



CODE71

Agile Requirements Management

Scrum *Pad*®

iterate, collaborate, deliver

Capturing requirements using “User Stories”

Syed Rayhan

Co-founder, Code71, Inc.

Contact: srayhan@code71.com

Blog: <http://blog.syedrayhan.com>

Company: <http://www.code71.com>

Product: <http://www.scrumpad.com>

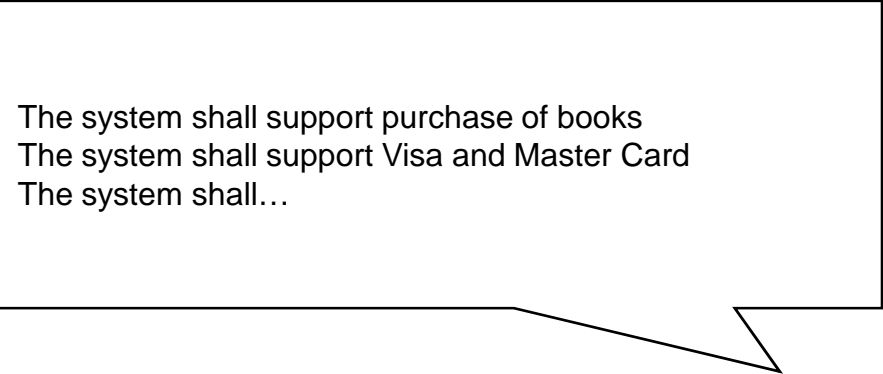
Confidential

11/17/2007

Agenda

- **Traditional approach to requirements gathering**
- **What is a User Story?**
- **Why User Stories?**
- **Key Elements of a good User Story (INVEST)**
- **Guidelines for writing good User Stories**
- **Organizing User Stories**

IEEE 830 Style of Software Requirement Specification (SRS)



The system shall support purchase of books
The system shall support Visa and Master Card
The system shall...

- Tedious to read
- Difficult to translate into a format usable by developers
- Assumes everything is knowable in advance
- Difficult to understand how end-user will use the system
- Difficult to estimate

Use Case style of SRS

Title: Checkout a product
Primary Actor: Purchaser

Main Path:

1. Purchaser provides credit card information
2. System validates CC
3. System charges CC the full amount for the product
4. System provides an unique confirmation number

- Not sized to support iterative incremental development
- Learning curve involved
- Inflexible because requires too much details
- Difficult to estimate

The Goal of Traditional Requirements

To be precise and complete



Entrée comes with soup or salad and bread

Soup or (salad and bread)?

(Soup or salad) and bread?

Requirements as User Stories

- **Card**
 - A high level description of a user's intent
 - Traditionally written on a card
 - Used for conversation with customers
 - Used for planning
- **Conversation**
 - Takes place over time
 - In-person discussions with supporting documentations
- **Confirmation**
 - A way to verify developers met the customer's intent of the story
 - Customers uses a set of acceptance tests to express/capture how he/she wants to verify the work
 - Details are added as tests

An example of requirement elaboration using Story

1

Card

[A user can purchase a book with a valid credit card.](#)

2

Conversation

[A user can purchase a book with a valid credit card.](#)

What kind of user?

What information needed to purchase?

What cards are supported?

Any confirmation needed?

3

Confirmation

[A user can purchase a book with a valid credit card.](#)

What information needed to reserve? **Name, Check-in & Checkout Dates,**

What cards are supported? **Visa, Master**

Any confirmation needed? **Email**

- Try it with a valid Visa then a valid MasterCard.
- Enter card numbers that are missing a digit, have an extra digit, and have two transposed digits.
- Try it with a card with a valid number but that has been cancelled.
- Try it with a card expiration date in the past.

Story Template

Title


One line describing the story activity

Narrative

As a [role]
I can/want [action] → Generates events
so that [benefit]

Acceptance Criteria

Title scenario name
Given [some context/assumption] → Defines single event
when [something happens]
then [some outcome is expected]



A Story Example

Title Students can purchase a book online

Narrative As a/an [Student]
I want [book a room at a hotel using a credit card]
so that [I can make overnight business trip at short notice]

Acceptance criteria Title Traveler books a room with a valid credit card
Given
when selects a hotel
and selects a room
then [some outcome is expected]

Use Case vs User Story

Use Case

- Attempts to replace on-going conversation
- Attempts to be complete
- Does not support evolutionary style of requirements elaboration
- Difficult to breakdown into more manageable yet meaningful chunk to fit within an interaction

User Story

- Attempts to trigger on-going conversations
- Attempts to convey the intention
- Supports just-in-time requirements elaboration
- Supports just-in-time appropriate level of estimation

So, why user stories?

- Shift focus from writing to talking
- Stories are comprehensible
- Stories are the right size for planning
- Stories support and encourage iterative development
- Stories can be progressively broken down into right size just-in-time for development
- Stories support opportunistic development
- Stories support *“participatory design”*

What Makes a Good Story?

INVEST

- Independent
- Negotiable
- Valuable
- Estimatable
- Small
- Testable

Negotiable

- **Stories are not**
 - Written contracts
 - Requirements that the software must fulfill
- **Do not need to include all details**
- **Too many details give the impressions of**
 - false precision or completeness
 - that there's no need to talk further
- **Need some flexibility so that we can adjust how much of the story gets implemented**
- **If the card is a contract then it needs to be estimated like a contract**

Which one is Negotiable?

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 from back of card. The system can tell what type 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.

Is this negotiable ?

Is this negotiable ?

OR

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

Note: Will we accept Discover cards?
Note for UI: Don't have a field for card type (it can be derived from first two digits on the card).

Valuable

- Stories must be valuable to either
 - Users
 - Customers paying for the software
- Should make sense to developers

Is the value clear?

All connections to the database are through a connection pool.

Is the value clear?

All error handling and logging is done through a set of common classes..

Estimatable

A story may not be estimatable if:

- Developers lack domain knowledge
- Developers lack technical knowledge
- The story is too big, a.k.a. **Epic**

Guidelines for writing good stories

- Start with goals (testable)
- Slice the cake (independent)
- Write closed stories (valuable, estimatable)
- Put constraints on cards (testable)
- Size the story to the horizon (negotiable)
- Include user roles in the stories
- Write for a user (small)
- Keep the UI out as long as possible (negotiable)
- Some things aren't stories

Start with goals

- Identify real users and group them with common attributes
- For each group of users, ask
 - What are their goals in using the system?
 - What they use the software for?

Write for one user

- It helps avoid confusion

Is it clear?

“Job Seekers can post resumes.”

- Sometimes it does not

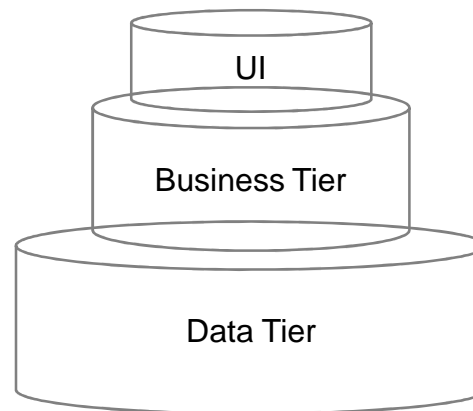
“Recruiters can search for good candidates.”

User roles

- Broaden the scope from looking at one user
- Allows users to vary by
 - What they use the software for?
 - How they use the software?
 - Background
 - Familiarity with the software / computers
 - Used extensively in usage-centered design
 - Definition
- Identify attributes that distinguish one user role from another

Slice the cake

- Our first inclination is often to write stories that are purely for one layer
- We're better off taking a slice through the entire cake



Write closed stories

- A closed story is one that finishes with the achievement of a meaningful goal
- User feels she's accomplished something

"A user can manage the ads she's placed."

Is it a closed story?

Include user roles in the stories

- Start thinking of the software as solving the needs of real people
- Sometimes all users want to act in a specific story but often it's a type of user
- Help everyone by putting that user in mind when looking at the story card:
 - A **Job Seeker** can post a resume.
 - A **Recruiter** can read submitted resumes.
- Avoid saying “the user” and instead say
 - “A Frequent Flier...”
 - “A Repeat Traveler...”
 - “Jim...”

Size the story to the horizon

- Sizes of stories
 - Epic
 - Compound
 - Complex
- Just-in-time requirements elaboration
 - Start with epics
 - Breakdown the epics that will be implemented in the near time horizon
 - Stop when the story can be fitted in an iteration

Keep the UI out as long as possible

- On a new project the UI doesn't exist, so leave it out of stories as long as possible
- Including UI detail in a story constrains the possible solutions
- Eventually, you'll have UI-specific stories
 - “Add a page size button to the print dialog.”
- Avoid too much UI detail

How to “split” a user story

Split along operational boundaries (CRUD)

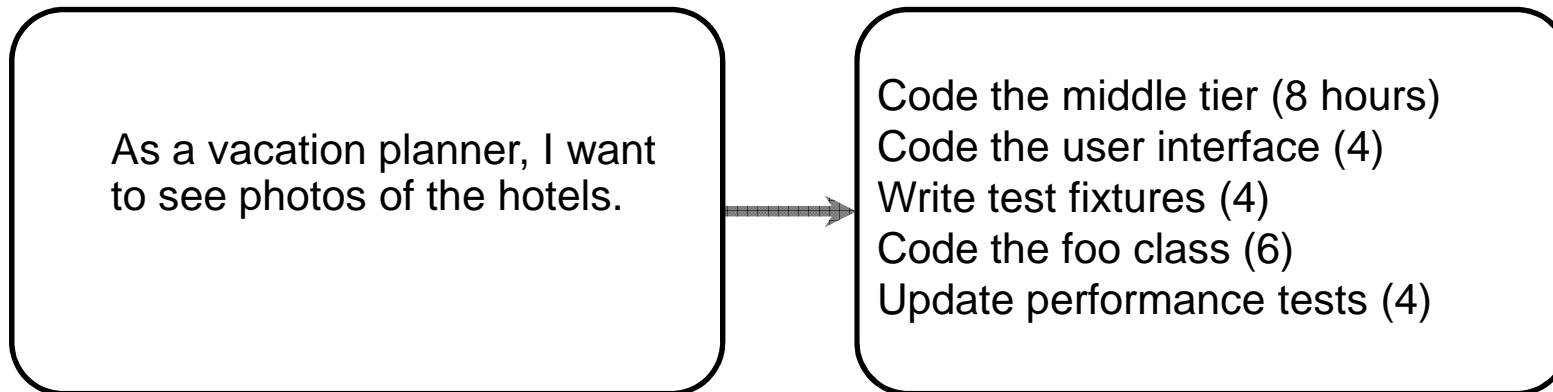
- A user can create resumes, which include education, prior jobs, salary history, publications, presentations, community service, and an objective.
- A user can edit a resume.
- A user can delete a resume.
- A user can have multiple resumes.
- A user can activate and inactivate resumes.

A user can create resumes, which include education, prior jobs, salary history, publications, presentations, community service, and an objective.

Split along data boundaries

- A user can add and edit educational information on a resume.
- A user can add and edit prior jobs on a resume.
- A user can add and edit salary history on a resume.
- A user can delete a resume.
- A user can have multiple resumes.
- A user can activate and inactivate resumes.

Break a story into tasks during sprint planning



Put constraints (a.k.a non-functional req.) on cards

- Write constraints on cards, just like any other stories
- Annotate with “constraint.”
- Put each into the earliest possible iteration
- Have tests to verify the constraint is met

What are some examples of constraints?

Q&A

Contact: srayhan@code71.com

Blog: <http://blog.syedrayhan.com>

Company: <http://www.code71.com>

Product: <http://www.scrumpad.com>