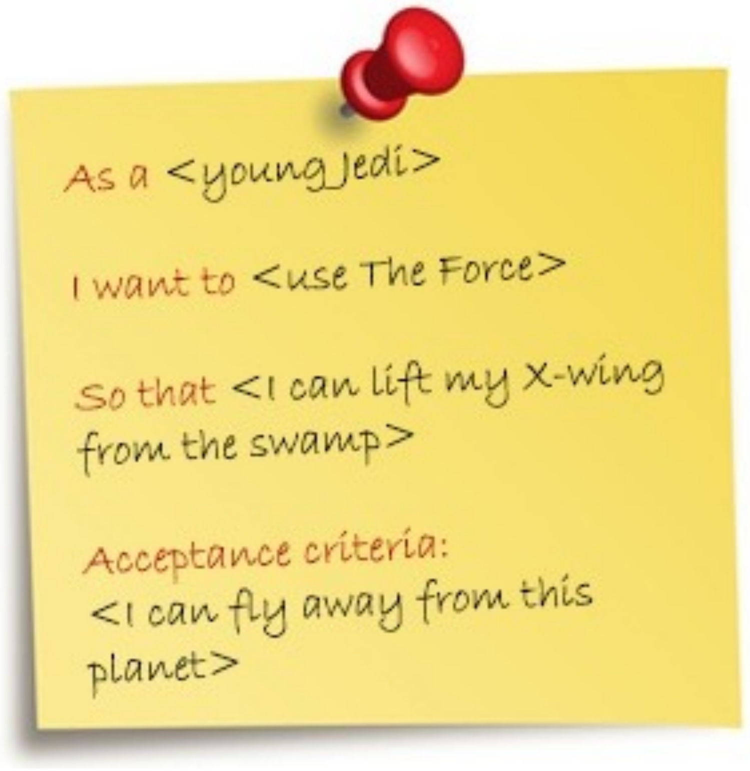
UNIVERSITAT DE BARCELONA

User Stories


As a <young Jedi>

I want to <use The Force>

So that <I can lift my X-wing from the swamp>

Acceptance criteria:
<I can fly away from this planet>

by Eloi Puertas

based on Oriol Pujol Software Engineering slides.

Product **backlog**

- A product backlog is a prioritized list of all the features and functionality needed to complete a project.
- Product backlogs include all work that the project will require and evolve over time. As such, they contain not only the new features for a product but also bug fixes and research—anything that will take the team’s time.
- The backlog answers the questions "what are we building and why are we building it?" This information is captured in work items.

Product **backlog**

- At the start of any project, it’s unlikely that you'll have a nice list of clean, well-defined product backlog items ready to be estimated and prioritized.

- Instead, you’re likely to have a stack of story note cards and maybe a functional specification or two.

How to write **good** stories

- The best piece of advice for writing a good user story is: write stories, not tasks.

- But what does that even mean? Aren’t user stories supposed to be tasks for implementing pieces of functionality?

- The purpose of the story is to start the conversation about what the business needs are and how it can be fulfilled with software.

Three c’s

CARD

Stories are traditionally written on note cards.

Cards may be annotated with estimates, notes,etc.

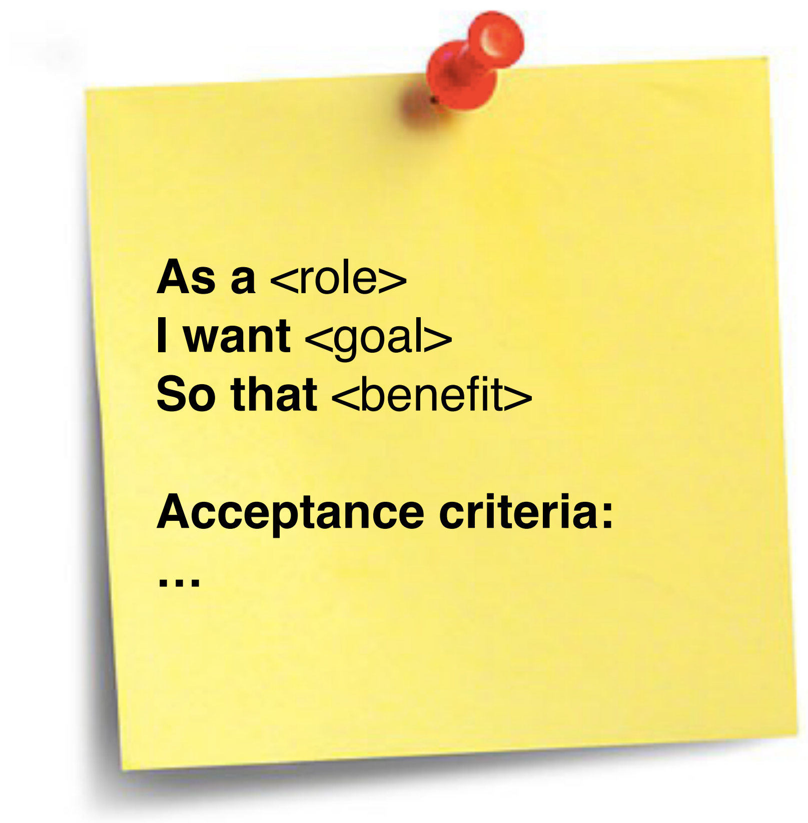
CONVERSATION

Details behind the story come out during conversations with the product owner.

CONFIRMATION

Acceptance tests confirm a story was coded correctly. Product owner expectations.

Card



As a <role>
I want <goal>
So that <benefit>

Acceptance criteria:
…

- Purpose:
  - To start a conversation between techies and business people from a common point of understanding.
  - User story is not supposed to be a final specification handed over for implementation

INVEST

Your user stories should be:

- **I**ndependent
- **N**egotiable
- **V**aluable
- **E**stimable
- **S**mall
- **T**estable

INVEST

Your user stories should be:

- **I**ndependent

  Stories should be as independent as possible. When thinking of independence it is often easier to think of "order independent."

  In other words, stories can be worked on in any order. It allows for true prioritization of each and every story. When dependencies come into play it may not be possible to implement a valuable story without implementing other much less valuable stories.

INVEST

Your user stories should be:

- **N**egotiable

  A story is not a contract. A story IS an invitation to a conversation. The story captures the essence of what is desired. The actual result needs to be the result of collaborative negotiation between the customer, developer and tester (at a minimum). The goal is to meet customer needs, not develop something to the letter of the user story if doing so is insufficient!

  Remember, you can always ask the magic question "How will I know I've done that?" to help drive the conversation.

INVEST

Your user stories should be:

- **V**aluable

  If a story does not have discernible value it should not be done.

  Hopefully user stories are being prioritized in the backlog according to business value, so this should be obvious.

  The story has value to the “user” in the user story.

  Finally, remember the “so that <value>” clause of the user story. It is there for a reason – it is the exact value we are trying to deliver by completing the story!

INVEST

Your user stories should be:

- **E**stimable

  A story has to be able to be estimated or sized so it can be properly prioritized. A story with high value but extremely lengthy development time may not be the highest priority item because of the length of time to develop it.

  What happens if a story can't be estimated? You can split the story and perhaps gain more clarity. Sometimes splitting a story doesn't help though. If this situation occurs it may be necessary to do some research about the story first.

INVEST

Your user stories should be:

- **S**mall

Obviously stories are small chunks of work, but how small should they be? The answer depends on the team and the methodology being used.

Suggestion: two week iterations which allow for user stories to average 3-4 days of work – TOTAL! This includes all work to get the story to a “done” state. By the way, you should do the simplest thing that works - then stop!

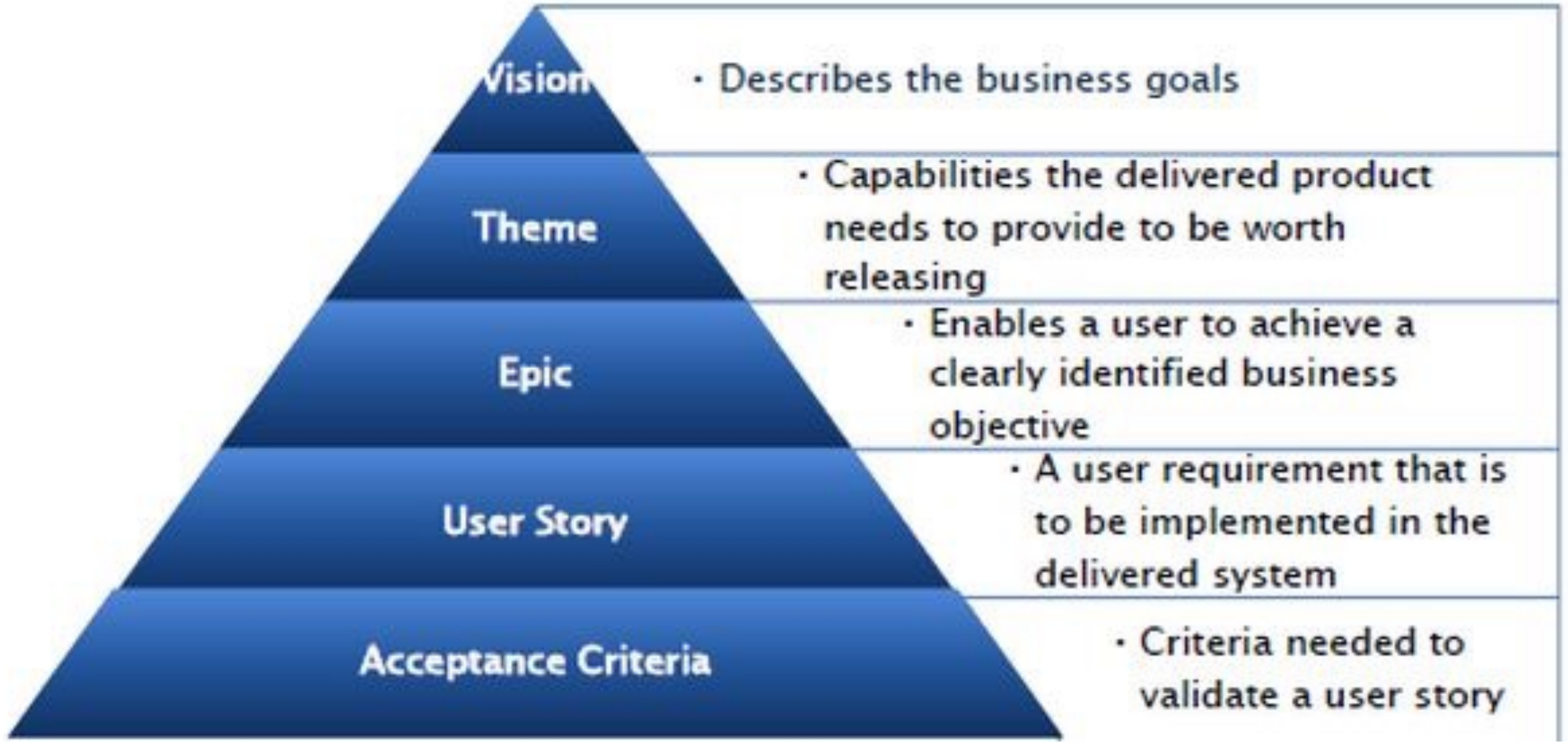
INVEST

Your user stories should be:

- **T**estable

Every story needs to be testable in order to be "done." In fact, it is useful to think of testable meaning acceptance criteria can be written immediately. Thinking this way encourages more collaboration up front, builds quality in by moving QA up in the process, and allows for easy transformation to an acceptance test-driven development (TDD) process.

Types of user stories


Vision
Theme
Epic
User Story
Acceptance Criteria
• Describes the business goals
• Capabilities the delivered product needs to provide to be worth releasing
• Enables a user to achieve a clearly identified business objective
• A user requirement that is to be implemented in the delivered system
• Criteria needed to validate a user story

Not everything is a **story**

Certain business needs cannot be even expressed as a story

- A change request
- A constraint: required performance, technology stack, database to use
- A technical requirement: communication protocol, security standard compliance

Don’t try to squeeze such requirements into the user story format, just write them down in plain text

User stories for backend systems are usually a hard topic on any project.

Bad user stories

- Auto-save every few minutes
  - As a player, I want auto-save the current game after key events, so that one does not have to repeat the whole stage.
- Add tips and reminders of user interface functionality.
  - As a novice player, I want UI tips so that I can get familiar with controls and icons
- **Fix bug #127**
- **Decrease loading time**

TIPS for writing good stories

- To identify stories, start by considering the goals of each user role in using the system.
- When splitting a story, try to come up with stories that cut through all layers of the application.
- Write smaller stories for functionality that will soon be implemented and broad stories for functionality further out.
- Keep the user interface out of the stories for as long as possible
- Write stories for a single user.
- Keep stories short
- Have the customer, rather than the developer, write the story

Conclusions

- Write stories, not tasks
- Follow INVEST principles
- Start the conversation as soon as possible

Starting Product Backlog

1. Choose one Data Science project (Theme)
2. Start with epics
3. Decompose Epics to User Stories
4. GOTO 1

Trello