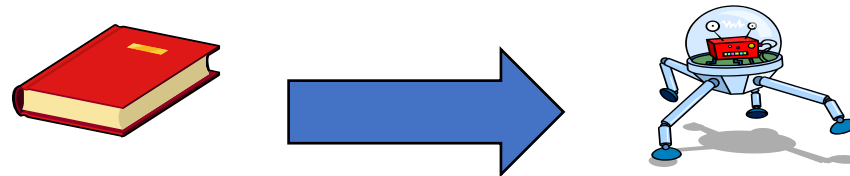# CS10102302
用户故事

赵君峤 副教授

计算机科学与技术系 电子与信息工程学院

同济大学

Requirements Management

- At its simplest, software development is about creating product that meets its users' needs
    - In Scrum terms, turning backlog into increments of functionality
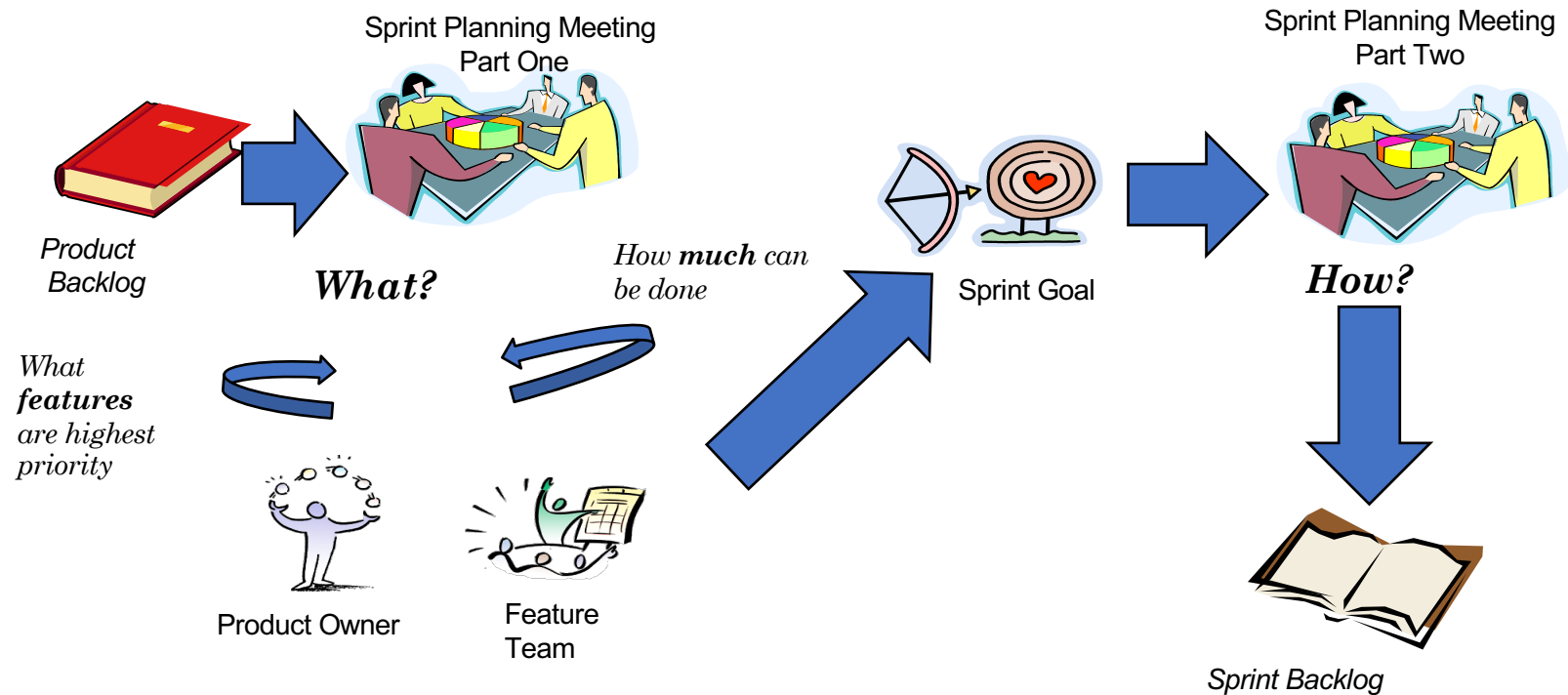   - Assuming needs are expressed as requirements

- Requirements issues rank top in most surveys of software project failures
    - For example, the CHAOS* report
        1. Incomplete requirements (13.1 %)
        2. Lack of user involvement (12.4 %)
        4. Unrealistic expectations (9.9 %)
        6. Changing requirements and specifications (8.7 %)
        8. Didn't need the project any longer (7.5 %)

*The CHAOS Report. The Standish Group, 1995-2010.

Agile Requirements Management

- Requirements Management is a broad discipline that includes
    - Requirements elicitation
    - Requirements analysis
    - Requirements specification
    - Requirements testing
    - Requirements tracing
- Traditional approaches rely on adding levels of management control
    - Produces reams of requirements documentation
    - Most of the work is done up front
        - And often needs to be completed before development work can proceed
- But for complex systems, it is impossible to define all requirements up front
    - Not least because they are liable to change
- Agile approaches tend towards doing "just enough" up-front requirements
    - Where "enough" means "enough to start development"
    - And then continuously evolving them

Recap: The Sprint Planning Meeting

Sprint Planning Meeting
Part One

Product Backlog

What?

*How **much** can be done*

Sprint Goal

Sprint Planning Meeting
Part Two

How?

*What **features** are highest priority*

Product Owner

Feature Team

Sprint Backlog
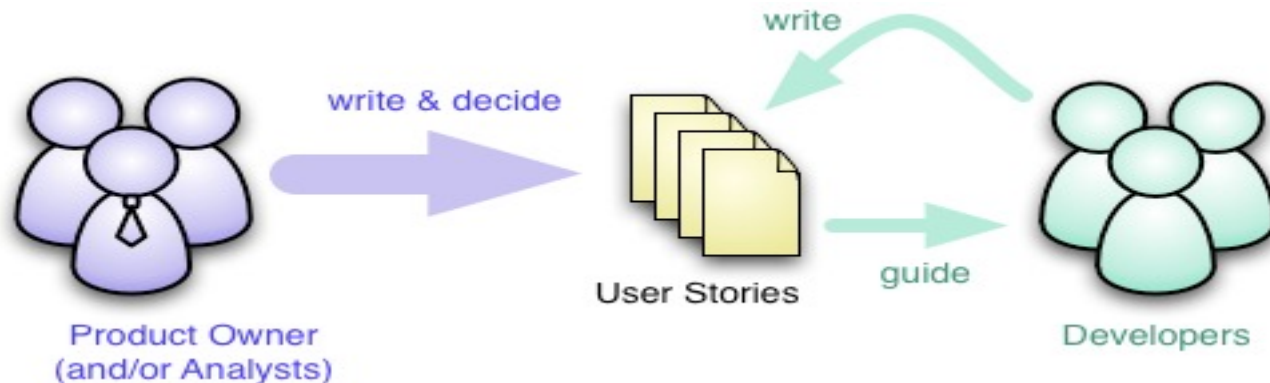
Requirements in Scrum

Product Backlog is the requirements in Scrum.

- ✓ Features/Acceptance Cases
- ✓ Estimation
- ✓ Priority

The most common way of writing a product backlog is through user stories.

RE in Scrum

PO's(Product Owner) responsibility to maintain the Product Backlogs.

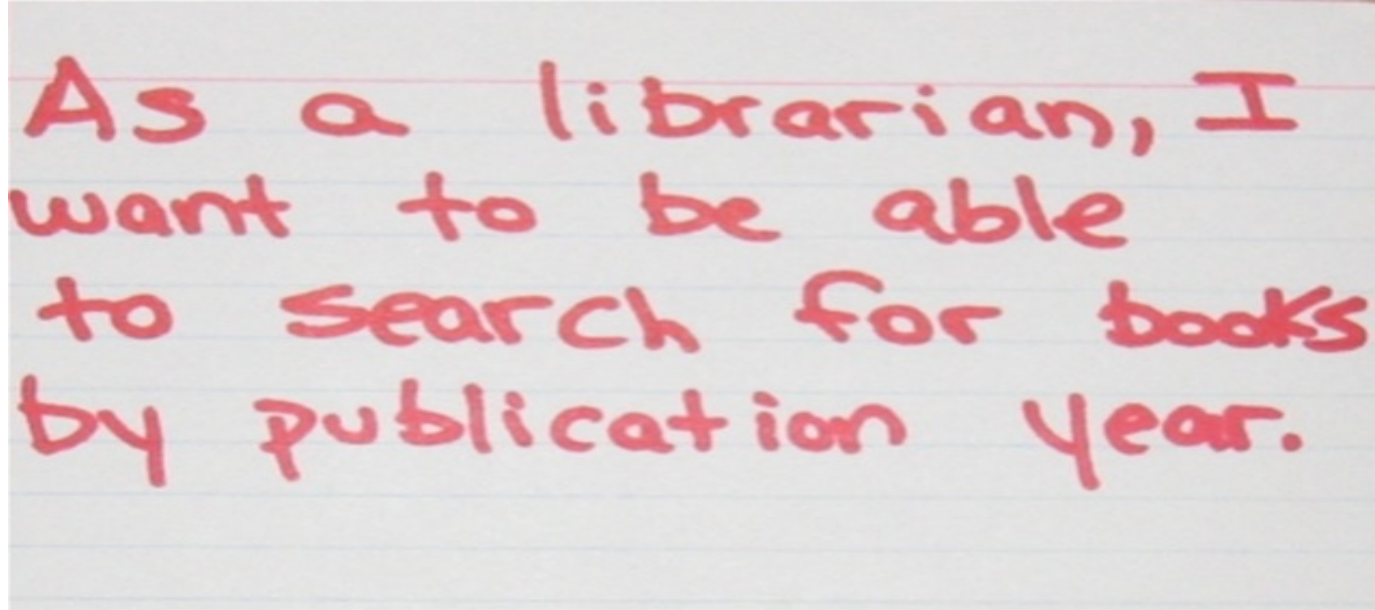

"No change" during Sprint.

Scrum team break backlogs into tasks.

User Stories

- A common Agile technique is user **S**tories

    - Derived from Extreme Programming (XP)

        - A popular Agile software development method

- A **U**ser story is a simple description of some requirement written from a user's viewpoint

    - Usually on an index card, typically 6" x 4"

    - Not a completely defined requirement

        - XP says, "There are no (system) requirements to start with"

    - A "reference point for a conversation"

        - Detailed requirements will evolve through interaction between the customer and the developers

- The aim is to shift toward having face-to-face conversations about requirements and technical issues

    - Rather than manage documentation about them

User Story Typical Format

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

User Stories: An Example

User Story #7	*View Shopping Cart Contents*

As a registered customer, I want to be able to view all items currently in my Shopping Cart together with their prices, so that I can see how much money I am spending.

Priority: Estimate:

- There is no standard format for writing user **S**tories

- The above example is typical:
    - `As a <role> I want <feature> so that <goal>`

- Often, the back of the card describes the acceptance test for the requirement

Activity: Refining the User Story

- Together with your group, quickly brainstorm a list of questions that would need to be answered to turn the user story into a "potentially shippable increment of functionality"

- If you have time, identify at what point a Feature Team would have sufficient information to *begin* work

    - The point here is that wireframes, prototypes and 'real' development can form part of the conversation(s) involving User Stories

The Definition of "Done"

- How does the Feature Team know when it has met a requirement?

- Scrum requires that every increment be "potentially shippable"
  - The Product Owner may decide to deploy it immediately
    - Even if the Sprint is not at the end of a scheduled release
  - This means gaining agreement about what it means to be "done"

- Different people have different perspectives
  - A programmer might believe "done" is when the code is complete
  - A tester is "done" when unit and integration testing is done
  - A customer might regard "done" as
    - Installation, deployment of the system
    - Together with user documentation
    - And (perhaps) user training

- In Scrum, "done" is an agreement between the Product Owner and the Feature Team
  - Minimally, the acceptance test for a story
  - At this level, it will vary from user story to user story

3C's

- A well written user story will describe what the desired functionality is, who it is for, and why it is useful.

- There are 3 parts to a fully fleshed out user story. If you like marketing-speak, then you can call them the "3 C's":

  - The Card
  - The Conversation
  - The Confirmation

The Card

- A typical user story follows this template:
  "As a [user], I want [function], so that [value]"
  - As a [role], I want [feature] because [reason]
  - As a [role], I can [feature]
  - As a [role], I can [feature] so that [reason]
- written on a plain card, usually to give a physical constraint which limits the possible length of the story.
- Here's a few hypothetical examples written for YouTube. I've defined a "Creator" as someone who contributes videos to the site, and "User" as someone who just watches them:
  - "As a Creator, I want to upload a video so that any users can view it."
  - "As a User, I want to search by keyword to find videos that are relevant to me."

Conversation

- Think of the conversation as an open dialogue between everyone working on the project, and the client. Anyone can raise questions, ask for things to be clarified, and the answers can be recorded down as bullet points for later reference. It is also a stage where you can reevaluate your user story, and possibly split it into multiple stories if required.

- For example, we discuss the creator upload user story, and decide that there are actually multiple things that can happen, so we split them, and expand on them.

- As a Creator, I want to upload a video from my local machine so that any users can view it.
  - The "Upload" button will be a persistent item on every page of the site.
  - Videos must not be larger than 100MB, or more than 10 minutes long.
  - File formats can include .flv, .mov, .mp4, .avi, and .mpg.
  - Upload progress will be shown in real time.

- As a Creator, I want to edit the video's metadata while a video is uploading, to save myself time.
  - Editable fields include Video Name, description, tags, and privacy settings.
  - Once saved, the user will be taken to their video's dedicated page.

- Once you write a few of these and get into the swing of it, you'll find you're getting a lot of data written down very quickly. Because of this, it's important to organise your user stories in a manageable way. I like to group them by interface, and present them next to a wireframe showing the intended functionality.
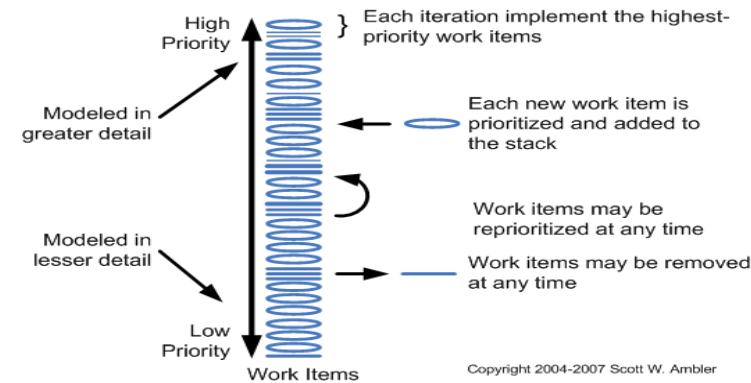
Confirmation

- The confirmation is basically just a test case. If you're not familiar with test plans and test cases, think of a test case as a series of steps that a user must do to achieve the User Story. A test plan is a collection of test cases.

- With full coverage of your system in your test plans, you can easily test every core piece of functionality and tick them off as you go through them.

- A test case for our YouTube example might look like this:

  As a Creator, I want to upload a video from my local machine so that any users can view it.
   1. Click the "Upload" button.
   2. Specify a video file to upload.
       1. Check that .flv, .mov, .mp4, .avi, and .mpg extensions are supported.
       2. Check that other filetypes aren't able to be uploaded.
       3. Check that files larger than 100MB results in an error.
       4. Check that movies longer than 10 mins result in an error.
   3. Click "Upload Video".
   4. Check that progress is displayed in real time.

Important considerations for writing user stories:

- **Stakeholders write user stories.**
- **Use the simplest tool**.
- **Remember non-functional requirements**.
- **Indicate the estimated size**.
- **Indicate the priority**.
- **Optionally include a unique identifier**.

High Priority — Modeled in greater detail
Low Priority — Modeled in lesser detail
Work Items

} Each iteration implement the highest-priority work items

Each new work item is prioritized and added to the stack

Work items may be reprioritized at any time

Work items may be removed at any time

Copyright 2004-2007 Scott W. Ambler

Front of Card

17B

As a student I want to purchase a parking pass so that I can drive to school

Priority: ~~XXXX~~ Should
Estimate: 4

Copyright 2005-2009 Scott W. Ambler

Back of Card

Confirmations:
The student must pay the correct amount
One pass for one month is issued at a time
The student will not receive a pass if the payment isn't sufficient
The person buying the pass must be a currently enrolled student.
The student may only buy one pass per month.

The Five Most Common Mistakes

- **1. User Story for a User**

- Example: "*As a user I want to be able to manage ads, so that I can remove expired and erroneous ads*."

- At the first glance you don't see any issues with that User Story, because all the required elements are present.

- Now tell me who is the user for whom you are going to build that feature of ads management and what does he understand ads management?

- Is that a portal administrator, who wants to have possibly of database clean-up and ads moderation?

- Or maybe it is an advertiser, who wants to have list of all ads he submitted to the side and have possibility of removing ads when they are not needed any longer or there was error found in the content?

- As you may noticed, I mentioned only two different roles with different expectations from the system.

- In this case what is missing is persona or role mentioned.

The Five Most Common Mistakes

- **2. User Story for Product Owner**
Example: "*As a Product Owner I want the system to have possibility of deleting ads, so that users have possibility of deleting ads*."

- And again, all three elements are here, but there is something wrong.

- First of all this User Story is type of 'you want story, here you have one'. Obviously person writing this User Story did that only for sake of doing it.

- You can have a Product Owner as role of person using the system if you build software for teams using agile software development. That indicates issues with implementation of agile methodology in that organization.

- Second of all you have here exactly the same issue as in the previous example. There is no role or persona to give you clues about user expectations.

The Five Most Common Mistakes

- **3. User Story for Developer**

- Example: "*As a developer I want to replaced the folder widget, so that I have maintained folder widget*."

- This typical example of Story for a Technical Backlog or technical requirement representation. Sometimes User Stories like this one are also part of so called technical debt.

- Technical debt consists of necessary maintenance tasks like software updates, re-factoring, changing frameworks and so on. They are totally rightful to be implemented, but represented like in the example above they do not represent value for the customer and you will not get a buy-in from Product Owner.

- Also from the 'agile point of view', you need to deliver a business value at the end of every iteration and the team needs to be able to present it to the stakeholders on the review meeting.

- How to write such a story correctly? Rewrite it from a user point of view with persona on role.

- Example: 'As a commercial user I want the system to allow me run multiple searches at the same time, so that I can do my job faster.'

- Task to go with it can be: 'Re-factor search handling mechanism to allow multiple threads for single user.', 'Update java version to 64-bit one.'

- Acceptance criteria need to define some measurable and testable definition of improvement like: 'Single user can run 5 searches at the same time.' and 'Search returns results in less than 4 seconds.'

The Five Most Common Mistakes

- **4. No Business Value or Benefit for Customer**

- Example: "*As an commercial advertiser I want to have filtering option.*"

- We have the role, we have the need, but reason and business value are missing. Why does an commercial advertiser want to have filtering option? What does he want to achieve?

- 5. No Acceptance Criteria or Conditions of Satisfaction

- Here you can use as an example one of the examples above. The problem in such practice is often underestimated.

- Not having Acceptance Criteria sometimes referred as Conditions of Satisfaction can cause the whole chain of mistakes starting with wrong definition of development tasks or wrong estimation.

- Story can fail the tests or Test Cases will cover different criteria due to lack of understanding.

- Acceptance Criteria pay great role is confirmation of requirement understanding and decide about acceptance of iteration deliverables. Conditions of Satisfaction question the User Story enable conversation between the Product Owner and the team. Good way of gathering Acceptance Criteria is asking questions such as 'What if … ?', 'Where …?', 'When …?', 'How …?'. Use examples and simple drawings to remove assumptions. It can happen that Story needs to be refined and re-planed or quite often split to smaller stories.

How detailed should a user story be?

Too broad

- "A team member can view iteration status."

Too detailed

- "A team member can view a table of stories with rank, name, size, package, owner, and status."

- "A team member can click a red button to expand the table to include detail, which lists all the tasks, with rank, name, estimate, owner, status."

Just right

- "A team member can view the iteration's stories and their status, with main fields."

- "A team member can view the current burndown chart on the status page, and can click it for a larger view."

- "A team member can view or hide the tasks under the stories."

- "A team member can edit a task from the iteration status page."

Real Example and Problems-1

User stories are not all written by POs and they are not identical in format

As the SDE, I expect that I can publish reports for <$SERVICE_NAME> back to the portal so that a customer may review them.

Hide "help" icon in the header if the login customer user's organization only purchases <$SERVICE_NAME> from Oracle

Make Subject field editable

Solution: Let it be

Real Example and Problems-2

We can't get conversation and confirmation of user stories from POs timely and quickly

➢ Distributed team

➢ Multiple POs

➢ Not full-time

Solution: weekly call, emails and a scrum tool

Real Example and Problems-3

Some tasks are necessary, but they are not business needs

e.g. setup development environment, defects, code refactoring, dev/qa/prod environment support etc.

Solution: Add a "Common Task Pool" user story in each sprint for each team

Real Example and Problems-4

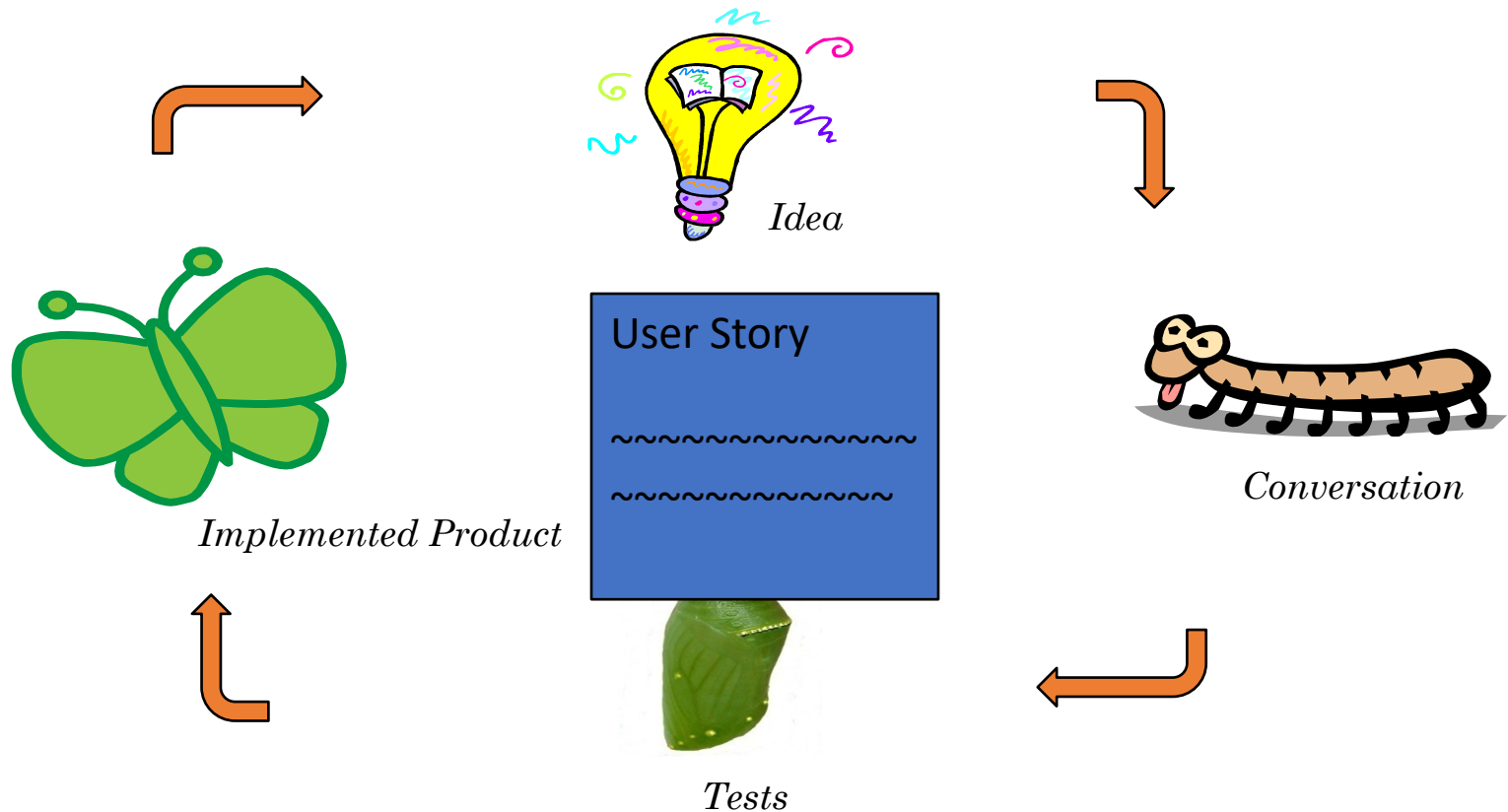*Inconsistent UI/Operation in a same system*

e.g. time format

YYYY-MM-DD, MM-DD-YYYY, DD-MM-YYYY

e.g. disable/deleted users

User A, ~~User B~~, User C

Solution: Common Team + Scrum of Scrum(SoS)

The Life Cycle of a User Story

Idea

User Story

~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~

Implemented Product

Conversation

Tests

尝试

- 现在急需开发一个"食堂食品溯源管理系统"
  - 如何开展需求抽取
  - 如何撰写用户故事?

Reference

- 清华大学国家级精品课程 《软件工程》主讲人 刘强 副教授 刘璘 副教授

- https://www.icourses.cn/sCourse/course_3016.html

- https://www.xuetangx.com/course/THU08091000367/5883555?channel=learn_title

谢谢大家！

THANKS