Requirements Engineering

David Janzen

Team Assignments

- Team Osos: Northwest
  - Robert Cook, Team Lead
  - Pavan Trikutan
  - Yeongshnn Ong
  - Vicki Lindem
  - Pete Corey
- http://wiki.csc.calpoly.edu/307S08osos

CAL POLY

Team Assignments

- Team Paso: Northeast
  - Natasha Lemak, Team Lead
  - Braydn Reynolds
  - Allen Pasch
  - Andrew Harrison
  - Mark Wazny
- http://wiki.csc.calpoly.edu/307S08paso

Team Assignments

- Team Pismo: Southeast
  - Dominic Camargo, Team Lead
  - Phillip Dettorre
  - Donald Kirker
  - Wesley Hamilton
  - Matthew Robertson
  - David Little
- http://wiki.csc.calpoly.edu/307S08pismo

Team Assignments

- Team Avila: Southwest
  - Kevin Finn, Team Lead
  - Henry Phan
  - Andrew Foong
  - Jay Schultz
  - Devon Laws
- http://wiki.csc.calpoly.edu/307S08avila

Project Assignments

- Setup Wiki Structure
  - Learn to use the Trac Wiki
    - See http://trac.edgewall.org/wiki/TracWiki
    - Practice in Sandbox
  - Add to your wiki homepage
  - Create roadmap structure
    - see http://wiki.csc.calpoly.edu/brackets/roadmap

Project Assignments

- Requirements
  - Add a link to a new requirements page from your requirements milestone in your roadmap
  - Document requirements on this new page
  - Document acceptance tests here as well
  - Due 4/11/08 (next Friday)

Intro to Software Requirements

- What question do software requirements answer?
  - Who, what, when, where, why, how

    What is the system to do?

    Who are the system user groups?

    Business case tells us why (and perhaps who, when, where).
    Project plan tells us when and who.
    Architecture tells us how.

Why do we care?

- Requirements issues are among the most commonly cited for project failure
  - See success/failure factors in http://www.standishgroup.com/sample_research/chaos_1994_2.php
- Sprint LARS
- Cost of avoiding/fixing defects increases as project progresses
  - Cheapest in requirements development

IEEE Definition of Requirement

- IEEE Standard Glossary of SE Terminology

  1. A condition or capability needed by a user to solve a problem or achieve an objective.

  2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.

  3. A documented representation of a condition or capability as in 1 or 2.
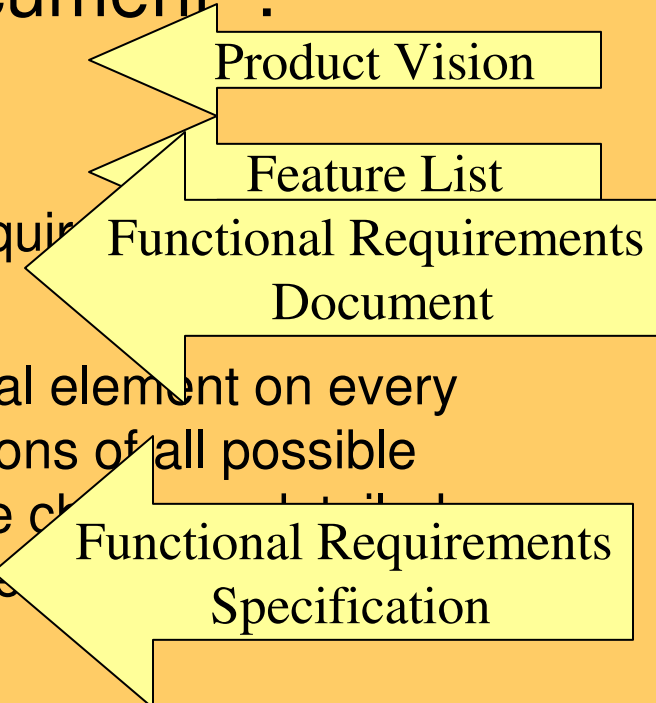
Types of Requirements

- Business
  - High-level objectives of the organization or customer who requests the system
  - Documented in a Vision and Scope document
- User
  - User goals or tasks that the users must be able to perform with the product
  - Use-cases often used to capture these
  - Ex. Make a reservation

Types of Requirements

- Functional
  - Specify the software functionality that the developers must build into the product to enable users to accomplish their tasks.
  - Ex. The system *shall* mail a confirmation to the user

- Non-functional
  - Quality attributes, performance goals, reliability, …
  - Ex. Reservation request submissions should receive a response in less than 10 seconds

Requirements Documents

- "different organizations might call any of the following a 'requirements document'":

  1. Half-page software product vision
  2. Two page list of key features
  3. 50 page list of detailed end-user-level requirements

  4. 250 page exhaustive listing of every visual element on every screen, input-field-by-input-field descriptions of all possible input conditions, all possible system state changes, detailed description of every persistent data element

Product Vision

Feature List

Functional Requirements Document

Functional Requirements Specification

1. McConnell, IEEE Software, Sept/Oct 2000, http://www.stevemcconnell.com/ieeesoftware/eic13.htm

CAL POLY

Requirements Problems

- Insufficient User Involvement
- Creeping User Requirements
- Ambiguous Requirements
- Gold Plating
- Minimal Specification
- Overlooked User Classes
- Inaccurate Planning

Excellent Requirements

- Statements
  - Complete, correct, feasible, necessary, prioritized, unambiguous, verifiable

- Specification
  - Complete, consistent, modifiable, traceable

- Discussion Question:
  - What would you add to the list?

Phases of a Software Lifecycle

- Standard Phases
  - Requirements Analysis & Specification
  - Design
  - Implementation and Integration
  - Operation and Maintenance
  - Change in Requirements
  - Testing throughout
- Phases promote manageability and provide organization

Requirements Analysis & Specification

- Problem Definition —> Requirements Specification
  - determine exactly what is the client (and user) problem
    - in their environment - with their environmental constraints
  - develop a contract with client
    - exactly what the software/computer solution will do
- Difficulties
  - client asks for wrong product or developer "knows better" (want vs need)
  - client is computer/software illiterate or developer domain illiterate
  - specifications will be ambiguous, inconsistent, incomplete (adequacy?)

Requirements Analysis & Specification

- Validation
  - extensive specification reviews check that requirements satisfy client wants
  - look for ambiguity, consistency, incompleteness
  - check for feasibility, testability
  - develop system/acceptance test plan

Requirements Elicitation

- Discovering user requirements
- Passive or Active Elicitation
  - Steve McConnell says,

    "The most difficult part of requirements gathering is not the act of recording what the users want; it is the exploratory, developmental activity of *helping users figure out what they want*."

Elicitation Interviews

- Do basic research first!
  - do NOT ask questions that have been answered
  - show you followed up to previous sessions
- Focus your questions
  - Beware of broad questions
    - Sometimes they can uncover missed requirements
    - short, simple, answerable: yes/no preferred
  - if complex, ask multi-part questions
  - *use models / documents as points of reference*
  - use a parking lot for tangent ideas

CAL POLY

What vs. How

- Remember: distinguish "requirements" from "design"

- Requirements are about "black box" external behavior of the proposed system
  - black box vs white box concepts
 - *software as transform of input to output*

Feedback

- Give feedback on the answers
  - offer an example, "is this what you mean?"
  - narrow the question if you must
  - do not move on until you understand or agree to look further
  - think like a customer who'll have to live with this thing you're going to describe
  - think like a coder who'll have to build it!

CAL POLY

Main Themes

- You are writing Requirements
- Your job: serve the customer
  - be prepared
  - make the customer's job as easy as possible
- Customer's job: help you serve them
- Be professional at all times

CAL POLY

Use Cases and Scenarios

- Use case: a set of scenarios tied together by a common user goal

- Scenario: a sequence of steps describing an interaction between a user and a system (Fowler)

Game Player

Make Move
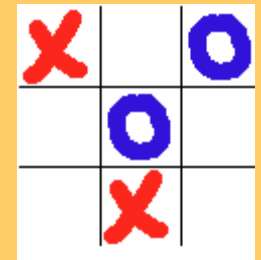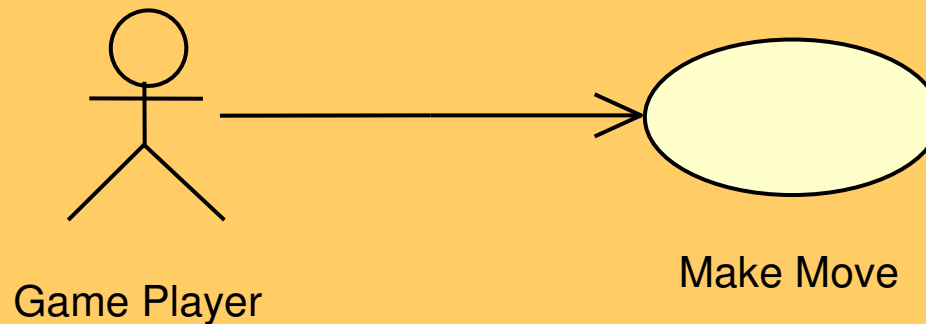
Actors

- Actors: roles that users (or systems) play
  - Actors carry out use cases
  - A single user could play several roles
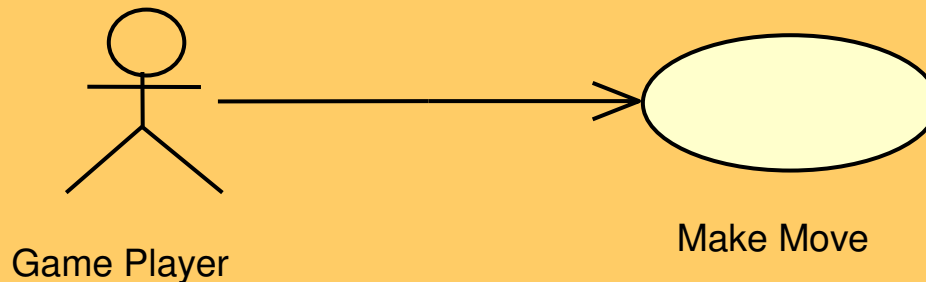 - Multiple users could play the same role

Game Player
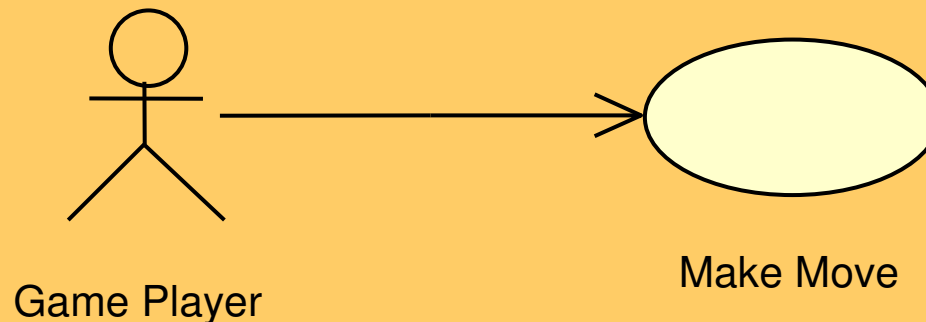
Make Move

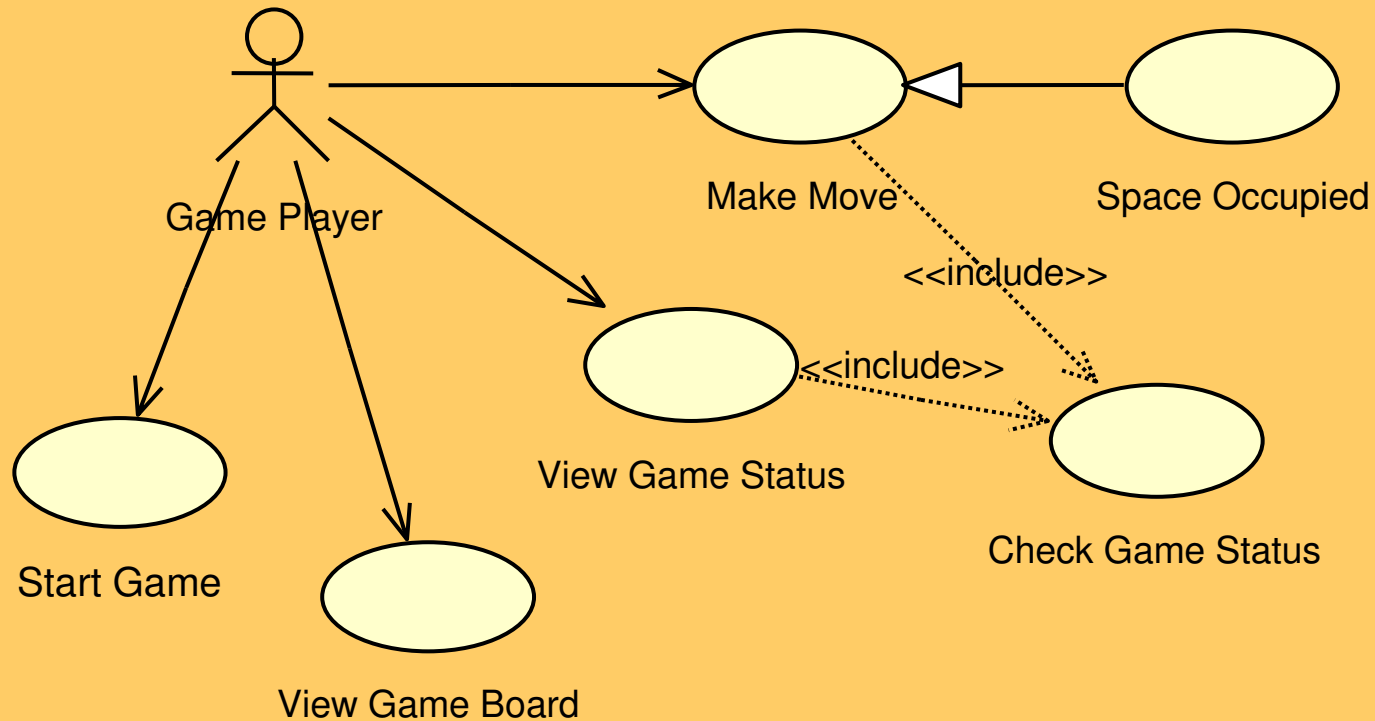Alex and Katie play tic-tac-toe against each other.

Alex and Katie are each filling the Game Player role.

Scenario

- Scenarios for the 'Make Move' use case:
  - A Game Player places a symbol on an open square on the Game Board
  - A Game Player places a symbol on an occupied square on the Game Board
    - Original symbol continues to occupy square
    - Allow Game Player to select another square

Game Player

Make Move

Use Case Diagram

Game Player

Make Move

Space Occupied

<<include>>

<<include>>

View Game Status

Check Game Status

Start Game

View Game Board

Use Case Diagram

- Relationships

Generalization:
alternate scenario or
same use case with
extra functionality

Game Player

Make Move

Space Occupied

<<include>>

<<include>>

View Game Status

Check Game Status

Start Game

View Game Board

Include:
redundant functionality

CAL POLY

Acceptance Test

- Scenario for the 'Make Move' use case:
  - 2.2.3 A Game Player places a symbol on an occupied square on the Game Board
    - Original symbol continues to occupy square
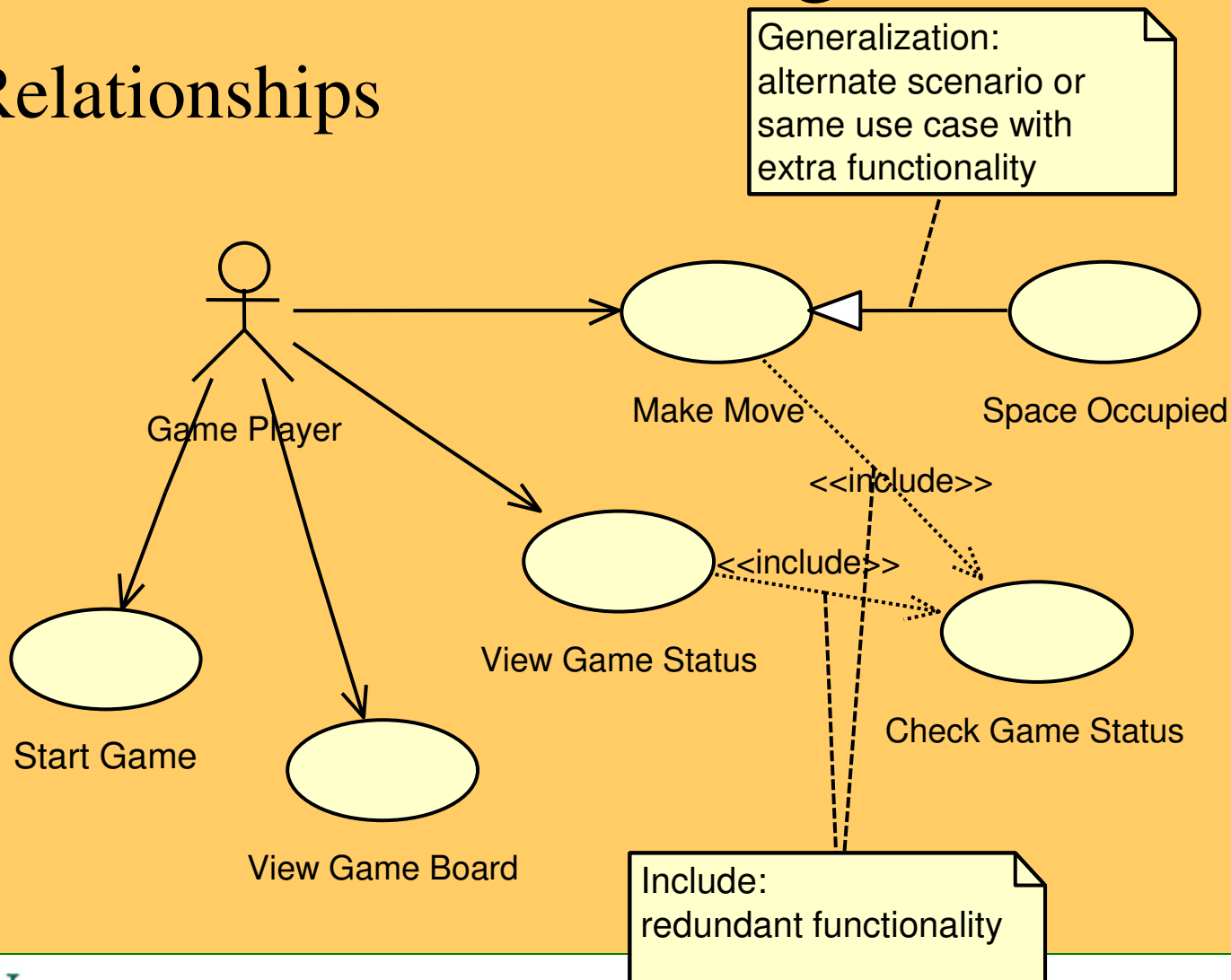   - Allow Game Player to select another square

- Corresponding acceptance test
  - 2.2.3-a
    - Player 1 enters 'X' in square 3
    - Player 2 enters 'O' in square 3
    - System prompts player 2 to select different square
    - Player 2 enters 'O' in square 4
    - Board has 'X' in square 3 and 'O' in square 4