Six attributes of good stories:

- **Independent**
- **Negotiable**
- **Valuable to users or customers**
- **Estimable**
- **Small**
- **Testable**

INVEST
1. Independent

Dependencies lead to **prioritization** and **planning** problems

**Prioritization problems**

Product Owner selects a high priority story that is dependent on a lower priority story

**Planning problems**

Stories for how companies can pay for job postings

*Story 1: pay with Visa card*
*Story 2: pay with Master card*
*Story 3: pay with American Express card*

*Estimate 3 days for first type and 1 day each for the remaining 2*

*Which type goes first and is assigned the 3 day?*
Dependency solution

• Combine the 3 credit card payment stories into 1 five day story
• If the combined story is too large, rethink the way to split the stories…for example
  Story 1: pay with one type of credit card
  Story 2: pay with two additional types of credit cards

OR

With just 1 story, add the following to the story card
“One estimate if the story is done before the other story, a lower estimate if done afterwards”
2. Negotiable

Stories are negotiable… short descriptions of functionality

Details should be negotiated in conversations between the Product Owner and the Development Team

Story cards are “reminders”… you do not need to include all the details

But, if details are known, include them as notes on the story card

Example:

A company can pay for a job posting with a credit card

Note: Accept Visa, MasterCard and American Express (consider Discover)
Not too much detail…

Notes on the “story card” help developer and Product Owner to resume conversation where they left off

**Example of Too much detail**

A company can pay for a job posting with a credit card.

**Note.** Accept Visa, MasterCard and American Express. Consider Discover. On purchases over $100 ask for card ID number form back of card. The system can tell what typed of card it is from the first two digits of the card number. The system can store a card number for future use. Collect the expiration month and date of the card.

- The problem…
  Details can create the illusion that all the details needed have been specified… and no need to further discuss the story with the user
Story **Card** is a reminder

**Contains:**

- A phrase or two that act as reminders to hold the conversations
- Notes about issues to be resolved during the conversations

**Example**

*A company can pay for a job posting with a credit card.*

*Notes: Will we accept Discover cards?*

*Note for UI: Don’t have a field for card type (if can be derived from first two digits on the card)*
Turn too much detail into tests…

- The story and notes are reminders of open questions
- Details about the story that can be converted to tests that should be added to the “back” of the story card

Example

Test with Visa, MasterCard and American Express (pass)
Test with Diner’s Club (fail)
Test with good, bad and missing card ID numbers
Test with expired cards
Test with over $100 and under $100
3. Valuable to users or customers!

**Users (use the software) versus Purchasers (purchase the software)**

- Development team building software that will be used by a customer with 5,000 computers
- Story that is not of value to users…
  *All configuration information is read from a central location*
- Stories important to buyers but not actual users…
  *Throughout the development process, the development team will produce documentation suitable for an ISO 9001 audit*
  *The development team will produce the software in accordance with CMM Level 3*
- Stories valued only by developers
  *All connections to the database are through a connection pool*
  *All error handling and logging is done through a set of common classes*
The ideas may be important…
but the focus is not on the User

Keep the idea… but refocus on user

Up to fifty users should be able to use the application with a five-user database license
All errors are presented to the user and logged in a consistent manner

Suggested that…

it is best to have the customer or users write the stories!
4. Estimatable

Reasons for problems when estimating the size of a story:

1. Developers lack domain knowledge
2. Developers lack technical knowledge
3. The story is too big

1. Solution to getting a general understanding: have additional conversations with the Product Owner (or author) who wrote the story!
2. **Developers lack technical knowledge**

Assign one or more developers to a *Spike*

They undertake a brief experiment to learn about an area of the application

Learning just enough to be able to estimate the task

Note: *Timebox* the Spike

**Example**

Building a website for long-term medical care for chronic conditions

Story statement: “New users are given a diabetic screening”

Developers and customer got together… found out that the screening was to be a simple web form with a handful of questions
5. Small

- Stories that are too large or too small cannot be used in planning
  Travel Reservation Epic: “A user can plan a vacation”
  Split Epics into smaller stories
- Appropriate size
  “Based on the team, its capabilities and the technologies in use”
Splitting Stories

Epics are Compound Stories (made-up of multiple shorter stories)

Example: “A user can post her résumé ”

Meanings?

• Résumé can include education, prior jobs, salary history, publications presentations, community service, and objective
• Users can mark résumé as inactive
• Users can have multiple resumes
• Users can edit résumé
• Users can delete résumé

Each could become a unique story
Splitting Stories … continued

Stories that are too small

Example - Job seeker can:
• Enter a date for each community service entry
• Edit the date for each community service entry
• Enter a date range for each prior job on a résumé
• Edit the date range for each prior job on a Résumé

Better to split into smaller stories

Example - User can:
• create Résumé
• , which include education, prior jobs, salary history, publications, presentations, community service and an objective
• edit a Résumé
• delete a Résumé
• have multiple Résumés
• activate and inactivate Résumé

There are usually many ways to disaggregate a compound story
Splitting Stories … continued

Complex Stories due to uncertainty – not easily split into a set of stories

**Example**: “A company can pay for a job posting with a credit card”… in this case, developers have no experience in credit card processing

The **Split**:

- Investigate credit card processing over the web (a *Spike* task)
  
  *Always timebox the Spike*

- A user can pay with a credit card
Splitting Stories … continued

Complex stories that extend known algorithms where the *first* story deals with the research needed to determine feasibility and the *second* story to add that functionality to the product.

- Put investigation story in one iteration and the other stories in one or more subsequent iterations
- Only the investigation story can be estimated

The customer can prioritize the research separately from the new functionality, *and then*

Decide whether or not to add the investigative story that add no new functionality over some other story.
Combining Stories

Those small stories that can be implemented faster than writing the story

Common examples: Bug reports or user interface changes

Example: Project with five bugs or a request to change some colors on the search screen.
Estimate the total work for all five and treat as a single story
6. Testable

• Stories must be testable!
• Testing proves the story was successfully developed… and the coding is finished
• Untestable stories:
  – User must find the software easy to use
  – User must never have to wait long for any screen to appear
• Tests should be automated … whenever possible
  – Similar to tests needed to ensure that new additions do not break the “Build”
  – Regression tests
Tests that cannot be automated

Example:

“Novice user is able to complete common workflows without training”

Can be tested but not automated.

Human factors expert can design a test that involves observation of a random sample of representative novice users

Obviously, time consuming and expensive

Another example:

“A user never has to wait long for any screen to appear.”

Never?

An alternative that is testable: “New screens appear within two seconds in 95% of all cases”
Summary

- Ideally, stories are independent from one another. Not always possible, but if it is... write them so they can be developed in any order.

- The details of a story are negotiated between the Product Owner and the Development Team.

- Stories should be written so that their value to users (Product Owner as proxy) is clear. The best way is to have the user write the story.

- Stories may be annotated with details, but too much detail obscures the meaning of the story and can give the impression that no conversation is necessary between the developers and the customer.
Summary - continued

• One of the best ways to annotate a story is to write test cases for the story.

• When stories are too big, compound and complex stories may be split into multiple smaller stories.

• If they are too small, multiple tiny stories may be combined into one big story.

• Stories need to be testable!