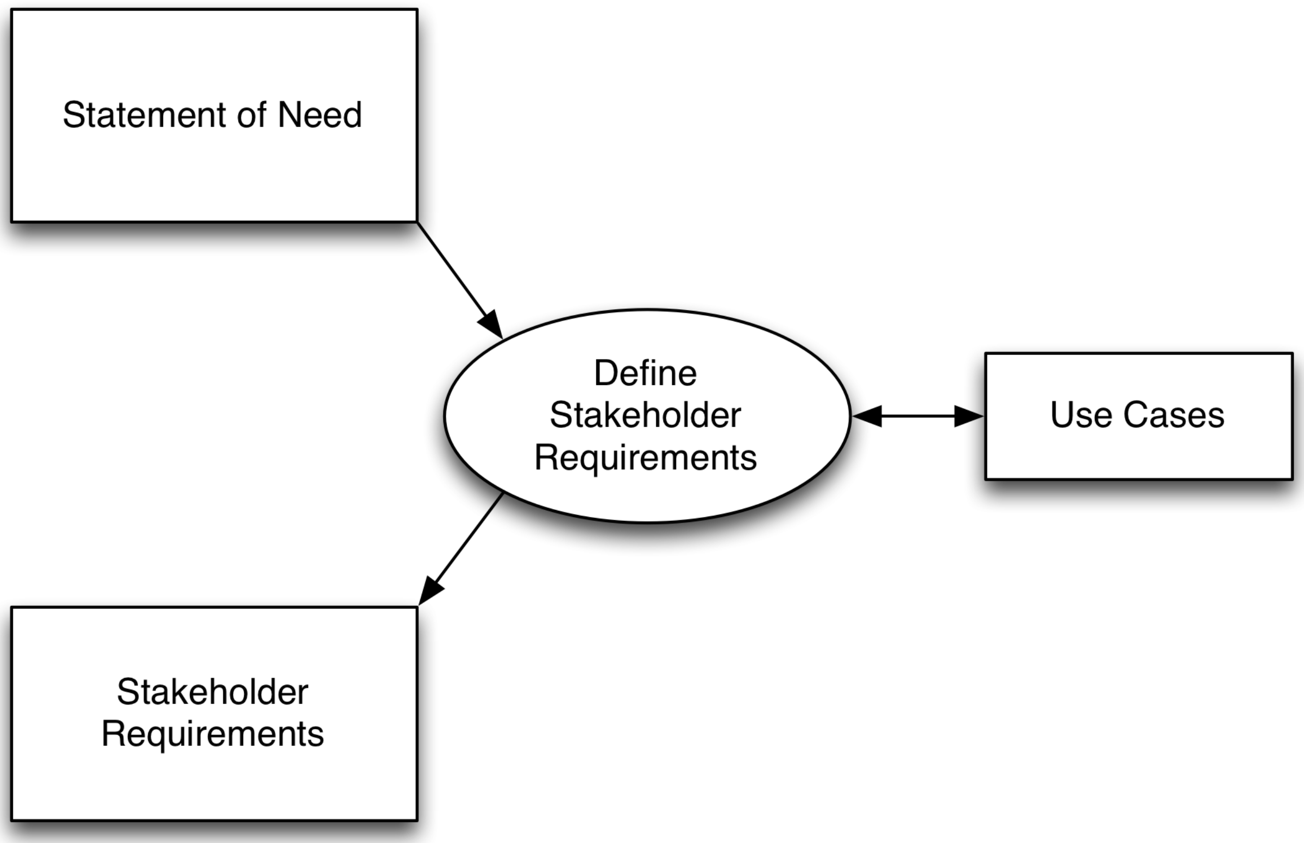
Use Cases

- Use Case components
- Documenting a Use Case
- Structuring Use Cases
- Use Case diagrams

Where Are We in the Module?



Use Cases: Functional Requirements

- Use Cases can be used to
  - Capture/structure the System's functional (behavioural) requirements
  - Define the System's functional acceptance tests
- A Use Case describes how
  - An Actor achieves a goal using the System
    - In terms of Scenarios (sequences of actions) performed by the System and various Actors
      - A Customer has the goal "Buy tickets for a film"
  - Interests of the Stakeholders are protected by the System
    - What the System guarantees to the Stakeholders
      - A Manager has the interest "Make money from ticket sales"

Use Case Actors

- In Use Case terminology, an Actor is someone or something external to the System, that interacts with the System
  - A person
  - A peripheral device/sensor/actuator
  - Another computer/system
  - An institution/company
  - …

Actors: Cinema Tickets System

- Movie Customer
  - A *person* who wants to see a film.
- Clerk
  - A *person*, an employee of the cinema who sells tickets and refreshments to people.
- Ticket System
  - A *software system* that has an inventory of what tickets have been sold, and how many seats are left.
- Cinema Manager
  - A *person*, who must ensure that customers are happy, employees are satisfied and the movie theatres are full..

Kinds of Actors

- Stakeholder
  - An Actor whose *goals* are met by the system, and/or whose *interests* are protected by the system
    - Some Stakeholders may not have goals, only interests
    - Some goals may be "to provide a service"
- Primary Actor (of a Use Case)
  - The Stakeholder whose goal is met by the Use Case
  - May trigger interaction directly, or via an intermediary
    - E.g., timed, or periodic, event
      - *"Job must be run every Friday"* - who is the Stakeholder?
- Supporting Actors (of a Use Case)
  - Other Actors whose participation is necessary to achieve the Primary's goal

Actors and Goals: Multiplex Cinema

- Actors and their goals
  - Customer : “Buy tickets for a film”, “Get a refund”, …
    - With Sales Clerk as Secondary Actor
  - Scheduler : “Update schedule”, “Forecast demand”, “Allocate a screen”, …
  - Cinema Publicist : “Show current schedule”, …
  - Film Distributor : “Show tickets sales for my film”, …
  - VAT Inspector : “Audit ticket sales”, …
- Stakeholders with interests, but no goals
  - Manager : “Make money from ticket sales”, “Obey the law on viewer age limits”, “Obey the law on VAT returns”, …

Scenarios

- Description of a particular sequence of actions or interactions
  - Captures a particular way of *achieving*, or failing to achieve, a Stakeholder goal
  - One particular path through a Use Case
- Concrete: personalised with names/values : a "story"
  - How "Jane buys three tickets for the 2:15 pm showing of *Batman: The Dark Knight*"
  - Can be useful focus during requirements elicitation
- General: impersonal placeholders
  - How "the customer buys some tickets for a showing of a film"
  - Useful when documenting the Use Case

Main Success Scenario

- The usual behaviour:
    - *Successfully* buy some tickets for a showing of a film
- The Primary Actor triggers interaction, to achieve a goal
    - *Customer requests some tickets for a showing of a film*
- The Scenario defines the usual way the goal is achieved
    - *Sales Clerk checks there are no age limits for the film*
    - *Sales Clerk checks sufficient tickets are available*
    - *Sales Clerk prints the tickets*
    - *Sales Clerk tells the Customer the price*
    - *Customer hands over the money*
    - *Sales Clerk hands over the tickets*

Form of Scenarios

- All Scenarios have the same precondition
  - The required/relied upon state of world prior to the interaction
- A Scenario is described by steps
  - The flow of events, on basic and alternative paths
- Each Scenario has its own postcondition
  - The guaranteed state of world after the Use Case exits
  - Success precondition holds for successful exit, achieves goal
  - Exception scenarios can describe error cases
    - May result in different postcondition from the success scenario
  - All scenarios must establish the minimal postcondition
    - Guarantee to protect Stakeholder interests (integrity constraints)

Secondary Scenarios

- For each step of the main success scenario:
  - *Alternative scenario*:
    - Is there some other action that can be done here?
  - *Exception scenario*:
    - Is there something that can go wrong here?
- Is there some alternative or exceptional behaviour that could happen at *any* step?
  - Abort transaction
  - Power failure
  - …

Scenario Examples: Cinema

- Main Success Scenario
  - *Buy tickets for a showing of a film*
- Secondary Scenarios
  - Alternatives - other ways of being successful
    - *Buy tickets for a different showing of a film*
    - *Buy tickets for a different film*
  - Exceptions - ways of failing
    - *Tickets sold out*
    - *Customer too young for film rating*
      - *Customer requests some tickets for a showing of a film*
      - *Sales Clerk asks the Customer's age*
      - *Customer responds with their age*
      - *Sales Clerk says they are too young for the film*

Use Case

- Named capability of the system
  - "Buy tickets"
  - "Withdraw money from ATM"
    - Name should reflect Primary Actor's goal
- Collection of the scenarios related to a particular goal
  - The interaction triggers, to achieve Primary Actor's goal
  - Each Scenario defines one way the goal is, or is not, achieved
  - The Use Case, the collection of scenarios, covers all possible results of triggering the interaction

Use Case Documentation: Format

- Textual description of conditions and scenarios
- No standardised format
  - Each author has their own preference
    - Not defined in the UML reference manual
    - Rational Unified Process™ (RUP) has a template
      - Use Cases were invented by Ivar Jacobsen, one of the RUP/UML trio (along with Rumbaugh and Booch)
      - But they are a general purpose technique
  - Format can be tailored to project requirements
    - Varying amount of formality/detail
    - [Cockburn 2001]: "low ceremony" *vs.* "high ceremony" projects
  - We use an adaptation of [Cockburn 2001] template

Use Case Documentation: Template

- **Name: < an active verb phrase, describing the goal >**
- **Context:** < longer description of the goal >
- **Actors**
  - ***Primary Actor*: < the Actor with the goal >**
  - ***Supporting Actors, Stakeholders*:** < interests to be protected >
- **Precondition**: < required state of world prior to Use Case >
- **Minimal Postcondition**: < holds for all exits, protects Stakeholder interests >
- **Success Postcondition:** < holds for successful exit, achieves goal >
- **Trigger**: < what starts the Use Case, e.g., Primary Actor, internal event, … >
- **Main Success Scenario: < steps >**
- **Secondary scenarios:** < steps, postcondition >
- **NFRs**
- **Other information** (catch all)

Example: “Withdraw Cash from ATM”

- **Primary Actor**: CardHolder
- **Supporting Actors**: Bank
- **Precondition**: Customer has an ATM card
- **Trigger**: Customer inserts card into ATM
- **Main Success Scenario**
  - 1. ATM offers menu of services. CardHolder selects “Withdraw Cash”
  - 2. ATM reads account details, PIN from card
  - 3. ATM passes account details to Bank. Bank validates account details
  - 4. CardHolder enters PIN. ATM validates PIN
  - 5. CardHolder selects amount to withdraw
  - 6. ATM notifies Bank of amount. Bank accepts request
  - 7. ATM delivers cash, returns card
- (a few) **Secondary Scenarios**
  - 1.1-5.1: CardHolder cancels: ATM discards intermediate data, terminates transaction
  - 4.2. PIN not valid : ATM prompts CardHolder for another PIN
  - 6.1. Insufficient funds : Bank rejects request, ATM prompts for new amount
- **Success Postcondition**: CardHolder has cash, and similarly reduced balance in Bank

Scenario Body

- Defines the steps (the flow of events) taken by the Actor(s) and the System to achieve the goal
- Again, no standardised format
  - Text is best, but still many possible formats
    - Free format for "low ceremony" or preliminary Use Cases
    - Or, numbered lists for "high ceremony" detailed Used Cases
    - Or, tabular form to show who "has the ball"
  - Every step of form : "<Actor/System> <active verb phrase>"
    - *6. Customer enters credit card payment information*
  - UML, e.g., Sequence Diagram
    - Use as supplement to the text, to visualise complicated flows of events
    - Do *not* use as the only definition - text usually has more content

Secondary Scenarios

- Break out sub-steps from the Main Scenario:
  - n.m < condition > : < action >
    - *3.1. Customer details missing: System prompts for details*
    - *2.1-6.1: Customer cancels: System terminates session*
  - These can be nested
    - *3.1.1. Customer details still missing: System terminates session*
- Or, pseudo-code-like statements in the Main Scenario:
  - If, for loop, while loop
  - Parallel actions
    - Use these sparingly – do not write code!
  - Sub-steps style usually easier to understand

A Different Scenario Format

Card Holder	ATM	Supporting Actor Bank
Insert Card	Request PIN	
Enter PIN	Verify PIN	
	Show Services	
Select Withdraw Cash	Show request for amount	
Enter amount	Send account details	Validate account details
	Send Amount	Validate funds available
		Send confirmation message
	Give card	
Take card	Give cash	
Take cash		

Structuring: Inclusion Use Case

- A single step in a scenario might need some elaboration
  - The main Use Case "calls" the Inclusion Case
  - Or, the main Use Case "include"s the Inclusion Case (cf. UML)

    Edit a Document

    Trigger: User opens application

    1. User opens a document
    2. User Modifies Text
    3. User Saves a Document
    4. User closes document

    Modify Text …

    Save a Document …

Structuring: Inclusion Use Case

- The same step might be applicable to several Use Cases

Create an Order

1. Customer provides order information
2. System Saves the Order

Cancel an Order

1. Customer provides Order ID
2. System Finds the Order
3. System deletes the order

Modify an Order

1. Customer provides Order ID
2. System Finds the Order
3. Customer provides replacement information
4. System Saves the Order

Find an Order …

Save an Order …

Extension Use Case

- The main scenario can be interrupted, not under its control

  Check Balance

  Precondition: A banking operation is taking place.

  Trigger: Anytime a banking operation will change the balance of a bank account.

- "Frozen" requirements need to be modified with extended behaviour
  - The Extension Case "extend"s the main Use Case

  Supply Special Offers on Mortgages

  Trigger: In "Withdraw money" if the person has a balance greater than £5000, present offer.

Inclusion or Extension?

- Inclusion
  - The trigger event is under the control of the main Use Case
  - So, the Main case "calls" the Inclusion Use Case
- Extension
  - The trigger event is not under the control of the Main case
  - So, the Main case is "interrupted" by the Extension Use Case
  - Or, the requirements are "frozen", and Extension is the only change mechanism available
    - Warning: this can lead to difficult-to-understand documents
      - It acts like the (mythical) "come from" program statement

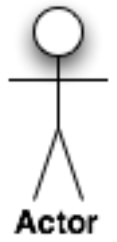
Use Case Diagram

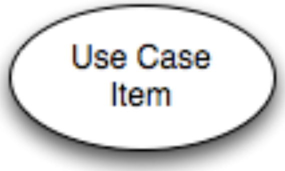
- Summarises the functionality of the whole system
- Shows packaging and decomposition of Use Cases
- Overview of relationships between Actors and Use Cases
- Useful as a graphical "table of contents"
- *Not* a means to record entire Uses Cases

  "UML defines graphical icons that people are determined to use. It does not address content or writing style, but does provide a lot of complexity for people to discuss." [Cockburn, 2001]

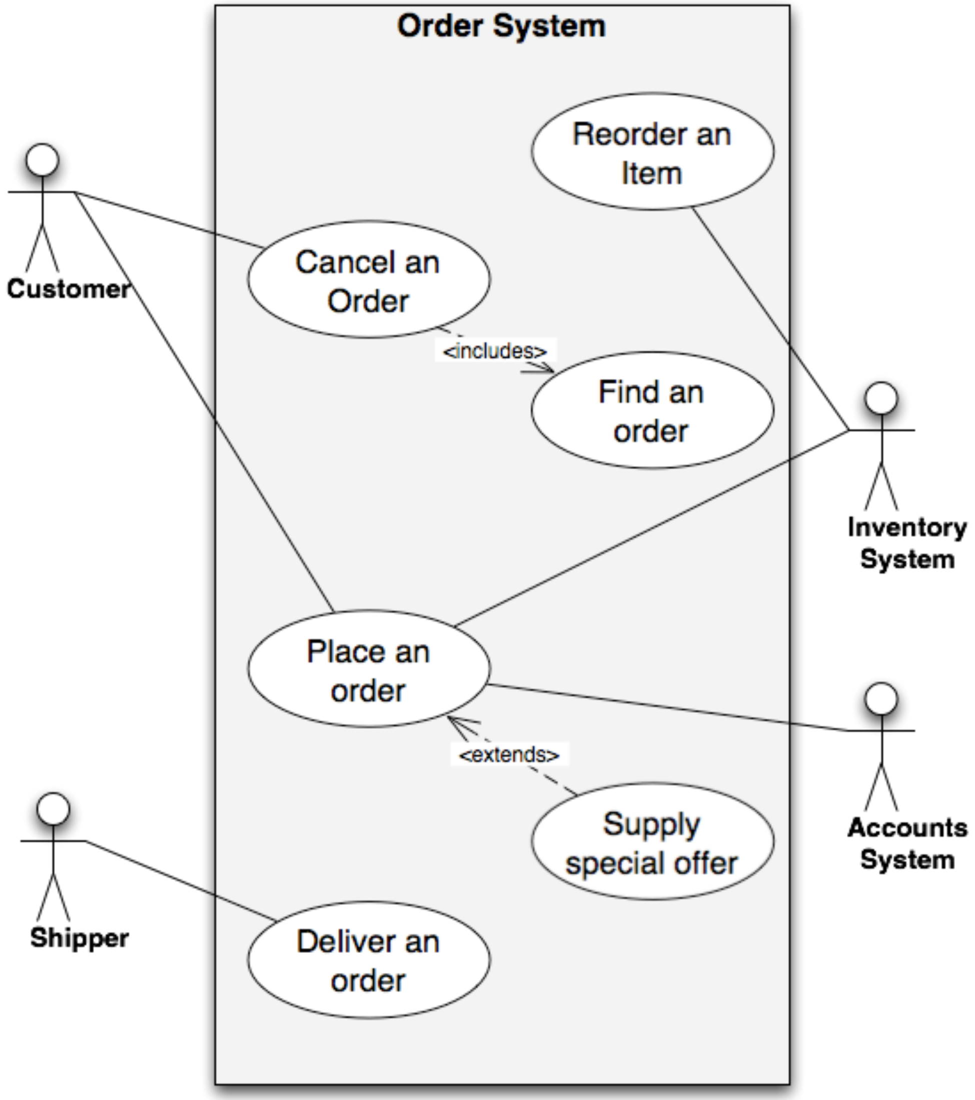
Use Case Diagram Components

- Actor: a named Stick Figure
  - The various Primary and Secondary Actors
  - Can use other icons for non-human Actors
- Use Case: a named ellipse
  - Use Case name
  - Joined by lines to its Actors
    - No distinction between Primary
    - and Secondaries
  - Inclusion and Extension relationships
- System boundary: a named box
  - Simple case merely has ellipses inside and Actors outside!
  - More complex cases show subsystem structure



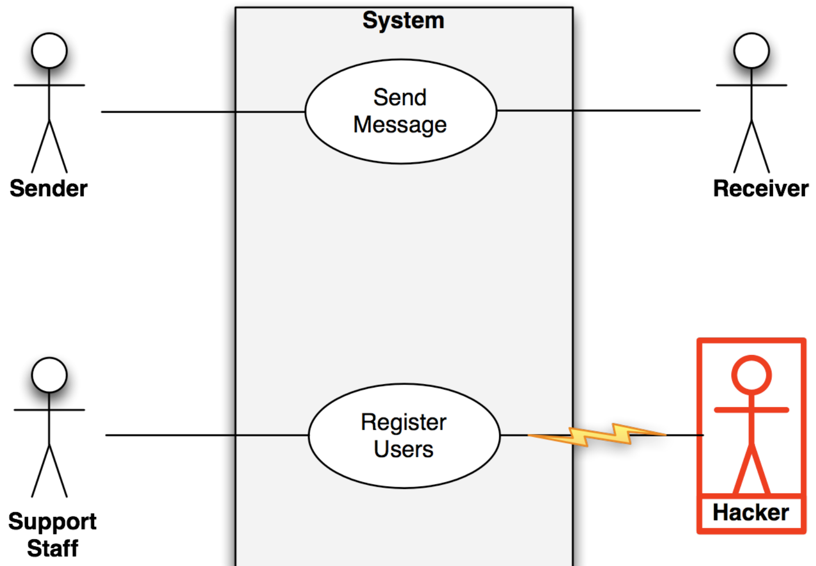


Use Case Diagram: Example



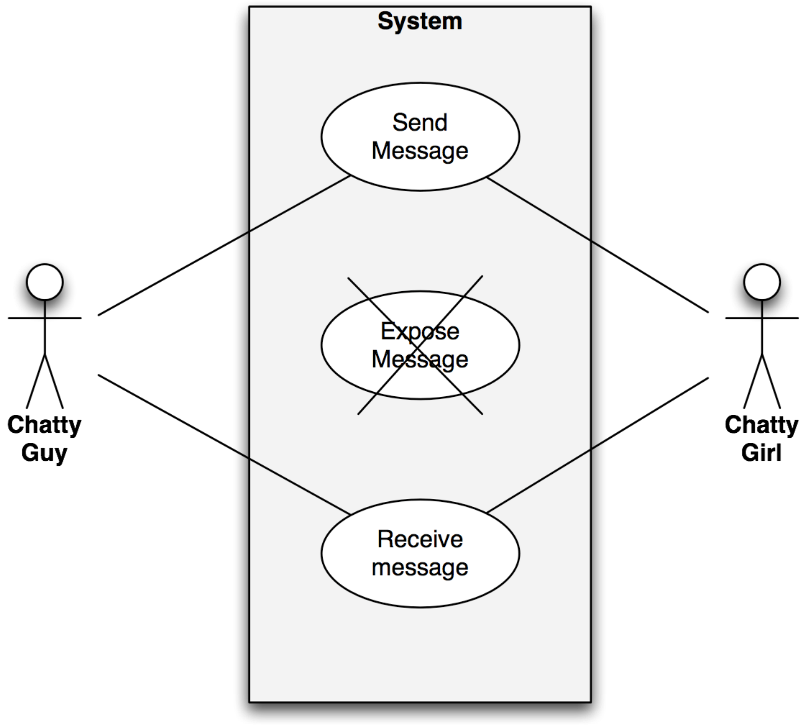
Abuse Cases

- Use Cases describe functions that are to be carried out
- Abuse/Misuse Cases can describe functions that must *not* be carried out
  - Negative Use Case
  - Forbidden Actor – example:



Negative Use Case

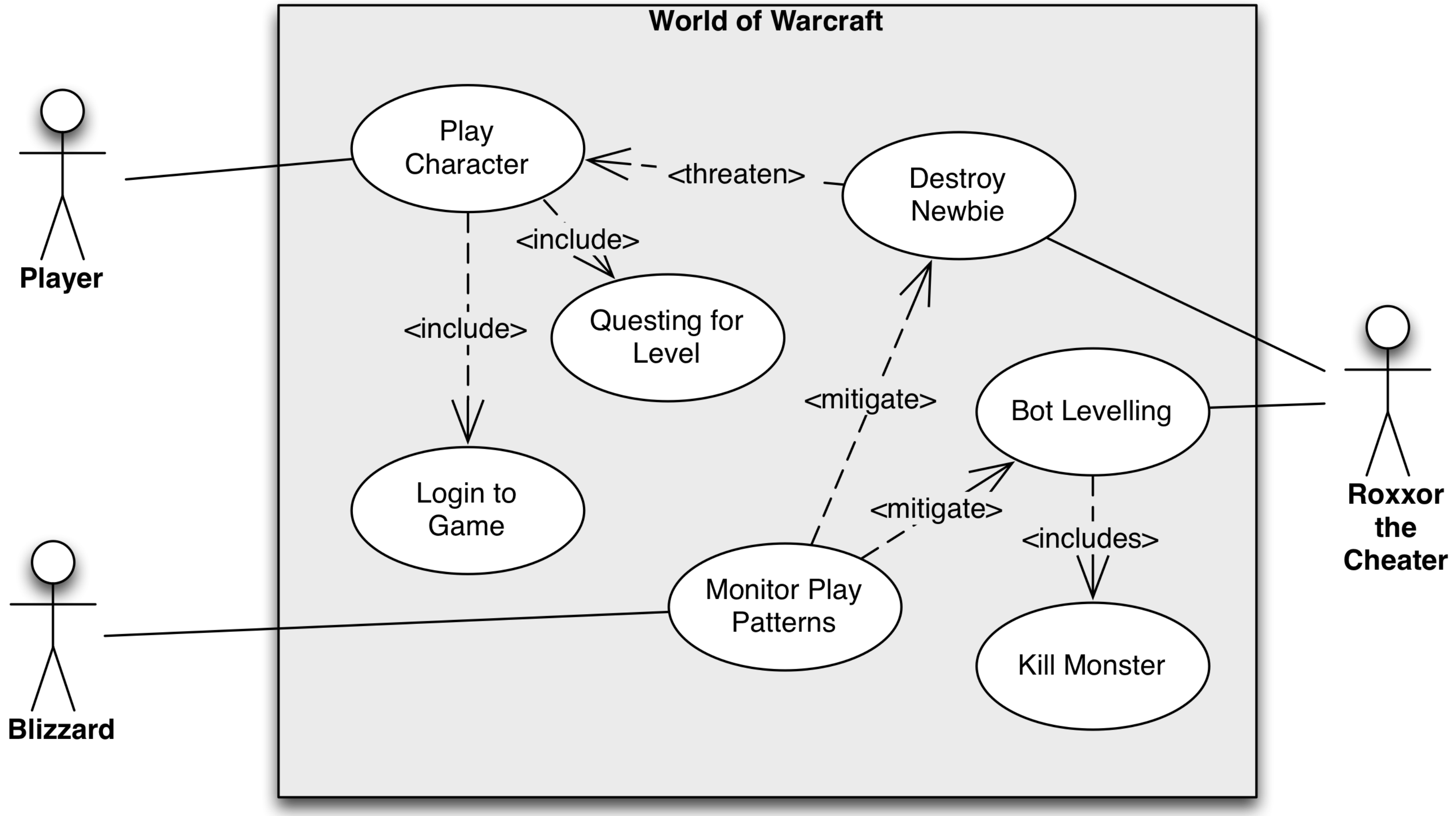
- Alternatively, a negative use case could represent misuse of the system; in particular things that should not be allowed to happen
- Or, information could be added as to how to mitigate damages from misuse



Misuse Case: Cheating Prevention

- In World of Warcraft there are players who will “bot” to become powerful.
- This threatens other people’s enjoyment because they cannot compete.
- How can we keep Roxxor the Undead Warrior from cheating and destroying Myxiplk the happy gnome with with misuse cases?

Misuse Case: Cheating Prevention



Summary

Use cases are good for:

- Modelling current work practices
- Modelling how a new system ought to operate, if it is relatively easy to elicit the Main Success Scenarios
- Uncovering
  - Error cases, and associated business rules
  - Pre- and post-conditions
  - Use Case and Scenario specific NFRs
  - Requirements on Secondary Actors (outside system boundary)
- Defining functional acceptance tests

But Use with Care!

- If modelling a new system so novel that scenarios are difficult to imagine
  - Prototype the scenarios
- Keep scenarios free of design assumptions
  - Do not write (pseudo-)code
- Not suitable for uncovering system-wide NFRs spanning several Use Cases
- Biased towards transactional style systems
  - May be better to use another approach for non-transactional systems/control systems

References

- Much of the literature on Use Cases is quite old. The original work proposing use cases was:
  - I. Jacobson, Object Oriented Software Engineering: A Use Case Driven Approach 1st Edition Addison-Wesley, 1992 (First book on use cases)
- An excellent resource for looking at how to write good use cases is
  - A. Cockburn, *Writing Effective Use Cases*, Addison-Wesley, 2001