Use Case Summary
In-class Exercise

Problems

- How to convey typical usage scenarios to stakeholders in a way that all can understand
  - Customers, marketers, architects, developers, testers
  - Provide a lightweight means for exploring requirements
- How to express, quickly, key requirements for users in a standardized way
- How to provide a basis for system testing
- How to identify issues for prototyping
- How to start thinking about traceability from requirements to architecture

“Use Cases” can be an effective technique
Use Cases

• Use Case: a story describing how the system and a user interact to accomplish a user task
• A form of User Centered Analysis – capturing requirements from the user’s point of view
  – Identify capabilities required by different users
  – Solve the right problem
  – Describe the “business logic” of the system
• Use cases specify a subset of functional requirements
  – Only system behavior observable to the user
  – Does not typically address quality requirements
• Use cases should not specify design or implementation (including UI design)

Scenario Analysis Process

Applying scenario analysis in the requirements process
• Requirements Elicitation
  – Identify stakeholders who interact with the system
  – Collect “user stories” - how people would interact with the system to perform specific tasks
• Requirements Communication (ConOps)
  – Record as use-cases with standard format
  – Use templates to standardize, drive elicitation
• Requirements verification and validation
  – Review use-cases for consistency, completeness, user acceptance
  – Apply to support prototyping
  – Verify against code (e.g., use-case based testing)
Identifying Actors

- Actors – identifies the roles different users play with respect to the system
  - Roles represent different classes of users (users with different goals)
  - Actors carry out use cases
- Helps identify requirements for different kinds of users
  - “How would depositors use the system?”
  - “How would a library patron use the system?”
- Diverse classes of users may have different goals and require different interfaces
  - E.g., users vs. administrators vs. content providers

Scenario Elicitation

- Each class of actor is interviewed and/or observed
  - How do you do task T?
  - How will the user interact with the system to do X?
- Collect in the form of “user stories”
  - Documented as scenarios (informal or standardized)
  - Identify relative priorities of tasks
  - Resolve conflicts, tradeoffs
Creating Use Cases (Basic)

- Identify a key actor and purpose
  - The purpose informs the use case title and description
- Identify the main flow (ideal path) from the starting point to the result
  - Preconditions: anything that must be true to initiate the Use Case
  - Trigger: event, if any, initiating the Use Case
  - Basic Flow: sequence of interactions from the trigger event to the result
  - Alternative Flows: identify sequences branching off the Basic Flow
  - Exceptions: identify responses to error conditions

Guidelines for Good Use Cases

- Use Cases should express requirements, not design
  - Focus on import results that provide value to specific actors
    - I.e., if nobody really cares about the outcome, it is not a good use case
  - Focus on what the actor is doing, not the details of how
    - Not: “The user left-clicks on the radio button labeled Balance and presses the Enter button”
    - “The user elects the option to view the balance.”
- Looking for a small number of use cases that capture the most important interactions
  - Read the IBM Use Case paper
Example Use Case

- Avoids design decisions
- References other use cases
- References more precise definitions where necessary
- Some terms need further definition (e.g. PIN)

Questions?
Deliverables Walkthrough

- Consider: What kinds of questions should your documents answer?
  - Assume a manager unfamiliar with the project is reviewing your status
  - Would your documents answer key questions about the project goals and current status?
- Team page: Who is on the team and what are their skills?
- Project plan
  - Who is responsible for which tasks?
  - What are the anticipated risks and what are you doing to mitigate them?
  - What is your development process and how does it help address the risks?
- Detailed Schedule & Milestones
  - What is the project schedule of tasks and deliverables?
  - What is the current status relative to schedule?

Walkthrough (2)

- Software Requirements
  - 2. ConOps: What capabilities will the software provide the user or customer?
  - 3. Behavioral Requirements: What are the detailed technical requirements?
    - Specific inputs accepted & outputs generated
    - Detailed behavior of any computation (e.g., sort, error responses)
  - 4. Quality Requirements: objective requirements for software qualities (e.g., reliability, performance)
- Software Design
  - Architecture: How is the software organized into components? How does it work (function)? Where is each requirement implemented (traceability)?
  - Module Interfaces: What are the component interfaces?
Walkthrough (3)

- **Quality Assurance**: How will you check whether the software satisfies functional and quality requirements?
  - Reviews: Which artifacts/properties will be checked by review?
  - Test Plans: How will you test the software?
- **User Documentation**: How will users understand how to install and use the application?
- **Code Documentation**: What do I need to know to find parts of the code responsible for implementing any given requirement or part of the design?
  - How is the code organized in the repository?
  - What does this code component do?