Requirements as User Stories and UI Design

CS 370 SE Practicum, Cengiz Günay

(Some slides courtesy of Eugene Agichtein and the Internets)

Did your spring break?
Due after Spring Break:

- Project interface mock-up and object-data model designs (see class #8 notes from 2/18)
- Details in Project tab on class website
- Deadline moved to Thursday, 3/20 :)

	
Due after Spring Break:

- Project interface mock-up and object-data model designs (see class #8 notes from 2/18)
- Details in Project tab on class website
- Deadline moved to Thursday, 3/20 :)

Today:

- User stories, use cases, state diagrams, scenario-based design
- Before that: WebIdol winner!
Dim the lights, here we go!
Requirements Analysis

- Requirements analysis
  - specifies software’s operational characteristics
  - indicates software's interface with other system elements
  - establishes constraints that software must meet

- Requirements analysis allows the software engineer (called an analyst or modeler in this role) to:
  - elaborate on basic requirements established during earlier requirement engineering tasks
  - build models that depict user scenarios, functional activities, problem classes and their relationships, system and class behavior, and the flow of data as it is transformed.
Domain Analysis

- Define the domain to be investigated.
- Collect a representative sample of applications in the domain.
- Analyze each application in the sample.
- Develop an analysis model for the objects.
Use-Cases

- a scenario that describes a “thread of usage” for a system
- *actors* represent roles people or devices play as the system functions
- *users* can play a number of different roles for a given scenario
Use-Case Diagram

homeowner

SafeHome

Configure SafeHome system parameters

Set alarm

Access camera surveillance via the Internet

cameras

safeHome

From a “use case” to “user stories”

- Use case diagram shows all things an actor can do
- Can be broken down to individual user stories
User Stories

Templates & Examples
User Stories are the central focus for developers

Each User Story should imply an acceptance test

Complexity is estimated in Story Points
  - Arbitrary measure of relative complexity
  - We use modified Fibonacci Sequence (0, 1, 2, 3, 5, 8, 13, 21)
  - Estimates are collaborative to uncover assumptions
  - Based on Staffing we estimate how many Story Points we can accomplish in a 2 week Iteration (Velocity)
User Story Template

User Story #XXX

**AS A [USER ROLE] I WANT TO [GOAL] SO I CAN [REASON]**

(optional)

*Screen should be grayscale only*

*Graph shows xxxxx with yyyyy*

*Screen drawing of the application with explanations where needed*

<table>
<thead>
<tr>
<th>Priority: 1</th>
<th>Importance of this user story in the overall development and/or order in which task must be accomplished to meet the goals in other User Stories.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimate: 6</td>
<td>Number of &quot;developer points&quot; to code and deliver this user story.</td>
</tr>
</tbody>
</table>

**Confirmation:**

**Success:**
1. List the specific events (screens, etc.) that occur if the user story succeeds.
2. Usually there is more than one "success" metric

**Failure:**
1. Screens or behavior that occur when the application fails
2. Not all "failures" are due to the application: Handle user-input errors too!
Example from Template

<table>
<thead>
<tr>
<th>User Story #001</th>
<th>Students can check their assignment grades online.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority: 1</td>
<td></td>
</tr>
<tr>
<td>Estimate: 6</td>
<td>(1 pts) create login page with username, password, course#</td>
</tr>
<tr>
<td></td>
<td>(1 pts) validate login</td>
</tr>
<tr>
<td></td>
<td>(2 pts) track failed attempts in a text file</td>
</tr>
<tr>
<td></td>
<td>(2 pts) display matrix of assignment and grade for the student</td>
</tr>
</tbody>
</table>

**Confirmation:**

**Success:**
1. User enters valid username, password, course# which produces a matrix of assignments and grades for his scores only.
2. Empty assignment matrix produces a page that says “no scores on file yet.”

**Failure:**
1. Incorrect name or password returns an error page stating “Incorrect name and/or password.” The page includes a button or link to return to the login page
2. Failed login attempts are maintained in a log stored at /website/logs/failures.txt
3. Server failure generates a message “Service unavailable; please try again later.”
Other Example User Stories*

- **Create Preferred Customer**
  As a Customer, when I purchase more than $50,000 in goods since my first purchase, I become a Preferred Customer so that I can receive the benefits associated with that status.

- **Calculate Employee’s Income Tax Deductions**
  As an Employee, obtain my income tax rate from the Pay System on [the mainframe], so that I can have an accurate amount for my Total Pay Deductions.

- **Search for Book by Author**
  As a Bookstore Customer, I can search for books by the Author’s Name, so that I can easily find all books by that Author.

- **Set Preferred Customer Discount**
  As a Preferred Customer, I receive a 10% discount on all prices, so that I’m rewarded for my prior purchases.

- **Search for Book by Title**
  As a Bookstore Customer, I can search for books by the Title, so that I can easily find all books with that Title.

- **Search for Book by ISBN**
  As a Store Associate, I can search for a book by its ISBN, so that I can determine the book’s in-stock quantity for the store.

These were excerpted from [http://www.westborosystems.com/2010/02/user-story-examples/](http://www.westborosystems.com/2010/02/user-story-examples/)
Also see [http://agile.csc.ncsu.edu/SEMaterials/tutorials/coffee_maker/](http://agile.csc.ncsu.edu/SEMaterials/tutorials/coffee_maker/) for a great set of examples and tutorials
Key Points

Cards:
physical (or virtual) medium where the story is recorded

Conversation:
Discussion and clarification, modification of the story via communication with the user

Confirmation:
Tests to verify the user story
I = independent. Not requiring other features.
N = negotiable. Can be excluded, revised, etc.
V = valuable. Clearly contributing to product usefulness
E = estimable. Reasonable development estimates can be made from the story.
S = small. Stories that are too big tend to be vague and to miss the other points, too.
T = testable. Stories need to have a means of verifying functionality.
So how to maintain user stories?

- Use online tool: Pivotal Tracker
  - To create, assign, and maintain user stories for your project
- **Mandatory**: We will track your progress through it.
  - your points for first iteration (due 3/20) depend on it!
- I own an academic account, we will send invites to teams.
So how to maintain user stories?

- Use online tool: Pivotal Tracker
  - To create, assign, and maintain user stories for your project
- **Mandatory**: We will track your progress through it.
  - your points for first iteration (due 3/20) depend on it!
- I own an academic account, we will send invites to teams.
User Interface Design
Interface Design

Easy to learn?
Easy to use?
Easy to understand?
Interface Design

**Typical Design Errors**

- lack of consistency
- too much memorization
- no guidance / help
- no context sensitivity
- poor response
- Arcane/unfriendly
Golden Rules

- Place the user in control
- Reduce the user’s memory load
- Make the interface consistent
Place the User in Control

- Define interaction modes in a way that does not force a user into unnecessary or undesired actions.
- Provide for flexible interaction.
- Allow user interaction to be interruptible and undoable.
- Streamline interaction as skill levels advance and allow the interaction to be customized.
- Hide technical internals from the casual user.
- Design for direct interaction with objects that appear on the screen.
## Find an Existing Value

<table>
<thead>
<tr>
<th>Request ID</th>
<th>Create Position Status</th>
<th>Department</th>
<th>Description</th>
<th>Position Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>P00000001</td>
<td>Created</td>
<td>027521</td>
<td>Test 1</td>
<td>00010720</td>
</tr>
<tr>
<td>P00000002</td>
<td>New</td>
<td>011004</td>
<td>Secretary</td>
<td>(blank)</td>
</tr>
<tr>
<td>P00000003</td>
<td>Created</td>
<td>027521</td>
<td>Test 2</td>
<td>00010725</td>
</tr>
<tr>
<td>P00000004</td>
<td>Created</td>
<td>053301</td>
<td>Director</td>
<td>00010723</td>
</tr>
<tr>
<td>P00000005</td>
<td>Created</td>
<td>027521</td>
<td>Test Director</td>
<td>00010721</td>
</tr>
<tr>
<td>P00000006</td>
<td>Created</td>
<td>027521</td>
<td>Test approval / position creation</td>
<td>00010724</td>
</tr>
<tr>
<td>P00000007</td>
<td>Created</td>
<td>027521</td>
<td>test</td>
<td>00010726</td>
</tr>
<tr>
<td>P00000008</td>
<td>Created</td>
<td>027521</td>
<td>Test grey-out logic</td>
<td>00010727</td>
</tr>
<tr>
<td>P00000009</td>
<td>Pending</td>
<td>031101</td>
<td>Other Job</td>
<td>(blank)</td>
</tr>
<tr>
<td>P00000010</td>
<td>Pending</td>
<td>027521</td>
<td>AI-Test Error Msgs</td>
<td>(blank)</td>
</tr>
<tr>
<td>P00000011</td>
<td>Created</td>
<td>025401</td>
<td>Assistant Director</td>
<td>00010728</td>
</tr>
<tr>
<td>P00000012</td>
<td>In Review</td>
<td>027521</td>
<td>test</td>
<td>(blank)</td>
</tr>
<tr>
<td>P00000013</td>
<td>In Review</td>
<td>032011</td>
<td>Assistant Director Level</td>
<td>(blank)</td>
</tr>
<tr>
<td>P00000014</td>
<td>In Review</td>
<td>031121</td>
<td>Administrative Assistant</td>
<td>(blank)</td>
</tr>
<tr>
<td>P00000015</td>
<td>In Review</td>
<td>053341</td>
<td>Assistant</td>
<td>(blank)</td>
</tr>
<tr>
<td>P00000016</td>
<td>Created</td>
<td>024141</td>
<td>Administrative Secretary</td>
<td>00010730</td>
</tr>
<tr>
<td>P00000017</td>
<td>Denied</td>
<td>011001</td>
<td>Business Manager</td>
<td>(blank)</td>
</tr>
<tr>
<td>P00000018</td>
<td>In Review</td>
<td>057201</td>
<td>Associate Professor</td>
<td>(blank)</td>
</tr>
<tr>
<td>P00000019</td>
<td>In Review</td>
<td>025101</td>
<td>Budget Manager</td>
<td>(blank)</td>
</tr>
</tbody>
</table>
Reduce the User’s Memory Load

- Reduce demand on short-term memory.
- Establish meaningful defaults.
- Define shortcuts that are intuitive.
- The visual layout of the interface should be based on a real world metaphor.
- Disclose information in a progressive fashion.
Make the Interface Consistent

- Allow the user to put the current task into a meaningful context.
- Maintain consistency across a family of applications.
- If past interactive models have created user expectations, do not make changes unless there is a compelling reason to do so.
Interface Design Steps

- Using information developed during interface analysis, define interface objects and actions (operations).
- Define events (user actions) that will cause the state of the user interface to change. Model this behavior.
- Depict each interface state as it will actually look to the end-user.
- Indicate how the user interprets the state of the system from information provided through the interface.
Activity Diagram

Supplements the use case by providing a graphical representation of the flow of interaction within a specific scenario.
Swimlane Diagrams

Allows the modeler to represent the flow of activities described by the use-case and at the same time indicate which actor (if there are multiple actors involved in a specific use-case) or analysis class has responsibility for the action described by an activity rectangle:

- **Actors**
- **Timing**
- **Flow**
Design Issues

- Response time
- Help facilities
- Error handling
- Menu and command labeling
- Application accessibility
- Internationalization
Usability Testing

- Define tasks
  - user stories
  - use cases

- Define users
  - Testers
  - Regular users

- Test and collect measurements
  - Time to complete task
  - Common errors
  - Complaints
  - Bugs
Facebook in 1997...
WebApp Interface Design

- **Where am I?** The interface should
  - provide an indication of the WebApp that has been accessed
  - inform the user of her location in the content hierarchy.

- **What can I do now?** The interface should always help the user understand his current options
  - what functions are available?
  - what links are live?
  - what content is relevant?

- **Where have I been, where am I going?** The interface must facilitate navigation.
  - Provide a “map” (implemented in a way that is easy to understand) of where the user has been and what paths may be taken to move elsewhere within the WebApp.
Effective WebApp Interfaces

- Bruce Tognozzi [TOG01] suggests...
  - Effective interfaces are visually apparent and forgiving, instilling in their users a sense of control. Users quickly see the breadth of their options, grasp how to achieve their goals, and do their work.
  - Effective interfaces do not concern the user with the inner workings of the system. Work is carefully and continuously saved, with full option for the user to undo any activity at any time.
  - Effective applications and services perform a maximum of work, while requiring a minimum of information from users.
Mapping User Objectives

List of user objectives

- objective #1
- objective #2
- objective #3
- objective #4
- objective #5
- objective #n

Navigating menu

Menu bar

major functions

graphic, logo, and company name

graphic

Home page text copy
Aesthetic Design

- Don’t be afraid of white space.
- Emphasize content.
- Organize layout elements from top-left to bottom right.
- Group navigation, content, and function geographically within the page.
- Don’t extend your real estate with the scrolling bar.
- Consider resolution and browser window size when designing layout.
Design Evaluation Cycle

1. Preliminary design
2. Build prototype #1 interface
3. User evaluates interface
4. Evaluation is studied by designer
5. Design modifications are made
6. Build prototype #n interface
7. Interface design is complete
12 Techniques for UI Design

From: “12 Useful Techniques For Good User Interface Design”  
by Dmitry Fadeyev

1. Highlight important changes  
2. Enable keyboard shortcuts in your Web application  
3. Upgrade options from the account page  
4. Advertise features of the application  
5. Use color-coded lists  
6. Offer personalization options  
7. Display help messages that attract the eye  
8. Design feedback messages carefully  
9. Use tabbed navigation  
10. Darken background under modal windows  
11. Lightboxes and Slideshows  
12. Short sign-up forms
Highly recommended reading from Joel Spolsky’s blog:

**Painless Functional Specifications**
- Part 1: Why Bother?
- Part 2: What’s a Spec?

* “Functional specifications” in Joel’s words is equivalent to our “functional requirements.”*
Happy Spring Break!