Requirements

John D. McGregor
Clemson University
Dept.of Computer Science
johnmc@cs.clemson.edu

Requirement

- A requirement is
  - something the system must be able to do in order for the customer to be satisfied (functional)
  - some property the system must possess in order for the customer to be satisfied (non-functional)

Process

- Elicitation
  - how we have done it - employees
  - how should we be doing it - domain experts
  - written as told in the domain
- Specification
  - organize and rewrite in technical terms
  - use some method

Types of Requirements

- Physical environment
- Interfaces
- Human factors
- Functionality
- Documentation
- Data
- Resources
- Security
- Quality Assurance

Characteristics of Requirements

- Correct
- Consistent
- Complete
- Realistic
- Really what the customer wants
- Verifiable
- Traceable

Expressing Requirements

- Can use any of the typical ways of expressing information
  - data abstraction - UML class diagram
  - state model - UML statecharts
  - algorithmic approach - UML activity diagram
- Basic statement is a scenario
  - UML use case model

UML use case

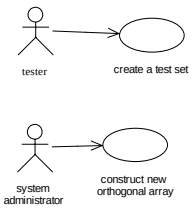
- Actor
  - a user of the system
  - abstract actor
- Use case
  - the description of a use of the system by an actor
  - abstract use case

Actors for OATS tool

- Tester
  - uses the system to define a set of test cases
- System administrator
  - uses the system to store a machine-readable form of the orthogonal array

Use case diagram

- The UML use case diagram has an icon for each actor and an oval representing each use case
- The diagram links to the detailed descriptions of each use case



Use Case Model

- In many projects this is the primary requirements description
- Describes “uses” of the system
- Captures all the actors
- Consists of actor descriptions and use cases

Actor description

- **Name:** **Abstract: Y/N**
- **Description: (Role)**
- **Skill Level:**
- **Actor Profile:**

Use Case	Primary/Secondary	Personality[1]	Relative Frequency

[1]Values are : Initiator, Server, Receiver, Facilitator

Use Case

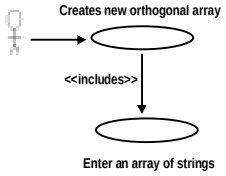
- **Use Case ID:**
- **Use Case Level:**
- **Scenario:**
- **Actor:**
- **Pre-conditions:**
- **Description:**
- **Trigger:** The user initiates an action by...
- The system responds by...
- **Relevant requirements:**

Use Case, cont'd

- **Post-conditions:**
- **Alternative Courses of Action:**
- **Extensions:**
- **Exceptions:**
- **Concurrent Uses:**
- **Related Use Cases:**
- **Decision Support**
- **Frequency: Criticality: Risk:**
- **Modification History**
- **Owner:**
- **Initiation date:**
- **Date last modified:**

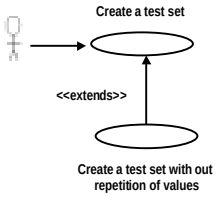
Relationships among use cases - *includes*

- A new use case *includes* an existing one when the scenario for the new use case aggregates the steps of the scenario for the existing one



Relationships among use cases - *extends*

- A new use case *extends* an existing use case when the new use case modifies the existing one by adding specific sub-steps between the existing steps of the scenarios



Roles

- User
- Customer
- Analyst
- Designer
- Tester

Validation Technique - exit criteria for some phases

- Guided Inspection
  - build test cases from the use cases, add detail sufficient to reach a single correct answer
  - conduct an inspection session
    - use testers in role of building test cases
    - include users as evaluators of answers
    - developers can evaluate answers for clarity
  - evaluate the effectiveness by measuring how many problems are found here vs how many are found later

Validation Criteria

- Correct
  - correctly capture the user's needs
- Consistent
  - one requirement does not contradict another requirement
- Complete
  - all "needed" requirements are included
  - this must be balanced by resources available

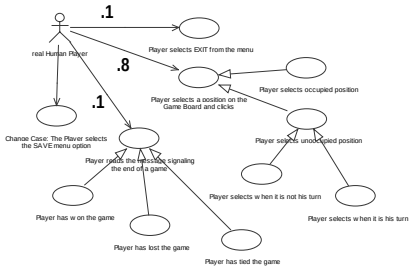
Inspection process

- A model has been created and must be validated
- A team is assigned that includes those who developed the model and a set of testers
- The testers use the scenarios of the use cases to write very specific test cases
- In an inspection session, the the developers show the developers how the model represents the test case

Model testing

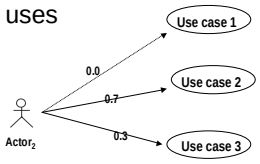
- When the developers can not adequately explain using the model, a defect has been found
- This technique can be used as the exit criteria for requirements, analysis and design phases

Actor profile



Actor Profile

- Capture these in the actor description
- Assume 0 probability for all untriggered uses



Actor Profile

Configuration Management

- Continuous snapshot of state of project
- All “important” project artifacts should be placed under “revision control”
- System prevents conflicting changes
- System allows the project to “roll back” to a previous state
- System provides a central repository that is always accessible

Configuration Management Process

- An initial artifact is created based on an assignment of responsibility
- The artifact is committed to the cm system
- A user “checks out” the artifact with either read or read/write access
- Authority to change the artifact may reside in a single “owner”or may be anyone who obtains a write lock from the cm system

Conclusion

- Must have a clear view of what the product is required to do
- Taking a use perspective makes it easier to be certain that the model is
  - correct
  - complete
  - consistent
- The use case model is validated to ensure quality