Writing User Stories
Product owners ...

- ... always have *unlimited desires* but *limited resources*

- ... have requirements, which necessitate communication with those who can provide the solution to said requirement.
Negotiation over Contracts

“Since users [product owners] don’t know how to solve their problems, we need to stop *asking* … and to *involve* them instead” - Mike Cohn

Involving a product owner in the refinement of their requirements via User Stories saves:

- **Time**: would you rather write a novella of requirements or simply an outline?
- **Money**: Legal fees in contract review; contractual change orders
A Story template

“As a <User or role>
_ I want _ <Business Functionality>
So _that_ <Business Justification>”

Example:

As a Account Holder,
_ I want _ to be able to withdraw funds from my checking account,
So _that_ I can buy some bling.
Stories are not

- “mini” Use Cases
- a complete specification
- a contract
- intended to be interpreted without a Product Owner
What is an Epic?

- Are usually compound Stories, that can be broken down into several smaller, more focused stories.

- May encompass enough work for several Sprints (iterations).
User Stories guidelines

- **Testable.** Tangible acceptance tests can be written against any delivered software
- The scope of the User Story is manageable enough for the team to provide an *Estimate*
- **Independent** and do not rely on other Stories
- **Sized appropriately.** Have a level of effort which the team can comfortably achieve in the duration of a single iteration
Some places to consider breaking Epics

- At C.R.U.D boundaries
- At system boundaries where two systems interface
- At Happy-Path / Exception-Path boundaries
At CRUD boundaries

- This solution is commonly used in environments that interact with a database

- Example:
  
  As an account holder, I want to be open a checking account …

  As an account holder, I want to deposit a check into my checking account …

  As an account holder, I want to view the updated balance in my checking account …
At system boundaries

- This solution is commonly used in environments where there are a large number of legacy systems.

- And can be used:
  - When there is a clear separation between two systems.
  - Where the interface between the two systems is well understood.

- Beware of creating dependencies between two different projects.
At Happy-Path / Exception-Path boundaries:

- Commonly used when transitioning from Use Cases
- The happy-path scenario may still need to be decomposed
- Breaking down Use Cases can be a lot of work … it may be simpler to just start using User Stories
What are Acceptance Criteria?

- Product Owner *expectations* on what will be delivered

- Acceptance Criteria can include:
  - Functionality that the system will perform
  - Interface look and feel
  - Necessary documentation (e.g. SOX compliance documentation)
Guidelines: Acceptance Criteria

- Be explicit
  - “The system will display the date.” …
  - In what format? Is “2006, April 1st” acceptable?

- Provide examples for clarity
  - “The system date will be displayed in the format 1/4/06”

- List any assumptions that the team may not be fully aware of
Guidelines: Acceptance Criteria

- Include what you’d expect the system to do
  “The checking account balance will be updated with the amount entered by the user.”

- And where ambiguous, what the system is not expected to do
  “Reconciliation with the amount of funds deposited is not expected at this time.”
Questions?
Presented by:
Bryan Liff
VP, IT Services
bliff@minerva-group.com
References

- Adapted from Kane Mar’s “Writing Stories”
- “Users Stories Applied”, Mike Cohen
- “Agile Estimating and Planning”, Mike Cohen
- http://www.ScrumAlliance.org