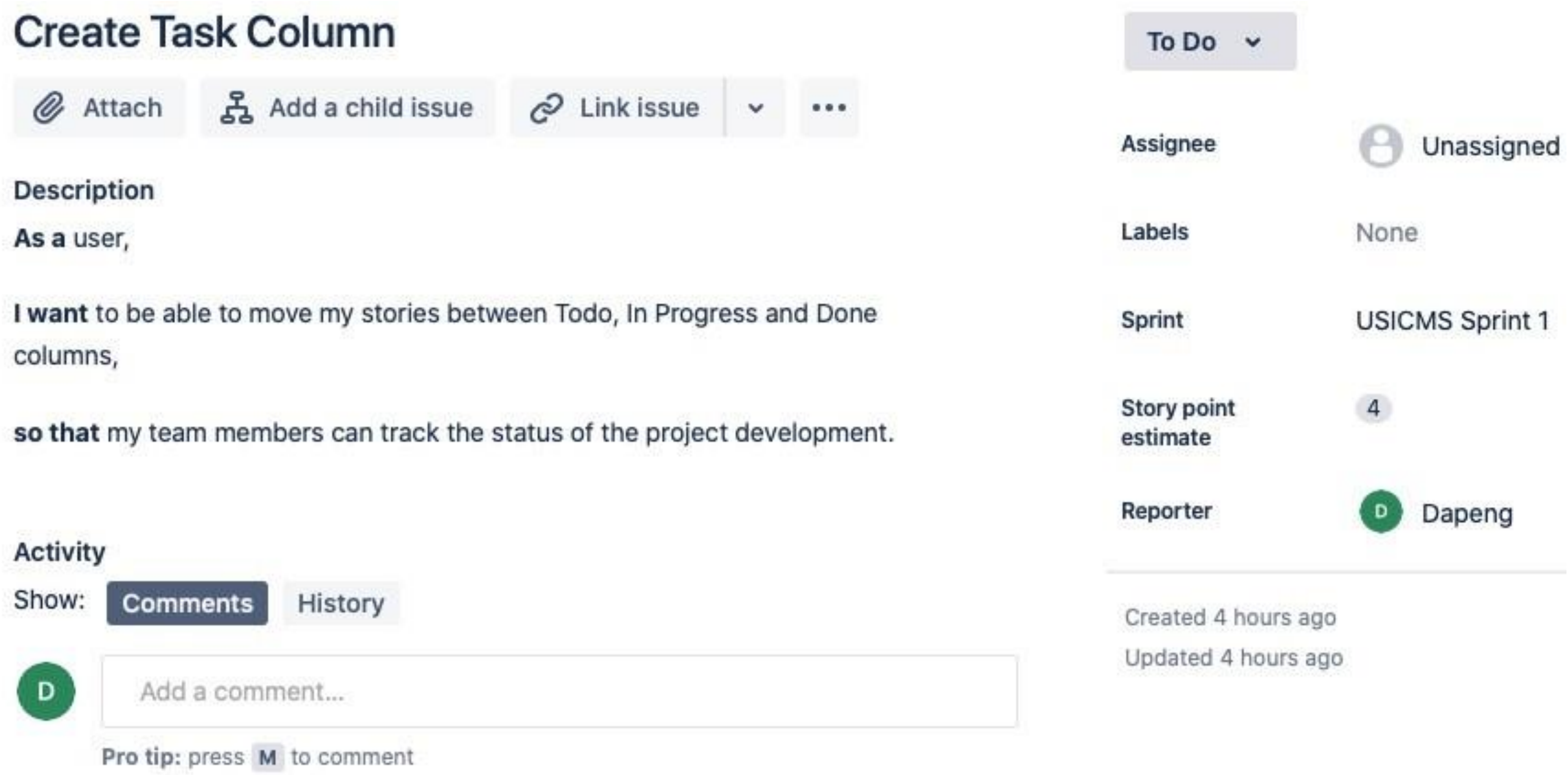


> ***"In any plan, include some minor tasks that can be dropped (Slacks) if you get behind."***
>
> *-- Beck, K., 1999. Embracing change with extreme programming. Computer, 32(10), pp.70-77.*

> ***"A spike is a skinny, minimal solution in throw-away code. The result of a spike is enough knowledge to attempt an estimate "***
>
> *-- William C. W., Extreme Programming Explored, 2000.*

Terminology: User Story

> ***"… a user story is a scenario of use that might be experienced by a system user."***
>
> *-- Sommerville, I., 2016. Software engineering., 10th Edition. Pearson Education.*



Create Task Column

Description

As a user,

I want to be able to move my stories between Todo, In Progress and Done columns,

so that my team members can track the status of the project development.

Activity

Show: Comments History

Add a comment...

Pro tip: press M to comment

Assignee	Unassigned
Labels	None
Sprint	USICMS Sprint 1
Story point estimate	4
Reporter	Dapeng

Created 4 hours ago
Updated 4 hours ago

Generated from JIRA

- Agile methods do not usually have a separate requirements engineering activity, rather integrate it with development.
- User stories are often written on user story index cards.

It addresses user needs

It can be used in planning system iterations.

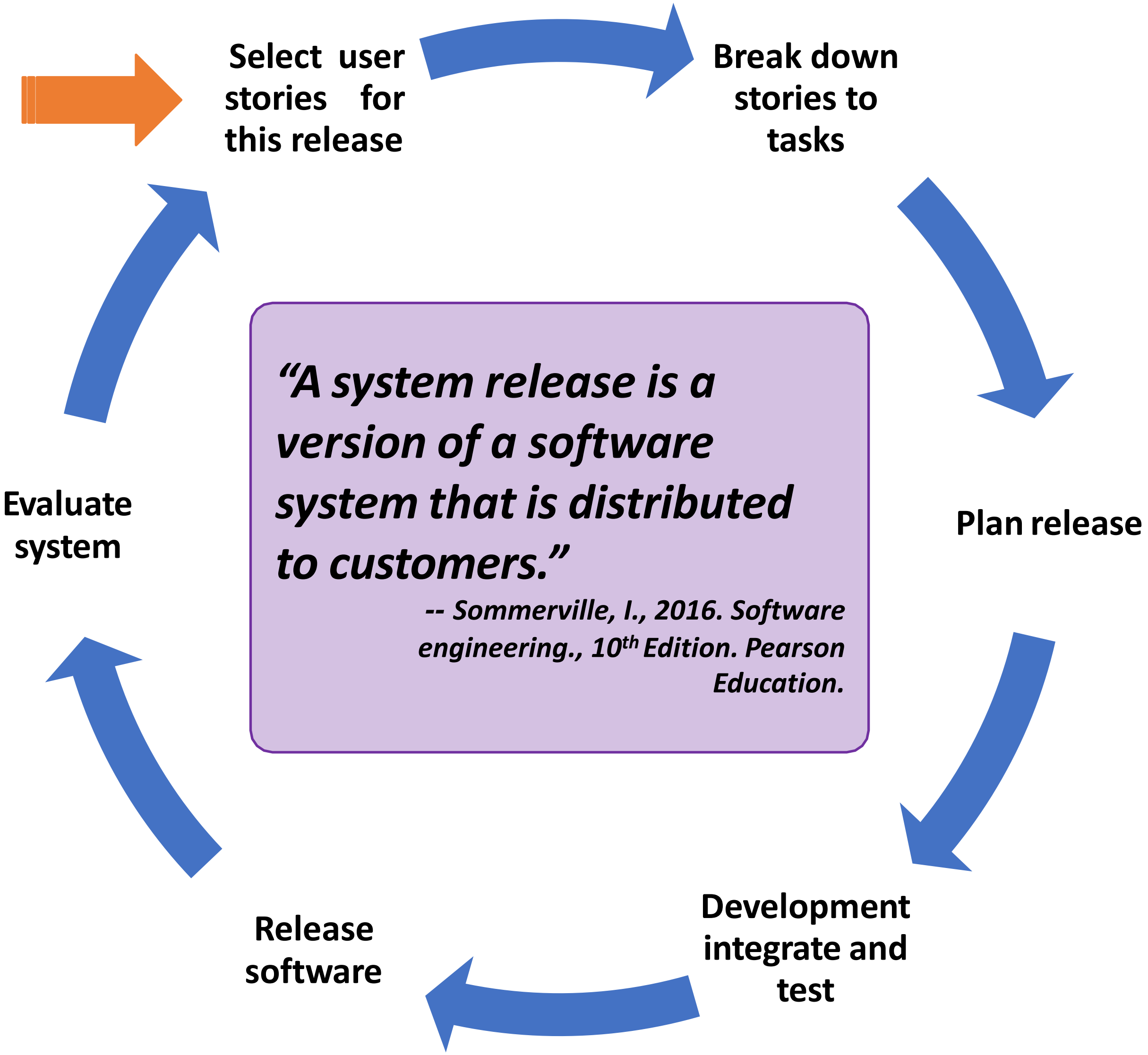
The use and/or the product manager prioritize the stories for implementation.

- High business value
- Significant impact on architectural design

Structured Template

- As a <role> (who wants to accomplish something)
- I want to <activity> (what)
- So that <value> (why)
- It may also include Acceptance Criteria and Comments

XP – Planning the Release


Select user stories for this release
Break down stories to tasks
Plan release
Development integrate and test
Release software
Evaluate system

> ***"A system release is a version of a software system that is distributed to customers."***
>
> *-- Sommerville, I., 2016. Software engineering., 10th Edition. Pearson Education.*

- How many user stories should we pick for each cycle?
  - depending on how many story-points the user/product manager expect for a cycle. The process is often called the *Declare the Velocity*.
- Who decides on the velocity?
  - It is usually the Tracker who tracks the iteration, the testing for acceptance, the code quality, customer management, etc.

XP Practices

XP Roles

XP Programmer Workflow (1)

Programmer

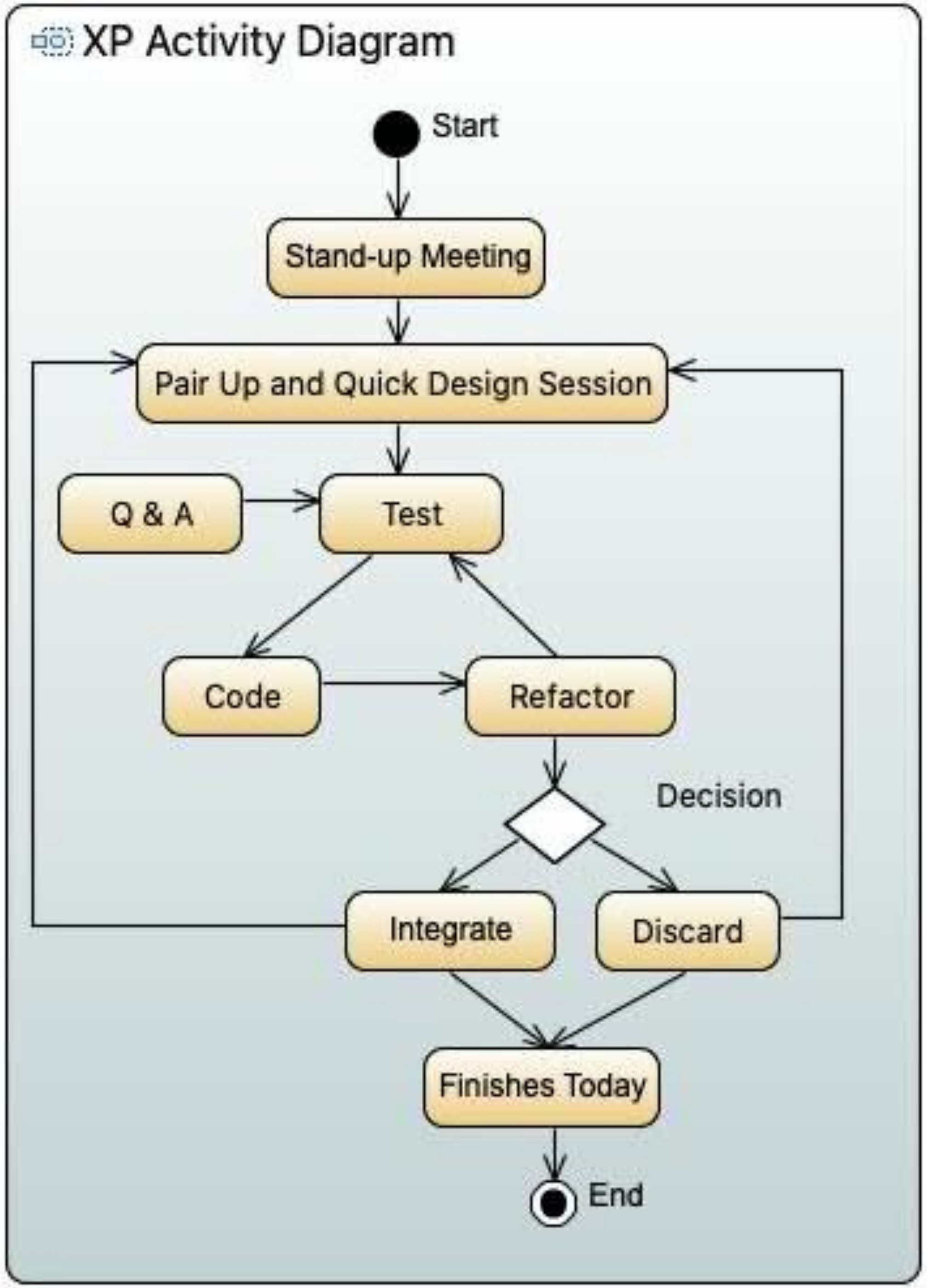
Incremental Design

Project Manager

1. Standup meeting (everyday in the morning, 10 minutes)
   - Identify problem, pick up the tasks that you want to do
2. Pair up with a colleague and do a quick design
   - All production code is produced by a pair **Pair Programming**
   - One programming, another thinking; switch roles periodically
3. Test
   - Write small unit-test code at a time
   - Test everything that could possibly break **Test-First Programming**
4. Code
   - Just write enough code to pass the unit-test
   - Using the team's coding standard
   - Ask users for feedback if there is a question
5. Refactor
   - The code should pass all unit-test, have no duplicate logic, ensure good coding practices

Refactoring

Energized Work



Reading (XP Practice): William C. W., Extreme Programming Explored, 2000.

XP Practices XP Roles

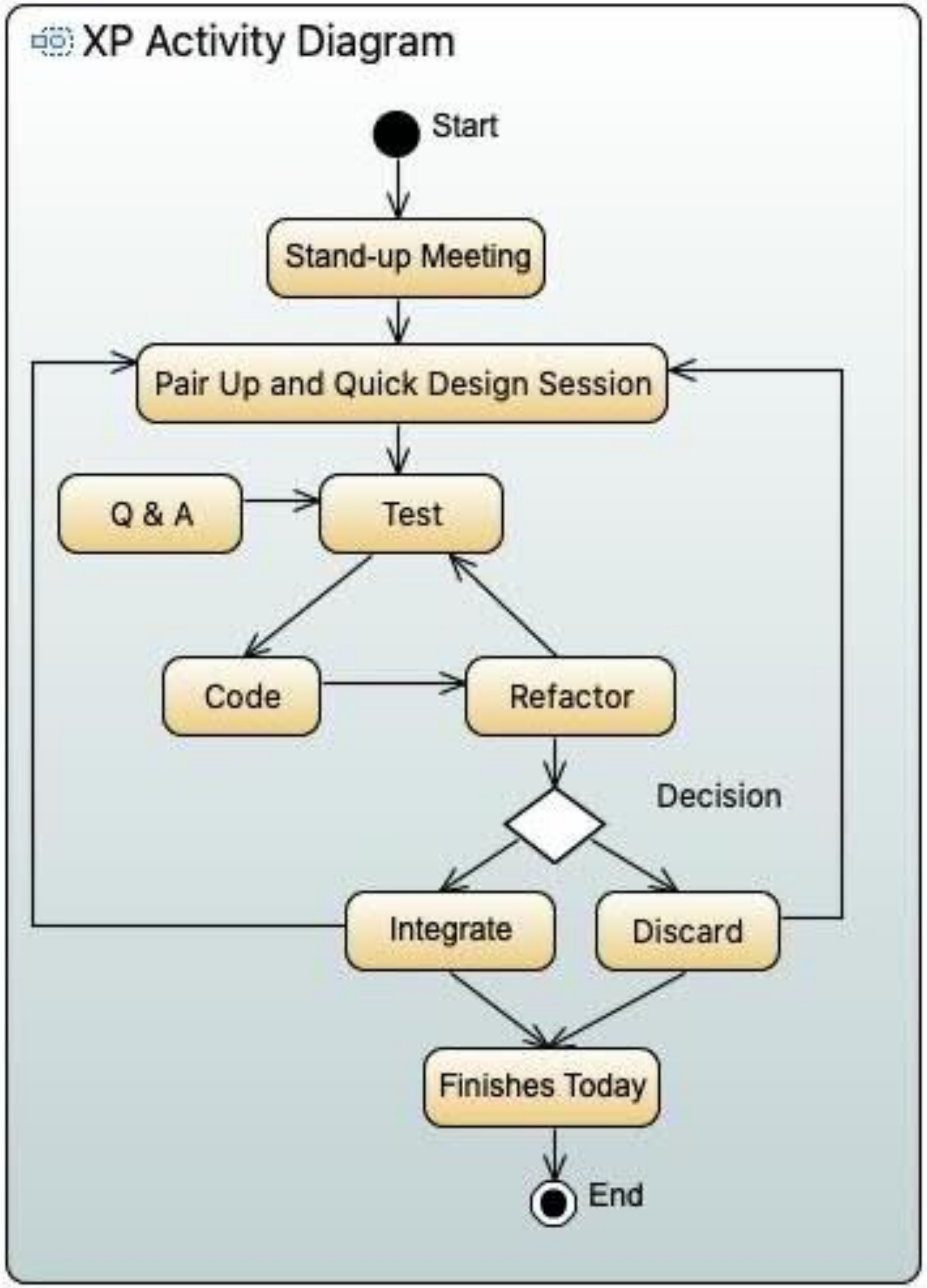
XP Programmer Workflow (2)

Programmer

Product Manager User

6. Q & A (Question and Answer) On-Site Customer
   - The user of the software system SHOULD be on-site to answer questions
   - The users and product manager should be able to make decisions
7. Integrate Ten-Minute Build Tester
   - Commit the code to the integration machine, build the system and run (pass) all tests

   Continuous Integration
8. Discard
   - If things don’t work, discard them
9. Return to ’Pair Up and Quick Design Session’
   - If you have enough time left in the day, you can work on anther task
10. Finishes today



Reading (XP Practice): William C. W., Extreme Programming Explored, 2000

XP Summary

Practices:

1. Energized Work
2. Pair Programming
3. Stories
4. Weekly Cycle
5. Quarterly Cycle
6. Slack
7. Ten-Minute Build
8. Continuous Integration
9. Test-First Programming
10. Incremental Design
11. Sit Together
12. Whole Team
13. Informative Workspace

Values:

1. Communication
2. Simplicity
3. Courage
4. Feedback
5. Respect

Roles:

1. Tester
2. Project Manager
3. Product Manager
4. User
5. Programmer
6. Tracker
7. Coach
8. Interaction Designer
9. Architect
10. Executive
11. Technical Writer
12. Human Resource

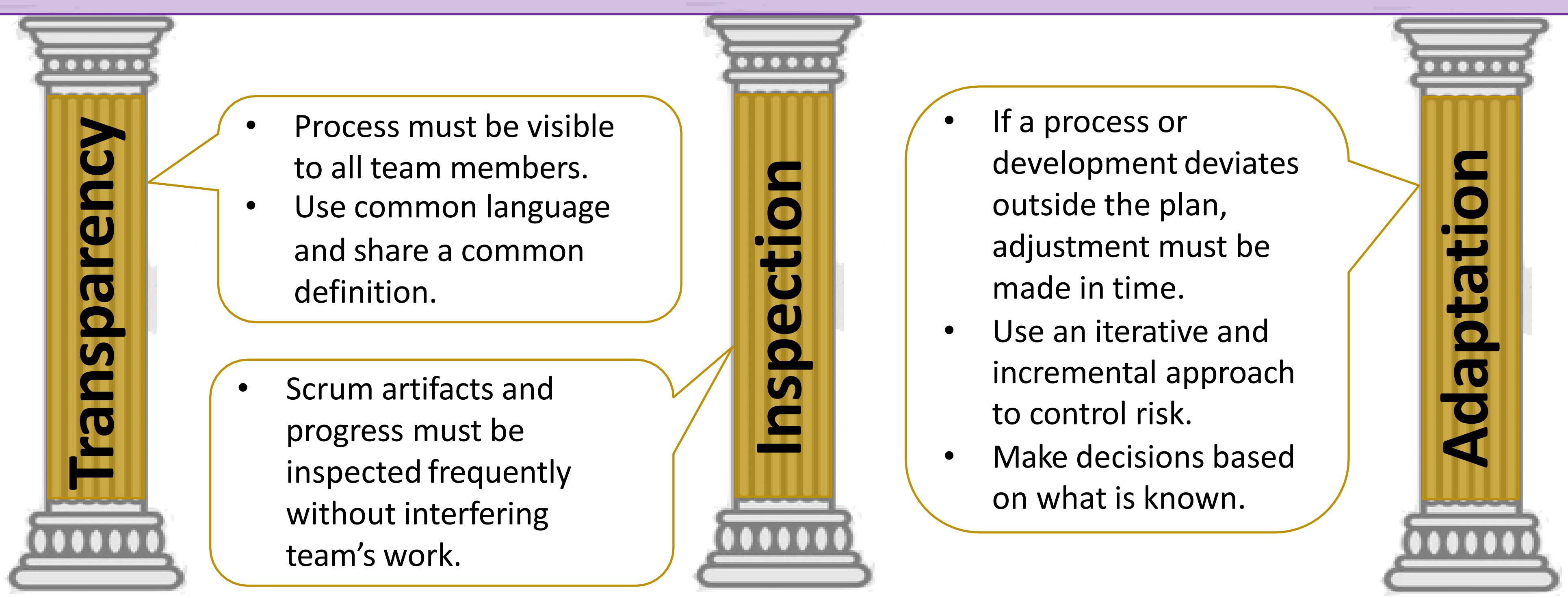
Principles:

1. Humanity
2. Economics
3. Mutual Benefit
4. Self-Similarity
5. Improvement
6. Diversity
7. Reflection
8. Flow
9. Opportunity
10. Redundancy
11. Failure
12. Quality
13. Baby Steps
14. Accepted Responsibility

Scrum

"A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value."

-- Schwaber, K. and Sutherland, J., 2011. The scrum guide. Scrum Alliance, 21, p.19.



Transparency

- Process must be visible to all team members.
- Use common language and share a common definition.

Inspection

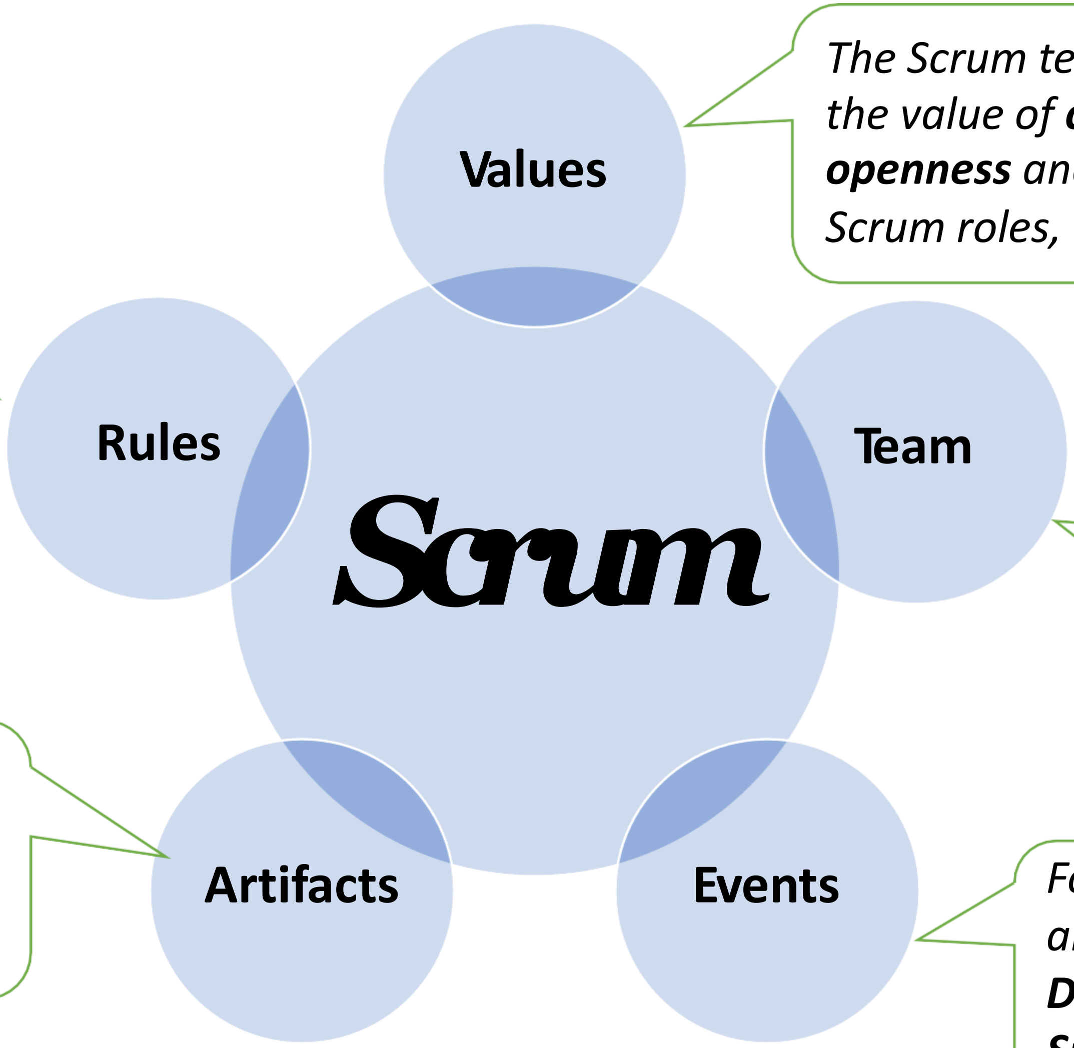
- Scrum artifacts and progress must be inspected frequently without interfering team's work.

Adaptation

- If a process or development deviates outside the plan, adjustment must be made in time.
- Use an iterative and incremental approach to control risk.
- Make decisions based on what is known.

Empiricism

The Components of Scrum Framework



The Scrum team members learn and explore the value of ***commitment****,* ***courage****,* ***focus****,* ***openness*** *and* ***respect*** *as they work with the Scrum roles, events, and artifacts.*

Binding all components together

A typical Scrum Team includes a ***Product Owner****, the* ***Development Team****, and a* ***Scrum Master****.*

- ***Product Backlog***
- ***Sprint Backlog***
- *Portfolio Backlog*
- *Program Backlog*

Four formal events are common to all Scrum teams, ***Sprint Planning****,* ***Daily Scrum****,* ***Sprint Review****, and* ***Sprint Retrospective****.*

Scrum Workflow -- Vision

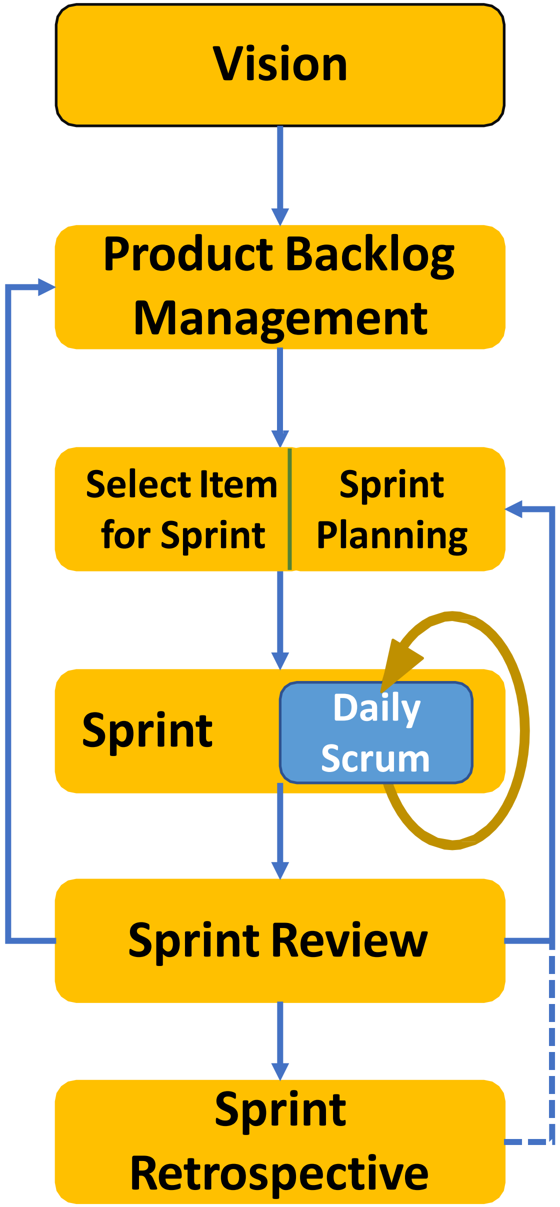
Who: Users, Product Owners, possibly Team members, and relevant stakeholders

Vision

Addresses:

- *What the problem does this software system solve?*
- *What the benefits does it provide?*
- *To whom does it provide?*
- *What the high-level system features are?*
- *What platforms, regulations, standards and constraints will it support and comply with?*
- *And other high-level business goals.*

Outcomes: *features, functions, requirements; needs brainstorming and possibly encapsulated in series of **user stories**.*



Scrum Workflow – Product Backlog Management

Who:

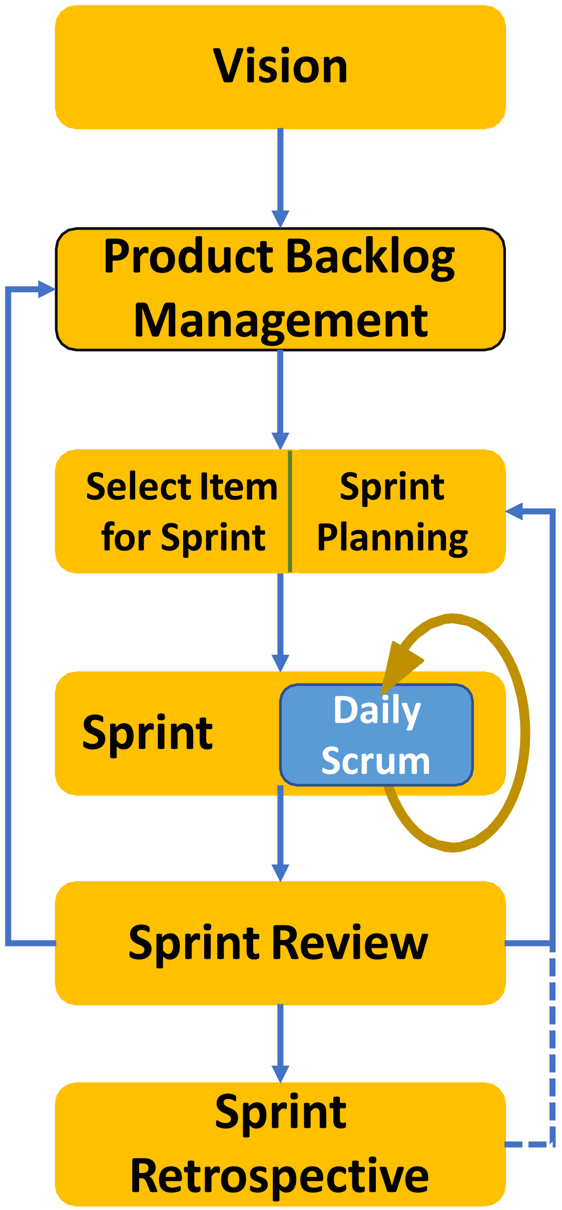
- The **Product Owner** manages the Product Backlog, i.e., the Product Backlog items, availability and ordering.
- The **Development Team** is responsible for all estimates, but all changes must be made by the product owner.

Product Backlog Management

What are in a Product Backlog?

- *all features, functions, enhancements and fixes*
- *the suggestions may be received from users after reviewing, sprint reviews or retrospectives*

***Outcomes:** a **Product Backlog**, the single source of requirements for any changes to be made to the product.*



Vision

Product Backlog Management

Select Item for Sprint

Sprint Planning

Sprint

Daily Scrum

Sprint Review

Sprint Retrospective

Product Backlog

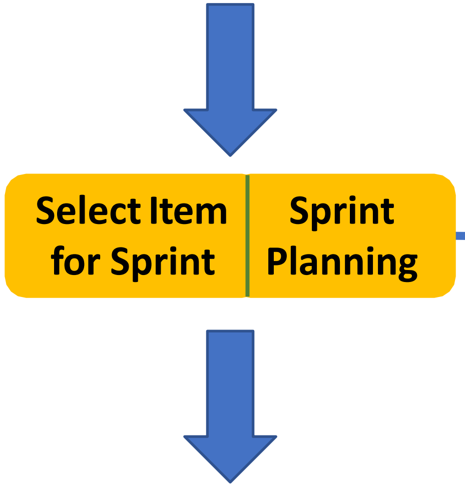
> ***”The Product Backlog is dynamic; it constantly changes to identify what the product needs to be appropriate, competitive, and useful.”***
>
> *-- Schwaber, K. and Sutherland, J., 2011. The scrum guide. Scrum Alliance, 21, p.19.*

- Each Product Backlog item should include the attributes of a description, order (priority), estimate, and value.
- Each item should include an acceptance criterion for assessing whether it is “*Done*”.
- Items in the Product Backlog may be grouped by their attribute as *themes* or *epics*.
- Higher ordered items often have clearer and more detailed descriptions as they are better understood.

Scrum Workflow – Sprint Planning

Who:

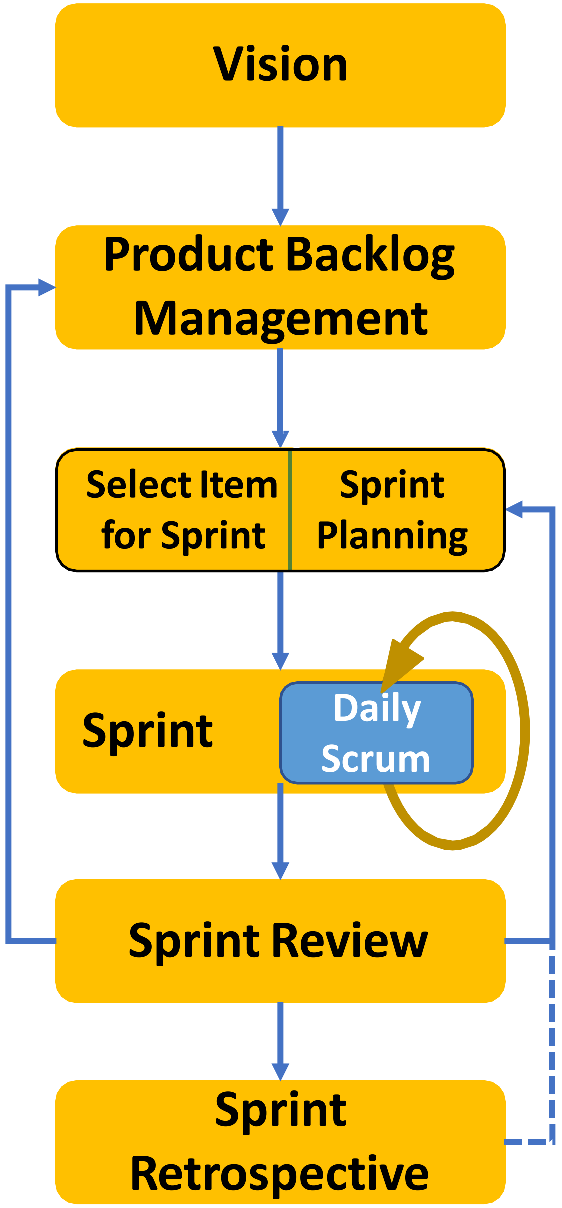
- The entire Scrum Team collectively create a plan for an iteration worth of work.
- The Product Owner gives the objectives for this iteration.
- The Development Team select items from the Product Backlog (NO ONE tells the team what to do).



Addresses:

- *What can be delivered after the iteration of the work?*
- *What will be needed for the iteration?*
- *Sprint planning is timeboxed to a maximum 8 hours for a one-month iteration, and shorter for shorter iterations.*

Outcomes: *a Sprint Backlog, consists of the selected items and the plan.*



Sprint Backlog

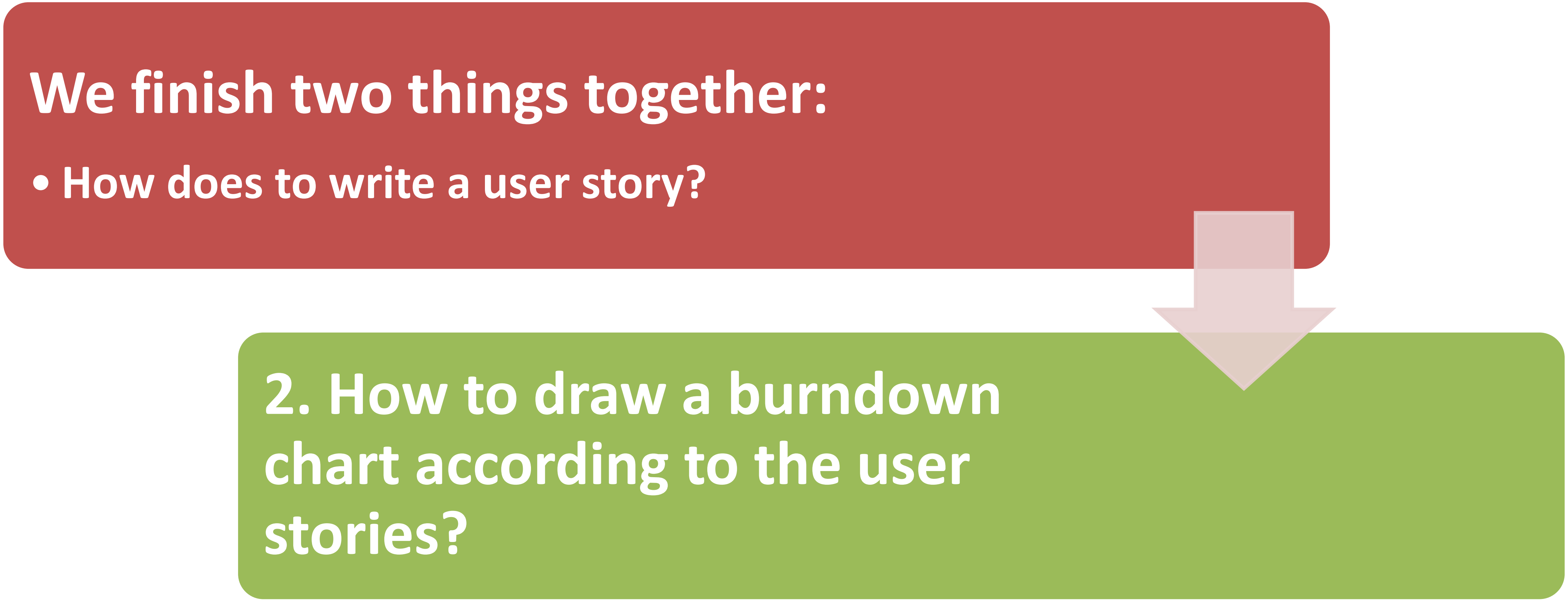
”The Sprint Backlog is the set of Product Backlog items selected for the Sprint, plus a plan for delivering the product increment and realizing the Sprint Goal.”

-- Schwaber, K. and Sutherland, J., 2011. The scrum guide. Scrum Alliance, 21, p.19.

- Items in a Sprint Backlog are chosen to meet the Sprint goal.
- A sprint Backlog should include at least one high priority process improvement identified in the retrospective meeting of the previous Sprint.
- The plan in the Sprint Backlog should have sufficient details to guide the Daily Scrum.
- Items in a Sprint Backlog can be changed, but can only be changed by the Development Team
- Sprint Backlog can be used to monitor Sprint progress.

XP VS Scrum

Similarities	Differences
Agile Principles Both XP and Scrum are based on Agile values and principles. They emphasize customer collaboration, responding to change over following a plan, and delivering working software frequently. For example, in both methodologies, the customer is involved throughout the development process to provide feedback and prioritize requirements.	**Process Focus** **Scrum**: It is more focused on the project management and process framework. Scrum defines specific roles (Product Owner, Scrum Master, Development Team), events (Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective), and artifacts (Product Backlog, Sprint Backlog, Increment). The main goal is to manage the project in a structured way and ensure that the team is making progress towards the product goal. **XP**: XP is more focused on engineering practices. It includes practices such as pair programming, test - driven development (TDD), continuous integration, and code refactoring. XP aims to improve the quality of the code and the development process by implementing these engineering - centric practices.
Iterative and Incremental Development They both adopt an iterative and incremental approach to software development. Instead of trying to build the entire software system in one go, they break the project into smaller chunks. In Scrum, these chunks are called sprints, and in XP, the development also occurs in short iterations. This allows for early and continuous delivery of value to the customer.	**Documentation** **Scrum**: While it values working software over comprehensive documentation, it still has some necessary documentation in the form of backlogs and reports related to the sprints. However, the emphasis is on keeping the documentation lightweight and useful for the team to manage the project. **XP**: XP generally advocates for less documentation. Since it relies on practices like pair programming and continuous communication within the team, the need for extensive written documentation is reduced. The code itself is considered the primary documentation in many cases.
Team - Centric Both methodologies place a strong emphasis on the team. In Scrum, the Scrum Team (including the Product Owner, Scrum Master, and Development Team) works together closely. Similarly, XP promotes a close - knit team environment where developers, customers, and other stakeholders collaborate effectively.	**Flexibility and Adaptability** **Scrum**: Scrum provides a relatively fixed framework with well - defined rules and ceremonies. While there is room for adaptation within the framework, changes to the core Scrum elements are not encouraged lightly. **XP**: XP is more flexible in terms of its practices. Teams can choose to adopt the XP practices that are most suitable for their project and context. For example, a team may decide to use only some of the XP practices like TDD and pair programming, depending on their needs.



We finish two things together:

- How does to write a user story?

2. How to draw a burndown chart according to the user stories?

User Stories

Before you start Sprint 2…investigate and generate User Stories

A set of ‘conversations’ about how users might interact with the software

As a < type of user >, I want < some goal > so that < some reason >

As an editing Lecturer, I want to be able to share documents without using moodle so that I have a more robust method to disseminate notes

200 example user stories on moodle From: https://www.mountaingoatsoftware.com/uploads/documents/example-user-stories.pdf

News

- As a site visitor, I can read current news on the home page so that I stay current on agile news.
- As a site visitor, I can access old news that is no longer on the home page, so I can access things I remember from the past or that others mention to me.
- As a site visitor, I can email news items to the editor, so they can be considered for publication. (Note: this could just be an email link to the editor.)
- As a site a site editor, I can set the following dates on a news item: Start Publishing Date, Old News Date, Stop Publishing Date so articles are published on and through appropriate dates. These dates refer to the date an item becomes visible on the site (perhaps next Monday), the date it stops appearing on the home page, and the date it is removed from the site (which may be never).
- As a site member, I can subscribe to an RSS feed of news (and events?) so I remain sufficiently and easily informed.
- As a site editor, I can assign priority numbers to news items, so I can indicate which articles I want featured most prominently on the site. Note: Items are displayed on the front page based on priority.

User Stories

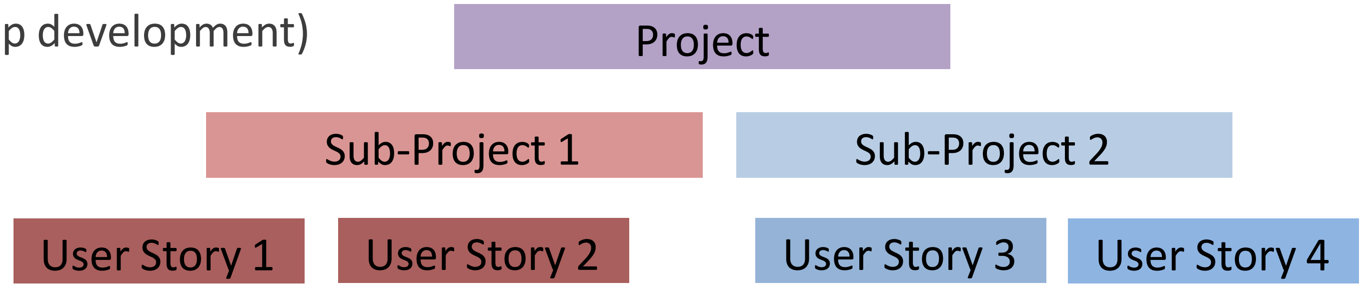
User stories (and providing he code to satisfy them) are the smallest discrete piece of work in an agile framework

A general explanation of a software feature written from the perspective of the end user

Stories that are accepted as a task will need to be ‘burned down’

Several user stories can be combined into a larger piece of work.

(Bottom up development)



Project

Sub-Project 1

Sub-Project 2

User Story 1

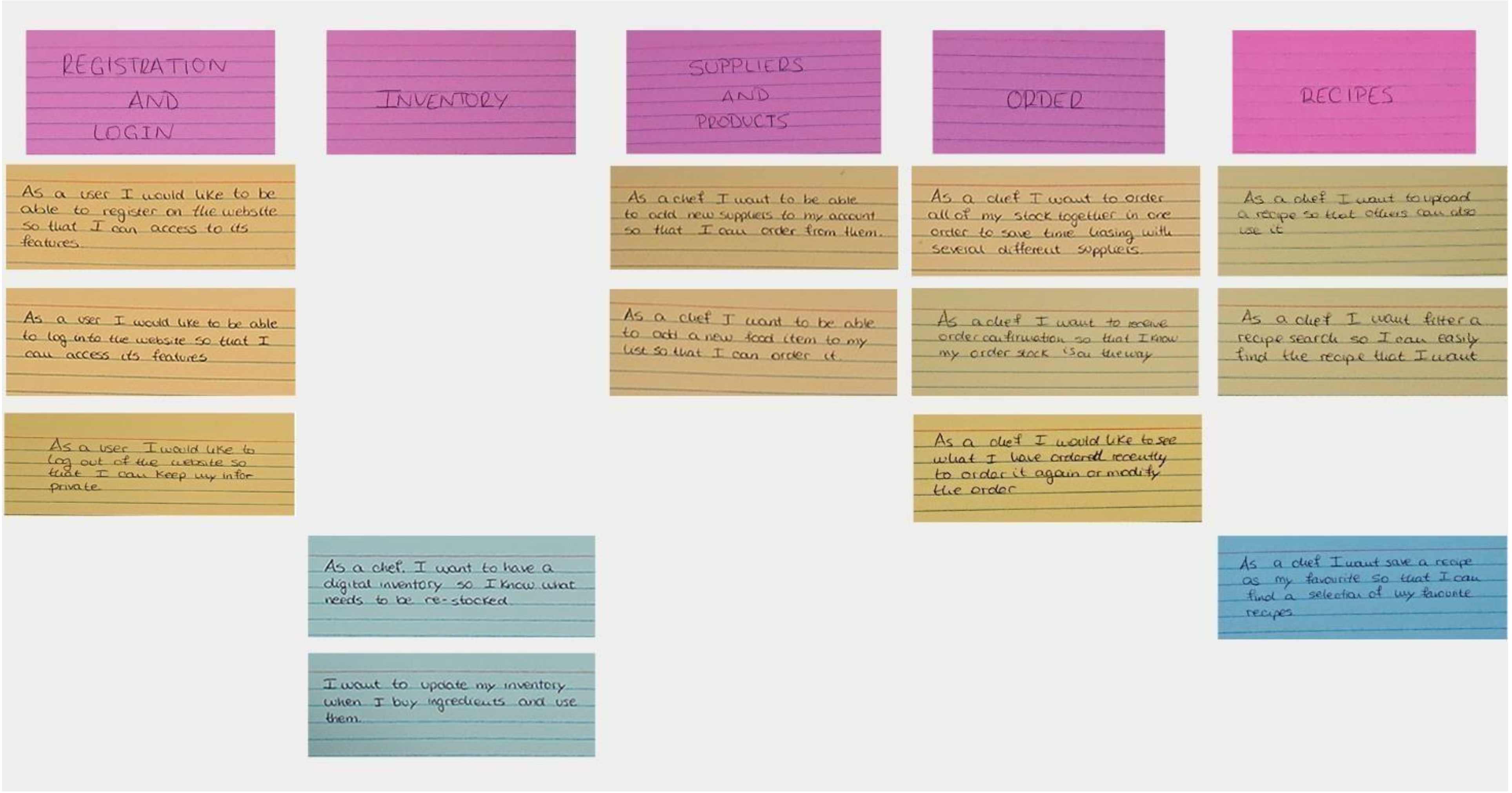
User Story 2

User Story 3

User Story 4

Example for the first step of setting up story map

Time



REGISTRATION
AND
LOGIN

As a user I would like to be able to register on the website so that I can access to its features.

As a user I would like to be able to log into the website so that I can access its features

As a user I would like to log out of the website so that I can keep my infor private

INVENTORY

As a chef. I want to have a digital inventory so I know what needs to be re-stocked

I want to update my inventory when I buy ingredients and use them.

SUPPLIERS
AND
PRODUCTS

As a chef I want to be able to add new suppliers to my account so that I can order from them.

As a chef I want to be able to add a new food item to my list so that I can order it.

ORDER

As a chef I want to order all of my stock together in one order to save time liasing with several different suppliers.

As a chef I want to receive order confirmation so that I know my order stock is on the way

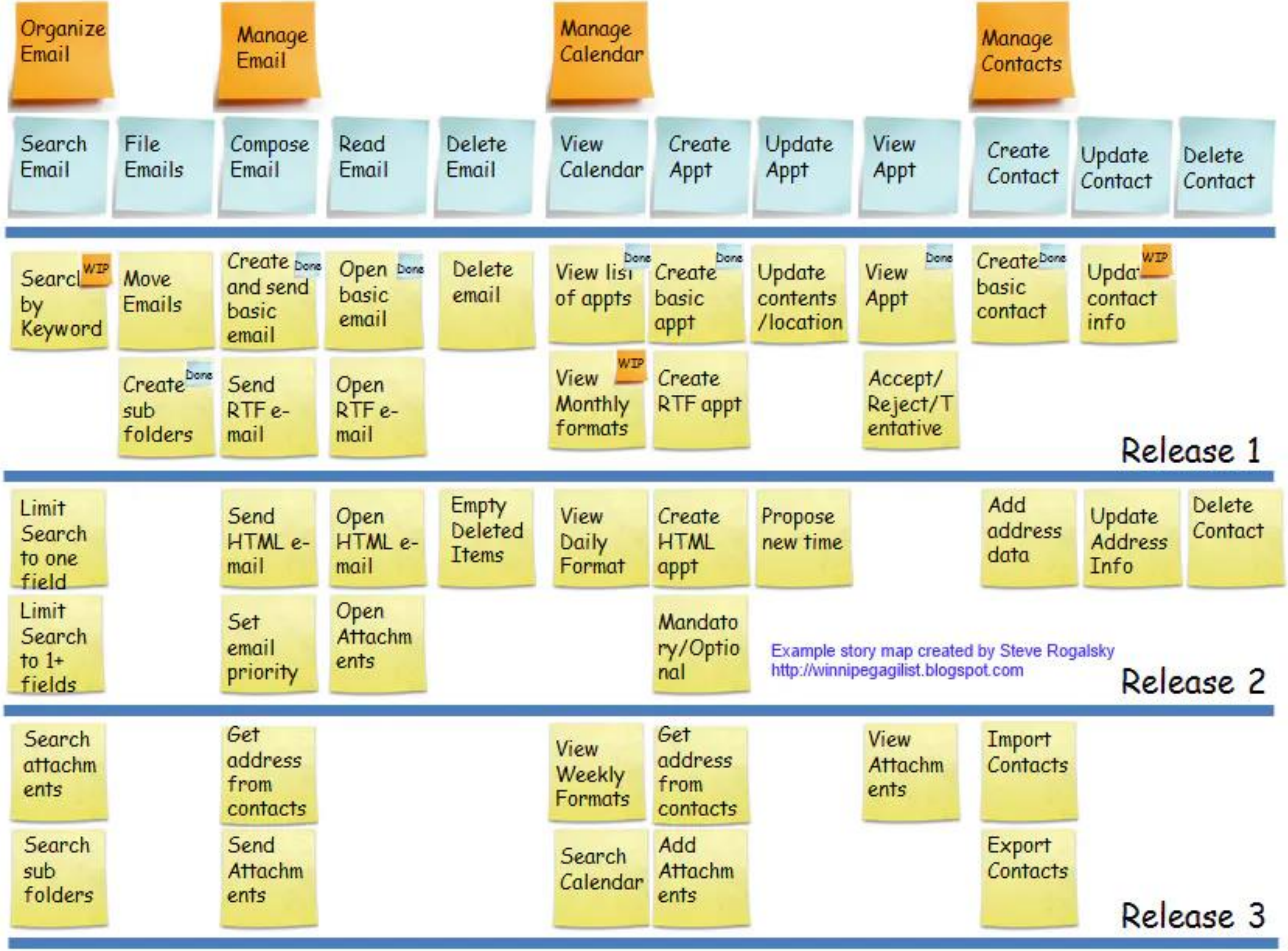
As a chef I would like to see what I have ordered recently to order it again or modify the order

RECIPES

As a chef I want to upload a recipe so that others can also use it

As a chef I want filter a recipe search so I can easily find the recipe that I want

As a chef I want save a recipe as my favourite so that I can find a selection of my favourite recipes

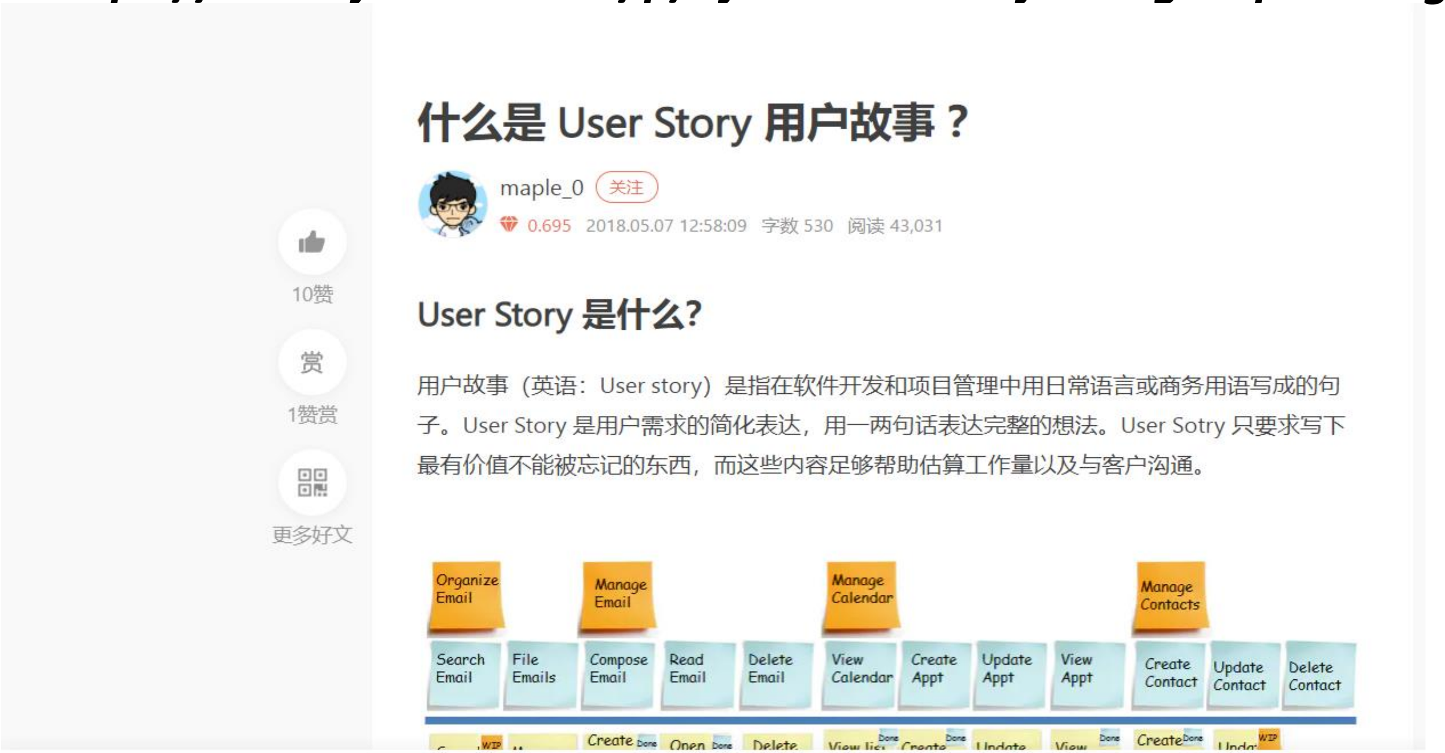


Organize Email		Manage Email			Manage Calendar				Manage Contacts			
Search Email	File Emails	Compose Email	Read Email	Delete Email	View Calendar	Create Appt	Update Appt	View Appt	Create Contact	Update Contact	Delete Contact	
Search by Keyword WIP	Move Emails Create sub folders Done	Create and send basic email Done Send RTF e-mail	Open basic email Done Open RTF e-mail	Delete email	View list of appts Done View Monthly formats WIP	Create basic appt Done Create RTF appt	Update contents /location	View Appt Done Accept/Reject/Tentative	Create basic contact Done	Update contact info WIP		Release 1
Limit Search to one field Limit Search to 1+ fields		Send HTML e-mail Set email priority	Open HTML e-mail Open Attachments	Empty Deleted Items	View Daily Format	Create HTML appt Mandatory/Optional	Propose new time		Add address data	Update Address Info	Delete Contact	Release 2
Search attachments Search sub folders		Get address from contacts Send Attachments			View Weekly Formats Search Calendar	Get address from contacts Add Attachments		View Attachments	Import Contacts Export Contacts			Release 3

Example story map created by Steve Rogalsky
http://winnipegagilist.blogspot.com

A good explanation for how to set up user story map (in Chinese)

https://www.jianshu.com/p/7fb8104c5cec?from=groupmessage



什么是 User Story 用户故事？

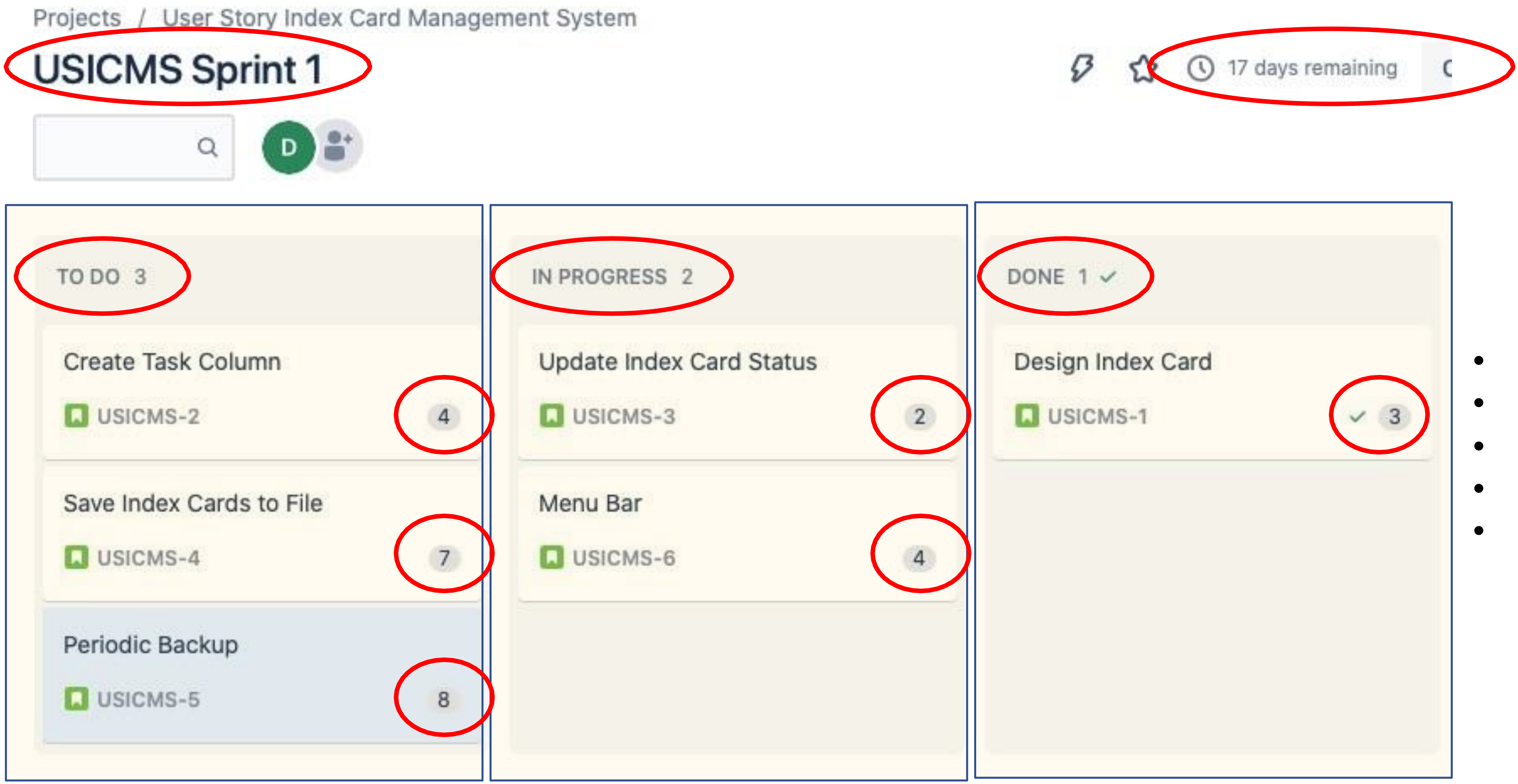
maple_0 关注

0.695 2018.05.07 12:58:09 字数 530 阅读 43,031

User Story 是什么?

用户故事（英语：User story）是指在软件开发和项目管理中用日常语言或商务用语写成的句子。User Story 是用户需求的简化表达，用一两句话表达完整的想法。User Sotry 只要求写下最有价值不能被忘记的东西，而这些内容足够帮助估算工作量以及与客户沟通。

Manage User Stories in Sprint Backlog



Projects / User Story Index Card Management System

USICMS Sprint 1

17 days remaining

TO DO 3

- Create Task Column — USICMS-2 — 4
- Save Index Cards to File — USICMS-4 — 7
- Periodic Backup — USICMS-5 — 8

IN PROGRESS 2

- Update Index Card Status — USICMS-3 — 2
- Menu Bar — USICMS-6 — 4

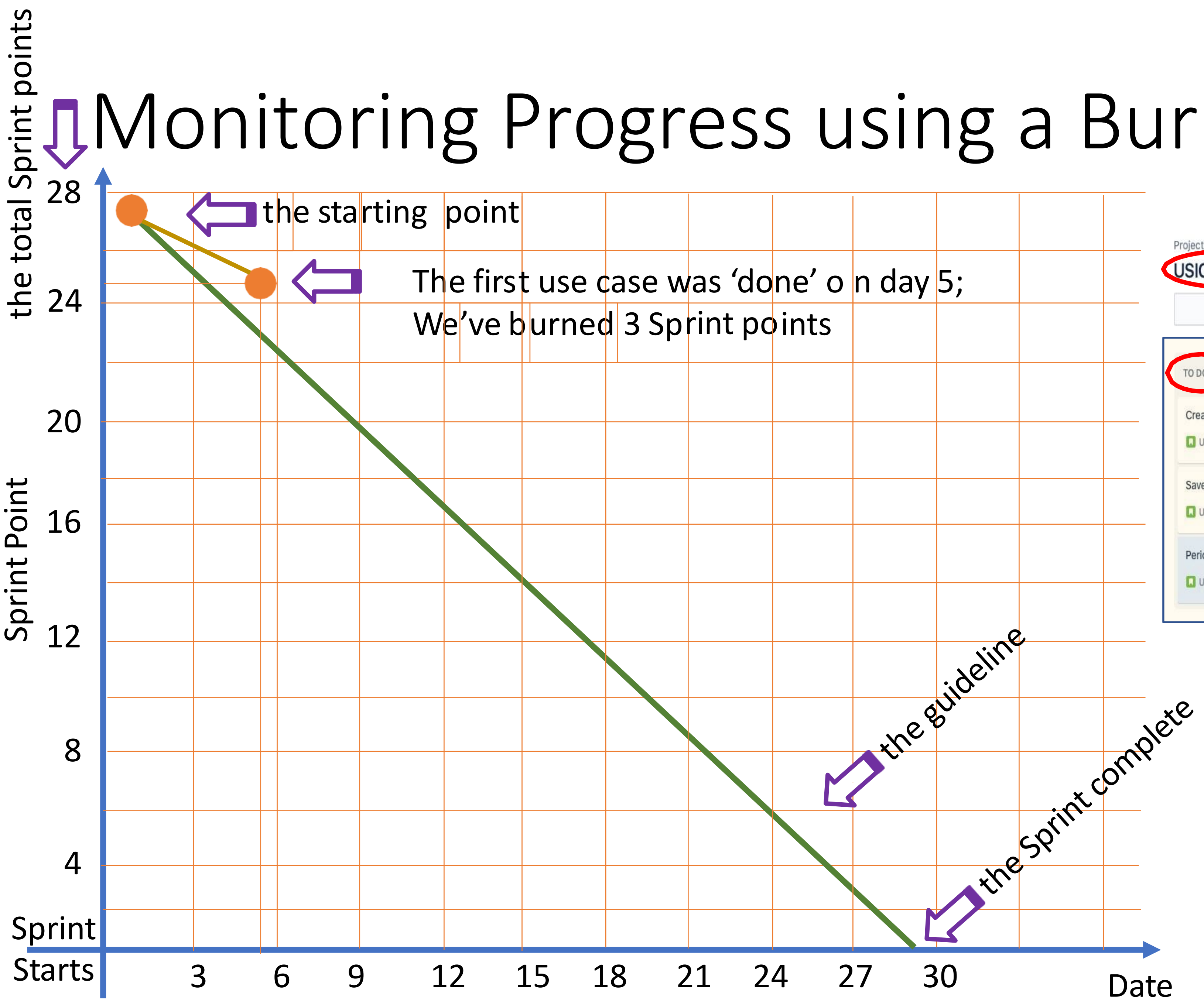
DONE 1 ✓

- Design Index Card — USICMS-1 — ✓ 3

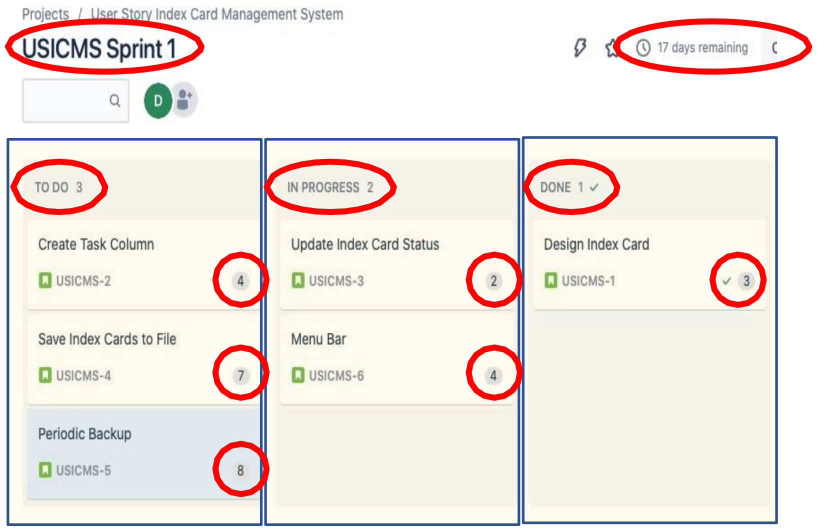
- The first iteration
- 4 Weeks
- 17 Days remaining
- 6 User Stories
- 28 points

Generated from JIRA

Monitoring Progress using a Burndown Chart



The first use case was ‘done’ o n day 5;
We’ve burned 3 Sprint points



Projects / User Story Index Card Management System

USICMS Sprint 1

17 days remaining

TO DO 3

- Create Task Column — USICMS-2 — 4
- Save Index Cards to File — USICMS-4 — 7
- Periodic Backup — USICMS-5 — 8

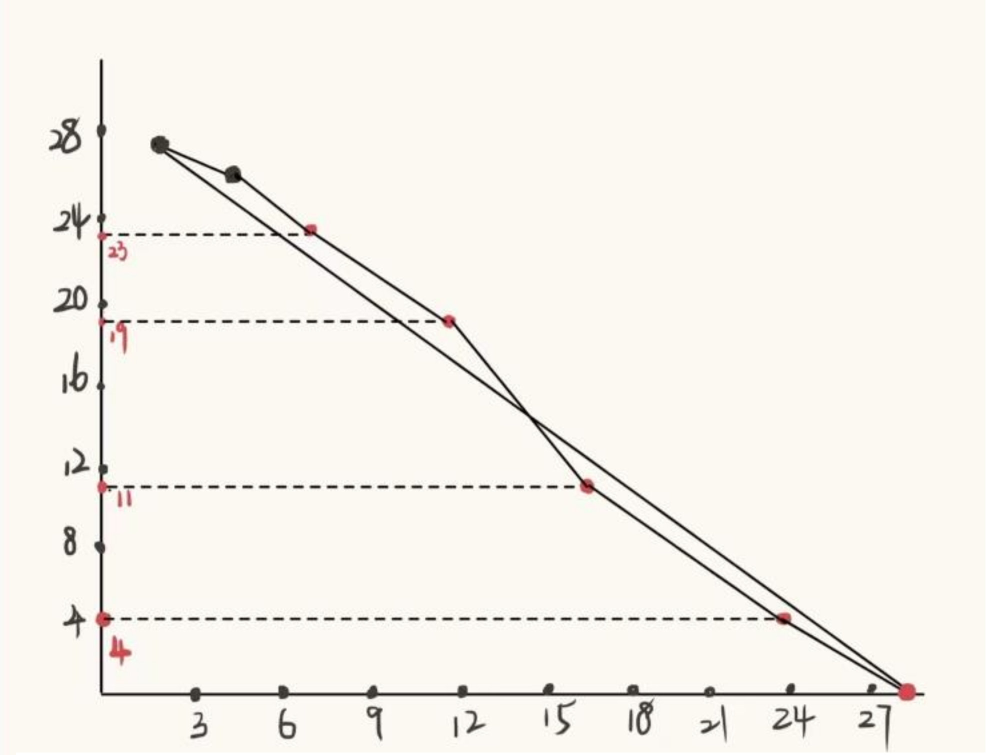
IN PROGRESS 2

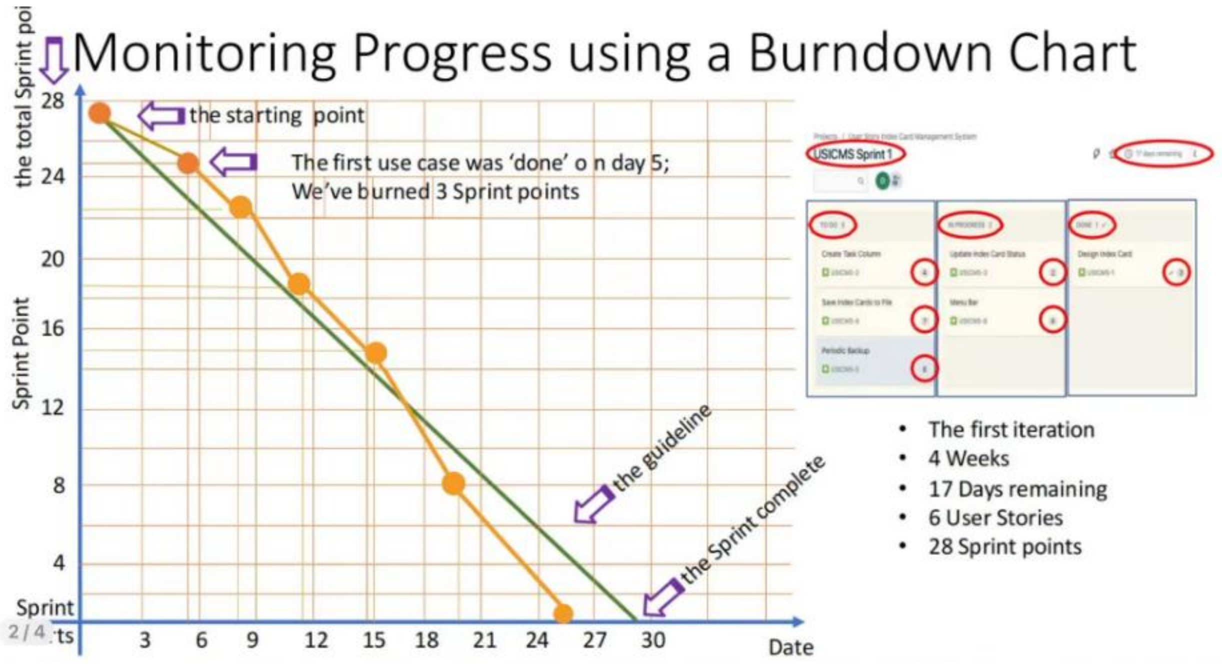
- Update Index Card Status — USICMS-3 — 2
- Menu Bar — USICMS-6 — 4

DONE 1 ✓

- Design Index Card — USICMS-1 — ✓ 3

- The first iteration
- 4 Weeks
- 17 Days remaining
- 6 User Stories
- 28 Sprint points



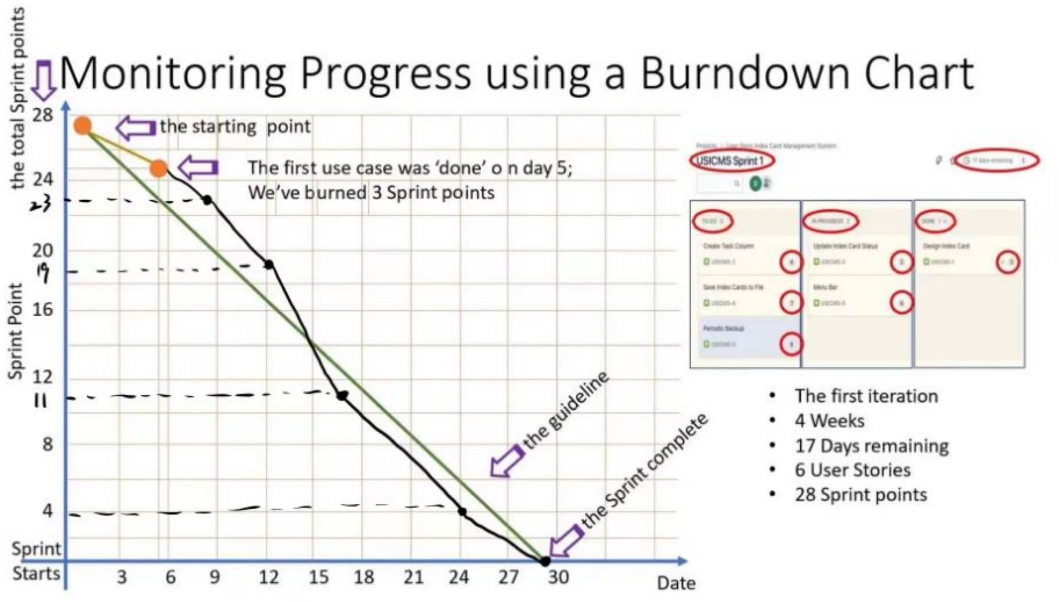


Monitoring Progress using a Burndown Chart

the starting point

The first use case was 'done' o n day 5;
We've burned 3 Sprint points

- The first iteration
- 4 Weeks
- 17 Days remaining
- 6 User Stories
- 28 Sprint points

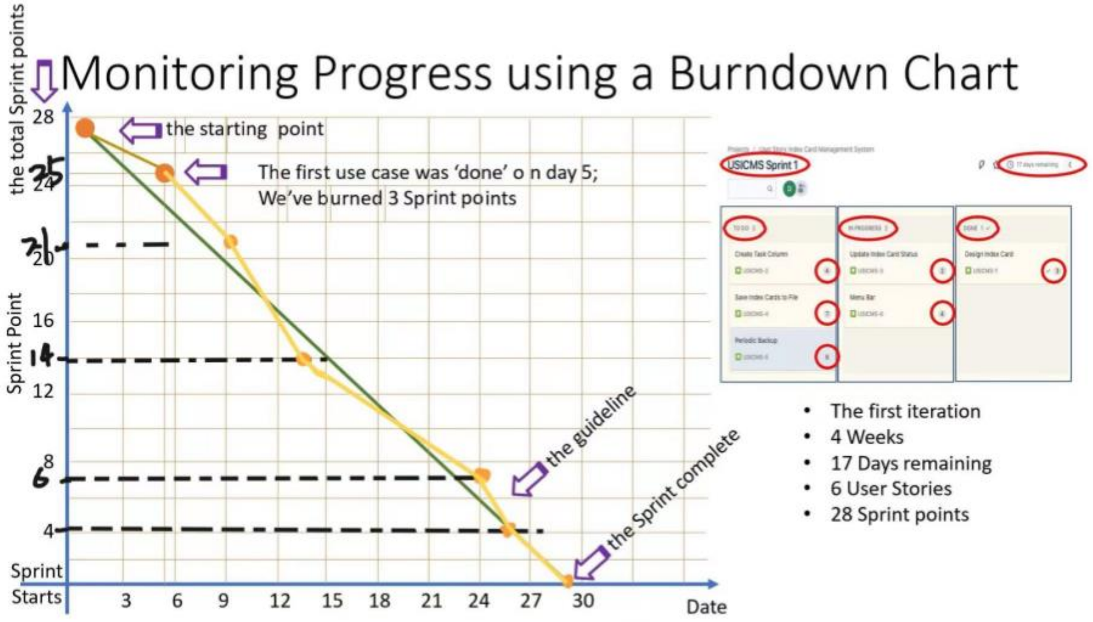


Monitoring Progress using a Burndown Chart

the starting point

The first use case was 'done' o n day 5;
We've burned 3 Sprint points

- The first iteration
- 4 Weeks
- 17 Days remaining
- 6 User Stories
- 28 Sprint points

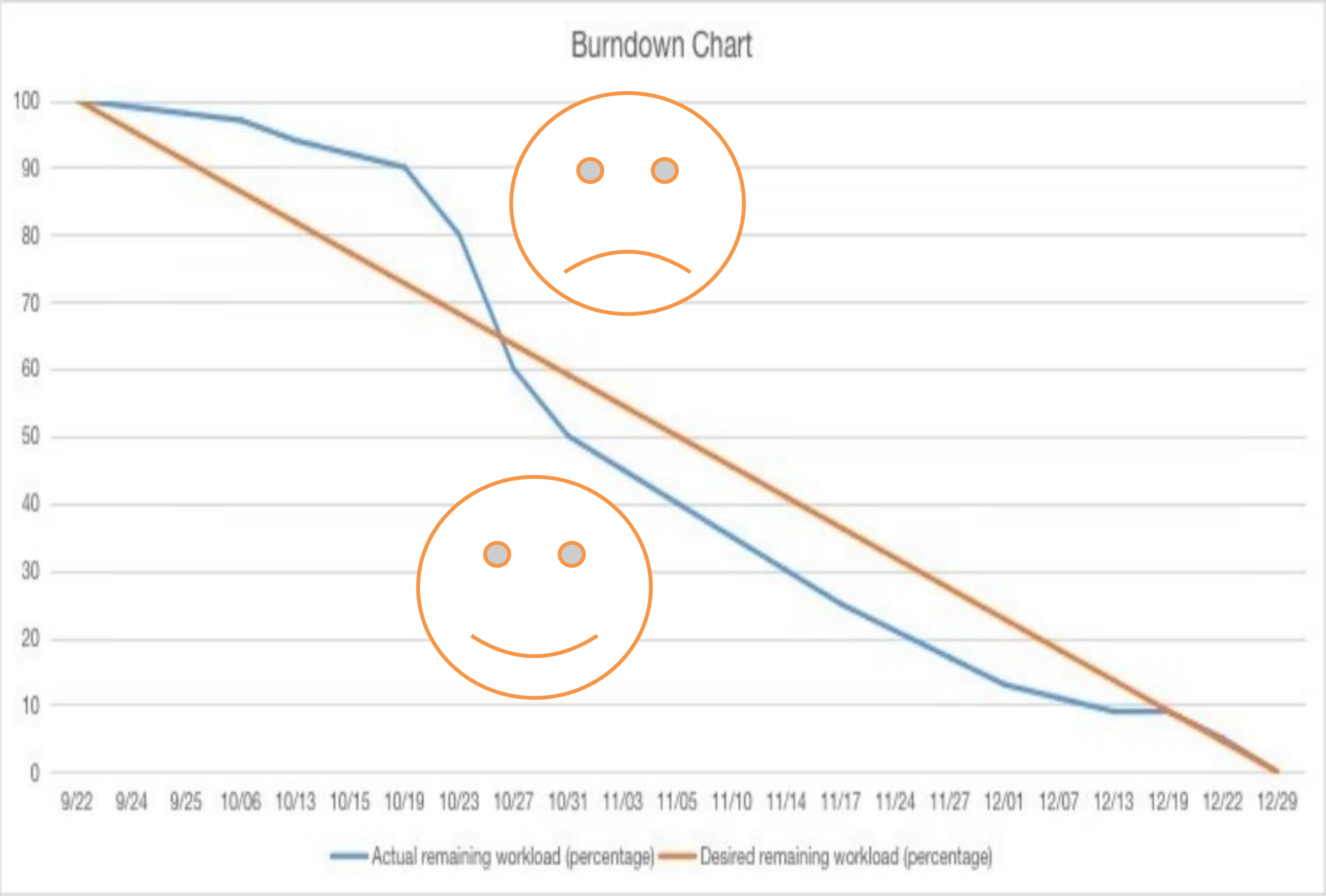


Monitoring Progress using a Burndown Chart

the starting point

The first use case was 'done' o n day 5;
We've burned 3 Sprint points

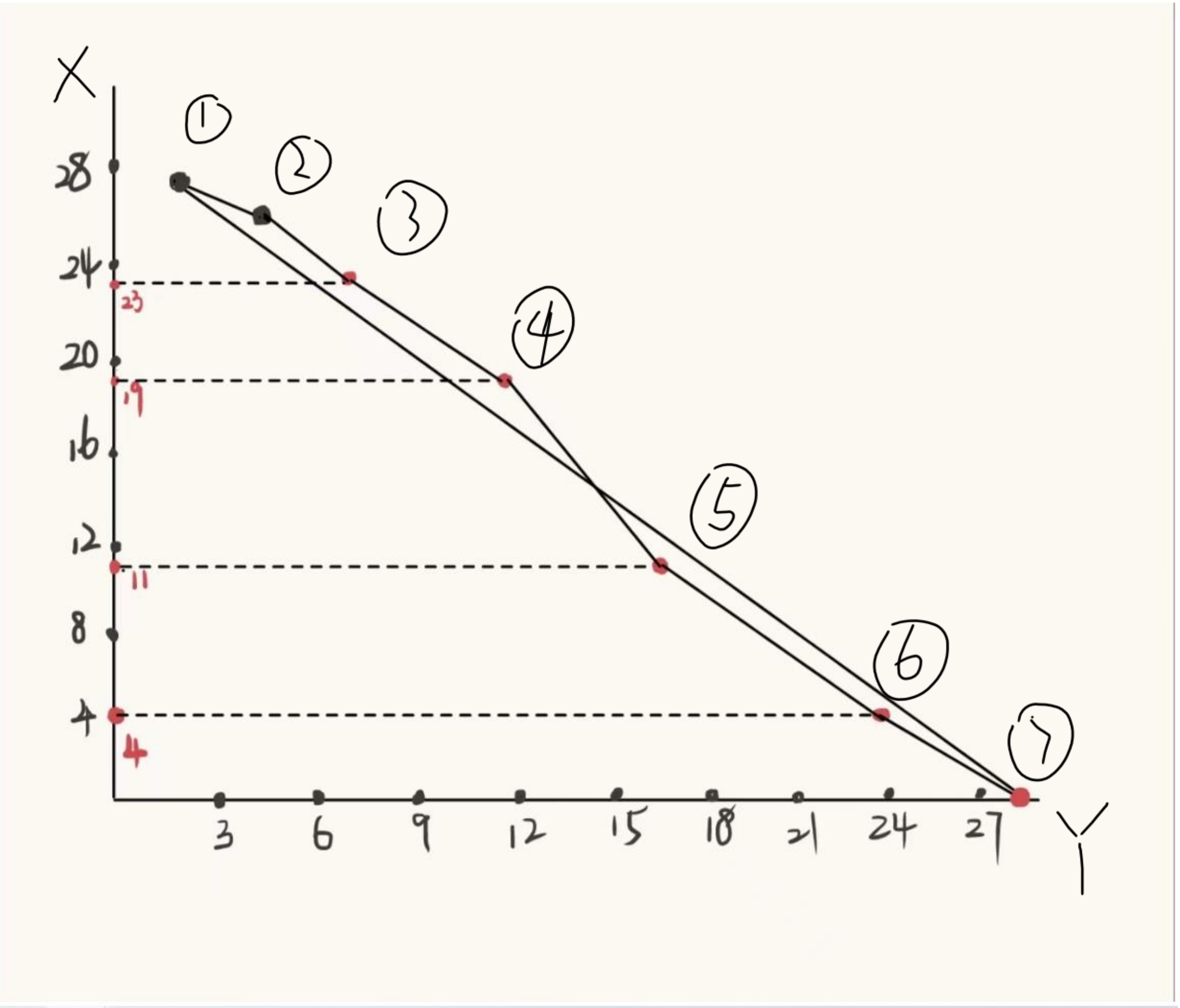
- The first iteration
- 4 Weeks
- 17 Days remaining
- 6 User Stories
- 28 Sprint points


Burndown Chart
100
90
80
70
60
50
40
30
20
10
0
9/22
9/24
9/25
10/06
10/13
10/15
10/19
10/23
10/27
10/31
11/03
11/05
11/10
11/14
11/17
11/24
11/27
12/01
12/07
12/13
12/19
12/22
12/29
Actual remaining workload (percentage)
Desired remaining workload (percentage)

Ideally, the actual effort would line up with the ideal effort. In reality, some work will be marked **below the ideal effort** line (indicating that the team **is ahead of schedule**), and other work will be marked **above the line** (indicating that the team is **behind schedule**).

Practice

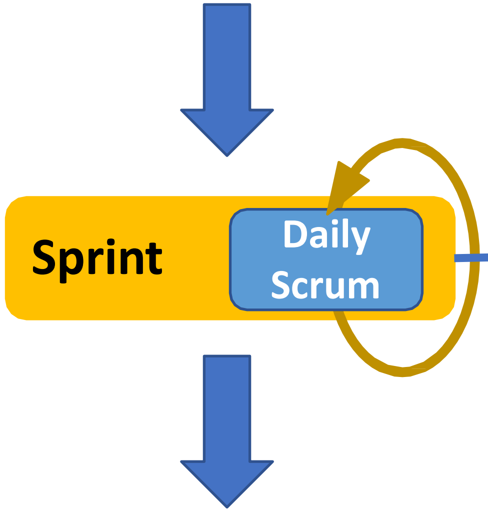
- In Scrum, a Burndown chart is frequently used for monitoring the development progress within a Sprint. According to the following Burndown chart, provide an appropriate explanation and analysis for the Sprint shown. (X: Sprint points, Y: days, the first point: the starting point)



Scrum Workflow – Sprint

Elements:

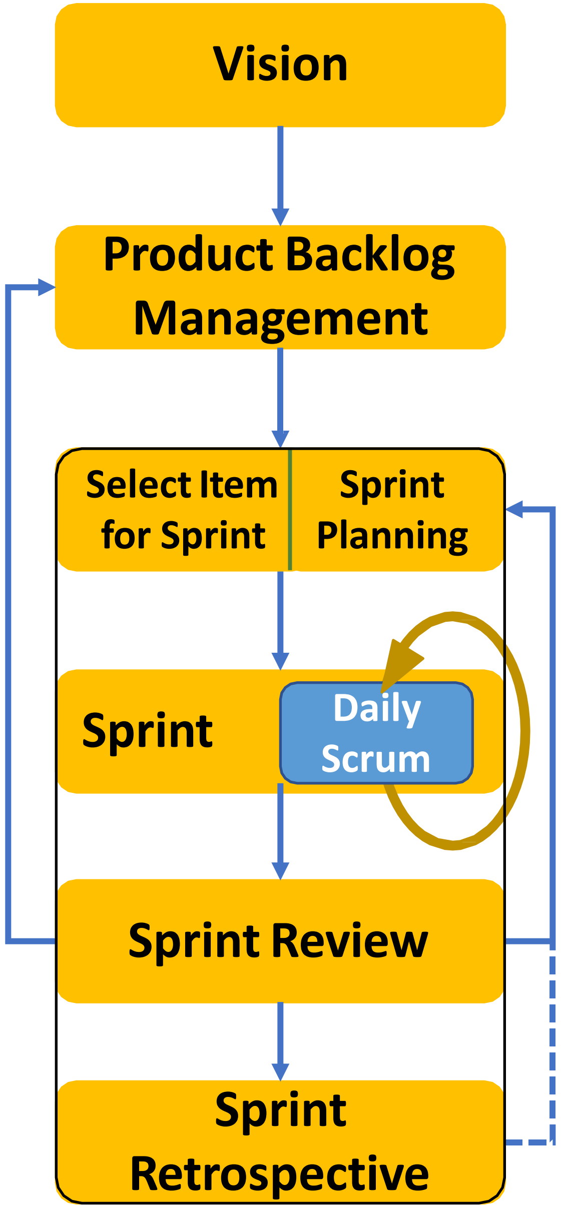
- *A Sprint contains the Sprint Planning, Daily Scrums, the development activity, Sprint Review, and the Sprint Retrospective.*



Rules:

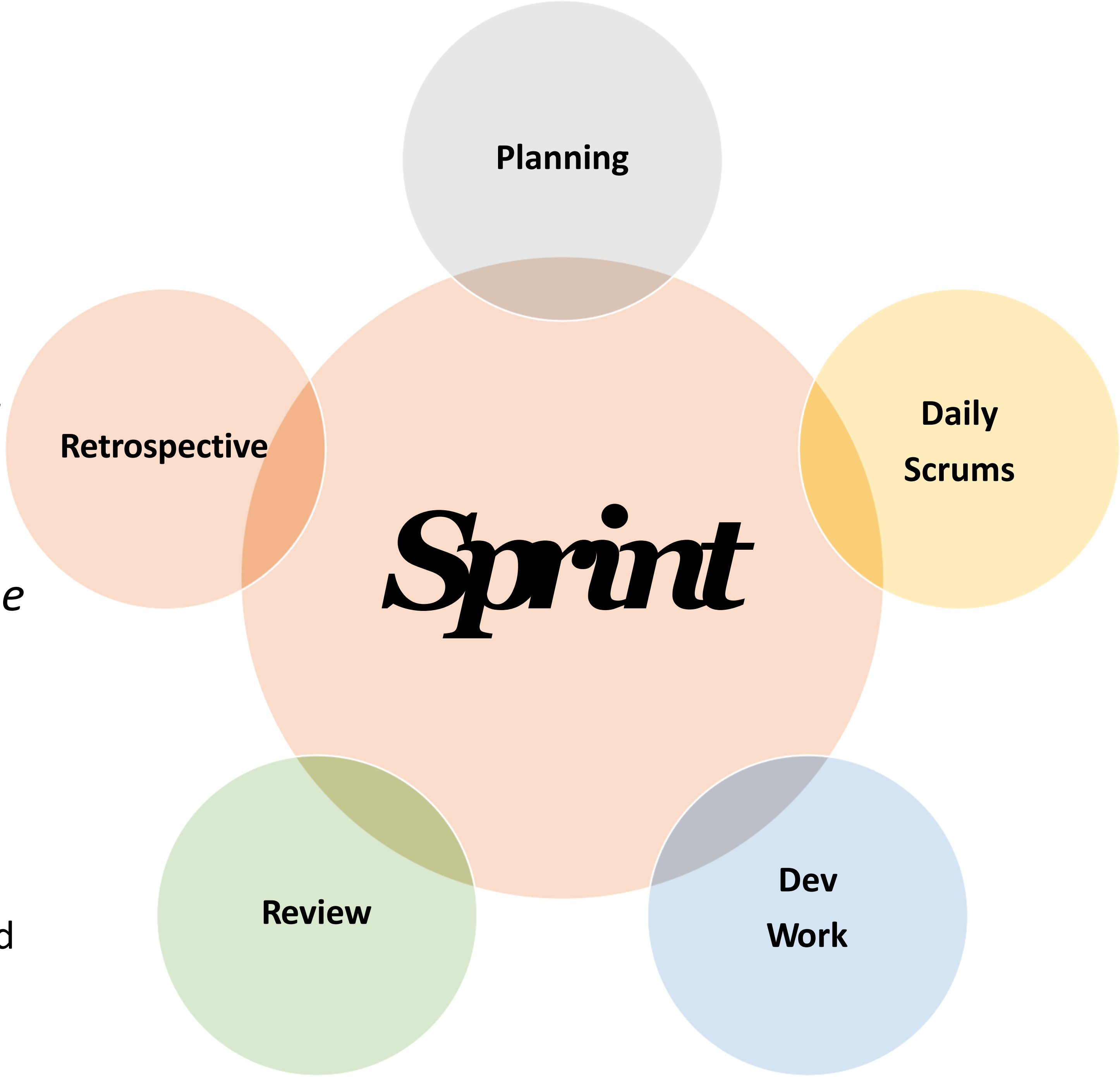
- *Work together as a self-organizing team*
- *Sprint goal can NOT be changed*
- *The quality of the goals can NOT be affected*
- *A Sprint can NOT be shortened*
- *A Sprint should NOT be longer than a month*
- *Only the Product Owner can cancel the Sprint.*

Outcomes: *Sprint Goal achieved; a possible shippable increment.*



Sprint

- *Sprints have consistent durations throughout a project.*
- *A new Sprint starts immediately after the retrospective of the previous Sprint.*
- *The short duration of Sprints limits the risk to small cost.*
- A Sprint may be cancelled when:
  - The Sprint goal becomes obsolete
  - In unexpected situations, e.g., an important team member has left
  - A much higher priority task comes in and requested by senior management



Scrum Workflow – Daily Scrums

> ***"The Daily Scrum is a 15-minute time-boxed event for the Development Team."***
>
> *-- Schwaber, K. and Sutherland, J., 2011. The scrum guide. Scrum Alliance, 21, p.19.*

Who:

- Development Team members need to attend the meeting at the same time and the same place
- The **Scrum Master** organizes the meeting

Daily Scrum

Discussion:

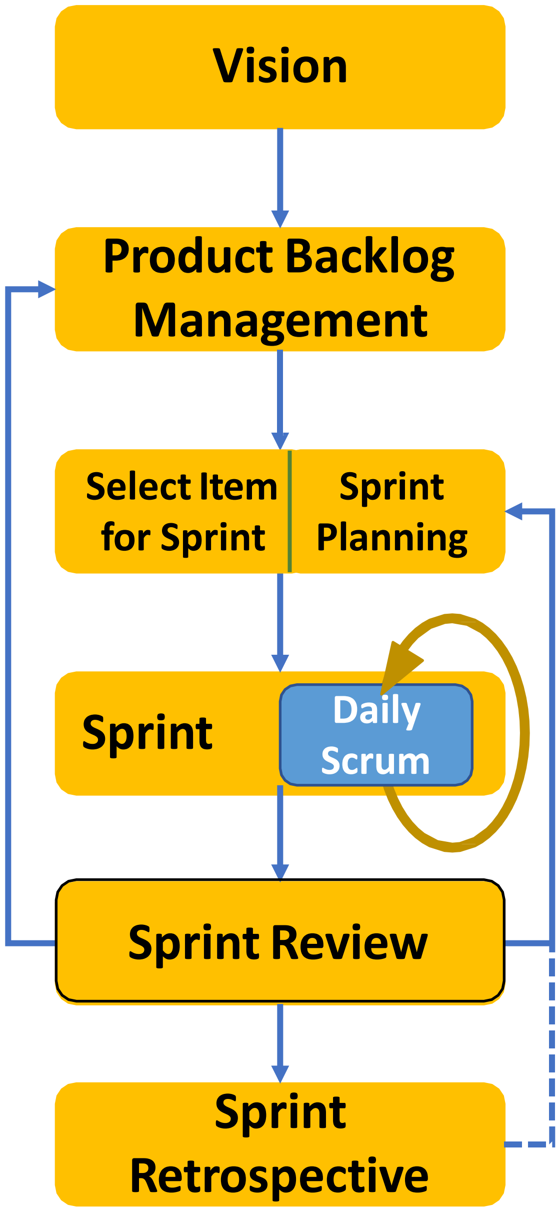
- *What did I do yesterday that helped the team meet the goal?*
- *What will I do today?*
- *Do I see any obstacles?*

Purpose:

- *To inspect progress toward the goal*
- *Planning for the next day*

Outcomes: *an informal plan for the next day.*

How long: *15 minutes max*



Vision

Product Backlog Management

Select Item for Sprint

Sprint Planning

Sprint

Daily Scrum

Sprint Review

Sprint Retrospective

Scrum Workflow – Sprint Review

Who:

- *The Scrum team and the relevant stakeholders review what was done.*
- *The **Scrum Master** organizes the meeting*
- *The **Product Owner** explains what has been done and what has not been done*

Sprint Review

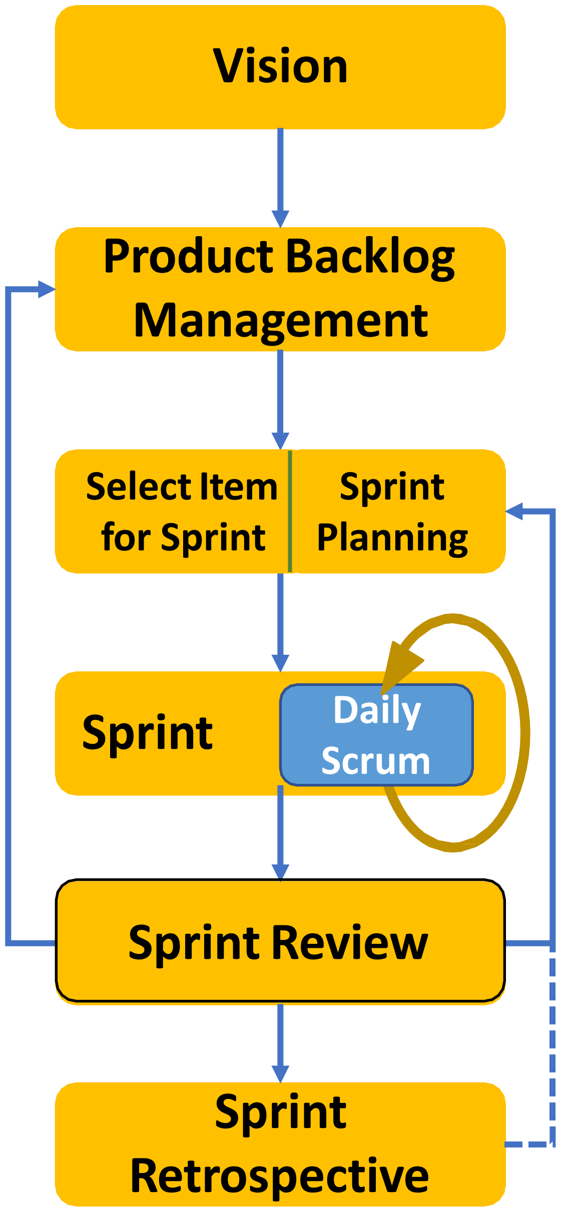
Elements:

- *Review of the timeline, budget, and potential capabilities*
- *Review what is the most valuable thing to do next*
- *Discuss the status of the Product Backlog*
- *Demonstrate the completed work and answer questions*

Outcomes: *a revised Product Backlog.*

When: *at the end of the Sprint*

How long: *Maximum 4 hours for a one-month Sprint*



Vision

Product Backlog Management

Select Item for Sprint

Sprint Planning

Sprint

Daily Scrum

Sprint Review

Sprint Retrospective

Scrum Workflow – Sprint Retrospective

Who:

- The Scrum Master organizes the meeting
- *The Scrum Master encourages the team to improve*
- *Scrum Team members plans the way to increase product quality*

Sprint Retrospective

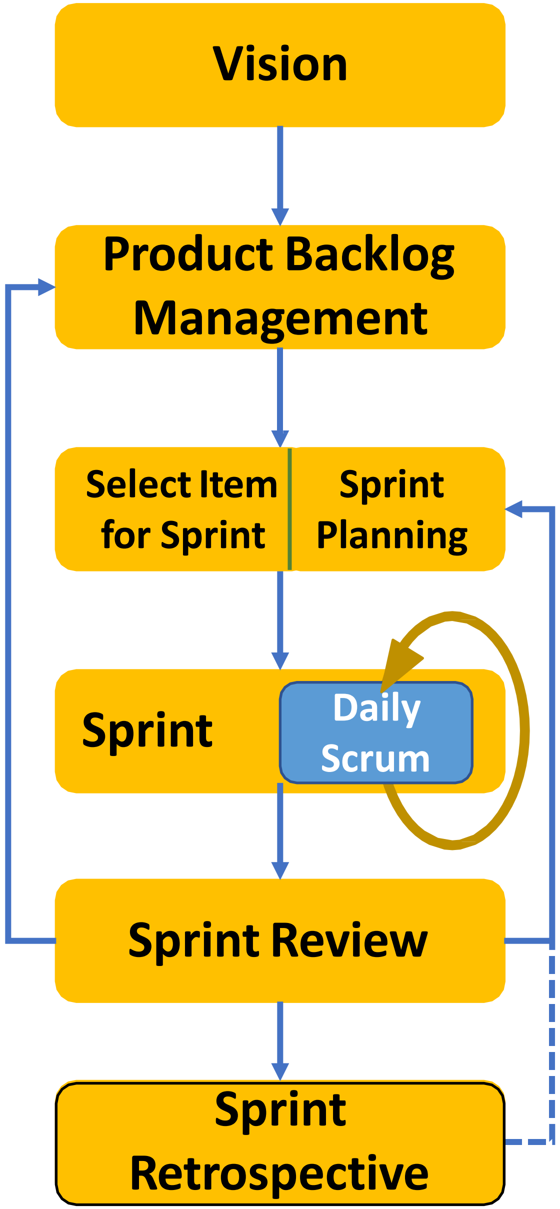
Purpose:

- *Inspect how the last Sprint went with regards to people, relationships, process and tools*
- *Identify potential improvements*
- *Create a plan for implementing improvements*

Outcomes: *plan for Improvements*

When: *after the Sprint Review*

How long: *Maximum 3 hours for a one-month Sprint*



Scrum Summary

Pillars:

1. Transparency
2. Inspection
3. Adaptation

Values:

1. Commitment
2. Courage
3. Focus
4. Openness
5. Respect

Artifacts:

1. Product Backlog
2. Sprint Backlog

Roles:

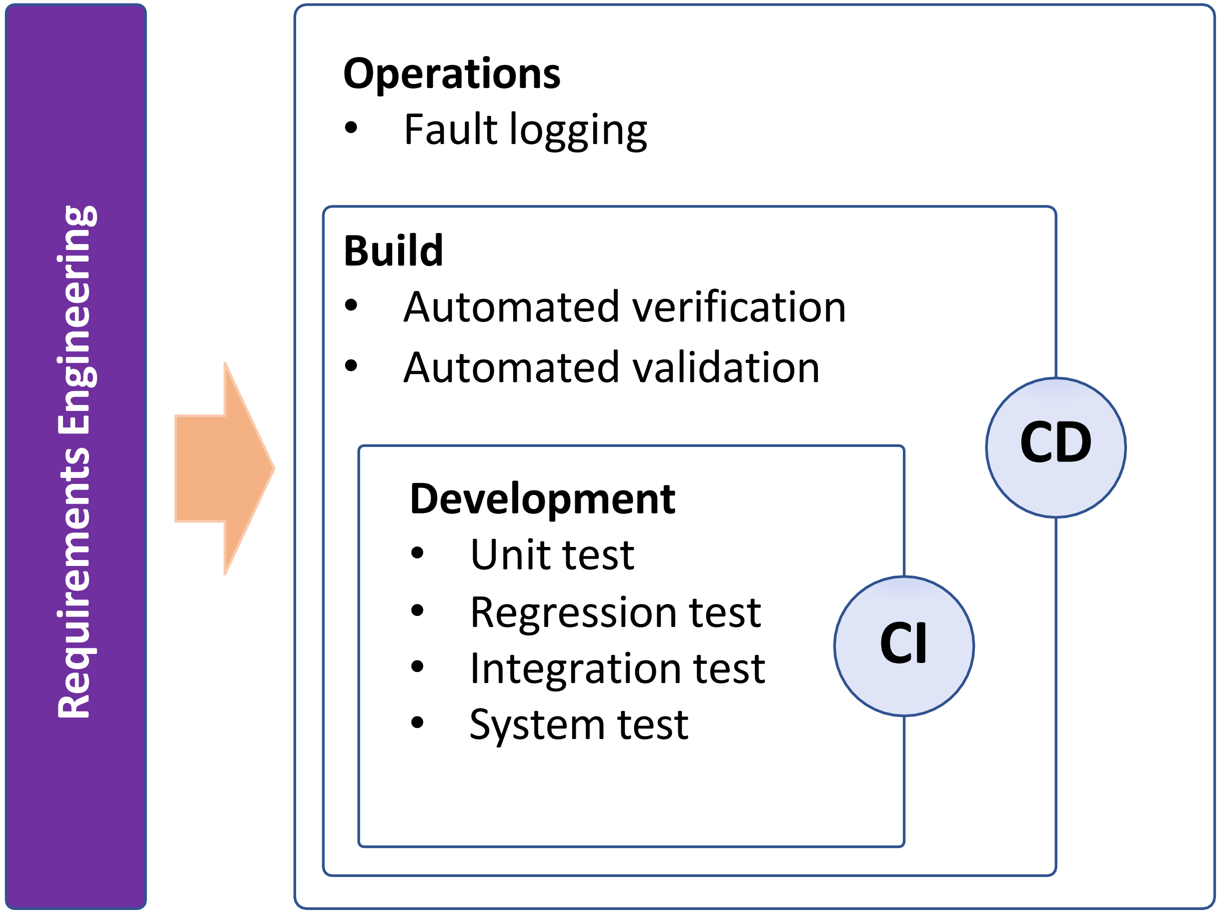
1. Product Owner
2. Scrum Master
3. Development Team

Events:

1. Sprint Planning
2. Sprint
3. Daily Scrum
4. Sprint Review
5. Sprint Retrospective

- The Scrum Master:
  - Helping the team to understand the project
  - Facilitating events
  - Coaching the Development Team for Scrum
  - Removing impediments
  - Planning Scrum implementations
  - Helping the Product Owner

DevOps



Testing and DevOps
CI/CD: Continuous Integration/Continuous Delivery

- DevOps refers to a combination of development and operations.
- The main drivers for DevOps to software development are:
  - to have fewer requirements changes,
  - to create a strong focus on testing and quality assurance,
  - to achieve a much faster delivery cycle.
- DevOps relies on automation tools where possible.

Practical Issues with Agile Methods

- Agile methods may not be suitable for embedded systems engineering or the development of large and complex systems
  1. The informality of agile is incompatible with the legal approach that is commonly used in large enterprises (Contractual Issues)

  2. Agile are commonly used for new software systems development, rather than for software maintenance

  3. Agile methods are designed for small, co-located teams. For large project with multiple geographically distributed teams, the management and coordination complexity increase significantly, thus the effectiveness of the Agile methods becomes questionable