DA-IICT

Scenario & Use Cases

Saurabh Tiwari

Developing Use Case Models of Systems

- Description of a sequence of interactions between a system and external actors
- Developed by Ivar Jacobson
  - Not exclusively for object-oriented analysis
- Actors – any agent that interact with the system to achieve a useful goal (e.g., people, other software systems, hardware)
- Use case describes a typical sequence of actions that an actor performs in order to complete a given task
  - The objective of use case analysis is to model the system
    - … from the point of view of how actors interact with this system
    - … when trying to achieve their objectives
  - A use case model consists of
    - A set of use cases
    - An optional description or diagram indicating how they are related

Use Cases

- A use case should describe the user's interaction with the system ...
  - Not the computations the system performs
- In general, a use case should cover the full sequence of steps from the beginning of a task until the en
- A use case should only include actions in which the actor interacts with the computer
  - Some views differ on this one!!!

Use Cases – Abstraction Level

- A use case should be written so as to be as independent as possible from any particular implementation / user interface design
- Essential use cases (Constantine & Lockwood)
  - Abstract, technology free, implementation independent
  - Defined at earlier stages
  - e.g., customer identifies herself
- Concrete use cases
  - Technology/user interface dependent
  - e.g., customer inserts a card, customer types a PIN

Scenarios (1)

- A scenario (according to the UML/UC community) is an instance of a use case
  - It expresses a specific occurrence of the use case (a specific path through the use case)
    - A specific actor ...
    - At a specific time ...
    - With specific data …
  - Many scenarios may be generated from a single use case description
  - Each scenario may require many test cases
- Rather used in a generic way in this course (as is often the case in requirements engineering)

Scenarios (2)

- A use case includes primary and secondary scenarios
- 1 primary scenario
  - Normal course of events
- 0 or more secondary scenarios
  - Alternative/exceptional course of events, variations of primary scenario
  - An alternative scenario meets the intent of the use case but with a different sequence of steps
  - An exceptional scenario addresses the conditions of main case and alternative cases that differ from the norm and cases already covered
  - Example with consensus as a goal
    - Primary scenario: vote in a session
    - Alternative scenario: voting in several sessions
    - Exceptional scenario: what to do with a non-registrant who wishes to vote

Types of Scenarios

- As-is scenario
  - Used in describing a current situation, usually used in re-engineering projects, the user describes the system
- Visionary scenario
  - Used to describe a future system, usually used in greenfield engineering and reengineering projects
  - Can often not be done by the user or developer alone
- Evaluation scenario
  - User tasks against which the system is to be evaluated
- Training scenario
  - Step by step instructions that guide a novice user through a system

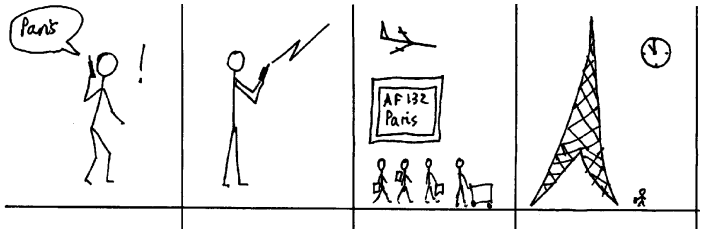
Representation of Scenarios (1)

Various approaches

- Text (informal, formal), diagrams (state, sequence ...), video, animation, comics, storyboard, collaborative workshops (pass the microphone or the ball)…
- There are specialized notation such as UML (sequence, activity, use case, interaction, and collaboration diagrams), Message Sequence Charts (MSC), Live Sequence Charts, and Use Case Maps (UCM)

Representation of Scenarios (2)

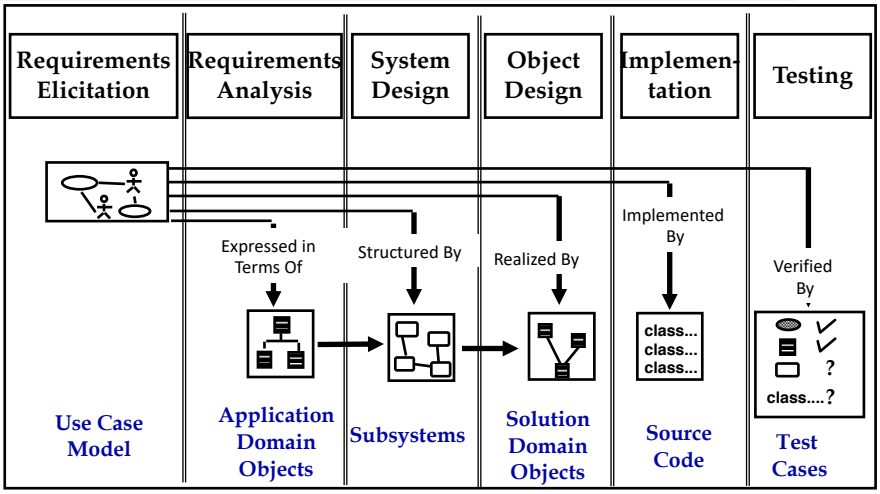
- Different representations may be useful in specific situations
  - For example, storyboards, often used in the film industry, can describe situations, roles, and sequences of tasks in a fast, compact, and polyglot way[1]



- Some scenario-based approaches are very ideological or dogmatic

[1] I. Alexander

Use Case-Driven Software Lifecycle Activities (1)

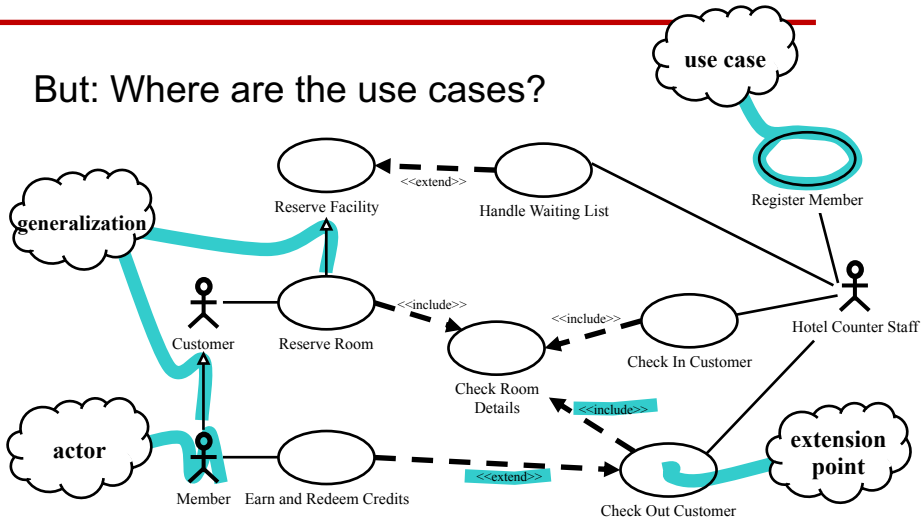


Use Case-Driven Software Lifecycle Activities (2)

- Scenarios guide elicitation, analysis, design, and testing
  - There are many scenario-based approaches
  - E.g., XP employs user stories (scenarios) to directly generate tests that will guide software design and verification
- Developers are often unable to speak directly to users
- Scenarios provide a good enough approximation of the “voice of the user”

Use Case Modeling with UML

But: Where are the use cases?



Use Case Diagrams

- To define system boundary (subject), actors, and use cases
  - Subject could be: a physical system, a component, a subsystem, a class
- To structure and relate use cases
  - Associate actors with use cases
  - Include relation
  - Extend relation
  - Generalization (of actors and use cases)

Use Case Diagrams – <<include>> (Inclusions)

- Inclusions allow one to express commonality between several different use cases
- Inclusions are included in other use cases
  - Even very different use cases can share a sequence of actions (reuse)
  - Enable you to avoid repeating details in many use cases (consistency)
- An inclusion represents the execution of a lower-level task with a lower-level goal (→ decomposition of complex tasks)
- Base use case references the included use case as needed
- Base use case cannot exist without the included use case
- When included use case is done, control returns to base use case
- Disadvantage: have to look in multiple places to understand use case

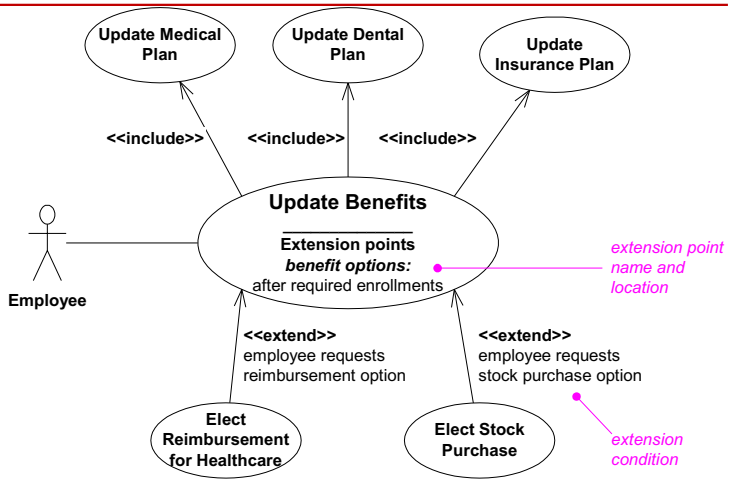
Use Case Diagrams – <<extend>> (Extensions)

- Used to make optional interactions explicit or to handle exceptional cases
- By creating separate use case extensions, the description of the basic use case remains simple
  - Note: the base use case can still be executed without the use case extension
- Extension points must be created explicitly in the base use case
- Use sparingly: there is disagreement over the semantics

Use Case Diagrams – Generalizations

- Much like super-classes in a class diagram
  - Need to satisfy “is-a” relation
- Applies to use cases and to actors
  - A generalized use case represents several similar use cases
  - One or more specializations provide details of the similar use cases
  - Inheriting use case can replace steps of inherited use case
  - Particularly useful for actors (clearer here, unlike use case generalization where the semantics are unclear – use generalization of use cases with caution)

Example: HR System



Use Case Templates (1)

- There are many different templates for use cases but they often consist of a subset of the following items:
- Identifier: unique label for use case used to reference it elsewhere
- Name: succinctly state user task independently of the structure or implementation
  - Suggested form “verb object” (e.g., Order a product)
- Authors: people who discovered use case
- Goal: short description of expected outcome from actors’ point of view
- Preconditions: what needs to be satisfied before use case can begin
- Postconditions: state of system after completion of use case
  - Minimal guarantee: state of system after completion regardless of outcome

Use Case Templates (2)

- Primary actor: initiates the use case
- Participants (secondary actors): other actors involved in use case, provide services to the system and interact with the system after the use case was initiated
- Related requirements: identifiers of functional and non-functional requirements linked to the use case
- Related use cases: identifiers of related use cases
  - Specify relationship: e.g.
    - Supposes use case UC2 has been successfully completed
    - Alternative to use case UC3
    - ...
- Description of events (scenarios)
  - Different use case description formats
  - Narrative, Simple column, Multiple columns

Use Case Templates – Narrative Form

- Paragraph focusing on the primary scenario and some secondary ones
- Very useful when the stakeholders first meet

A User inserts a card in the Card reader slot. The System asks for a personal identification number (PIN). The User enters a PIN. After checking that the user identification is valid, the System asks the user to chose an operation…

Use Case Templates – Simple Column Form

Linear sequences (main and alternatives)

1. A User inserts a card in the Card reader slot.

2. The system asks for a personal identification number (PIN).

3. The User enters a PIN.

4. The System checks that the user identification is valid.

5. The System asks the user to chose an operation

1.a The Card is not valid.

1.a.1. The System ejects the Card.

4.a The User identification is not valid.

4.a.1 The System ejects the Card.

Use Case Templates – Multiple Column Form

- One column per actor
- Allows for a more detailed view

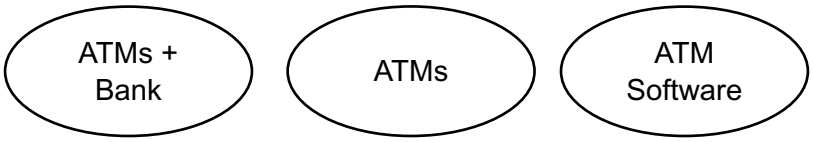
User's actions	**System responses**
1. Insert a card in the Card reader slot. (card is not valid see alternative 1.1)	2. Ask for a personal identification number (PIN).
3. Enter a PIN.	4. Check that the user identification is valid. (identification is not valid see alternative 4.1) 5. Ask User to chose an operation

Use Case Templates – Example Template

- **A. Name**: Give a short, descriptive name to the use case
- **B. Actors**: List the actors who can perform this use case
- **C. Goals**: Explain what the actors are trying to achieve
- **D. Preconditions**: State of the system before the use case
- **E. Description**: Give a short informal description
- **F. Trigger**: Describe the event that initiates the activity
- **G. Summary**: Summarize what occurs as the actors perform the use case
- **F. Related use cases**: (in accordance with use case diagram)
- **G. Steps**: Describe each step using some form (single/multiple columns)
- **H. Postconditions**: State of the system following completion
- **I. Special Requirements**: Non-functional requirements and constraints

Use Case Development – Example: ATM System (1)

1. Determine candidate system scope and boundaries
   - Identify stakeholders
   - Identify problem - Create problem statement
   - Use interviews and other techniques



1	2	3
- Customer	- Customer - Bank	- Card reader - Cash dispenser - Key pad - Touch screen - Printer - Bank

Use Case Development – Example: ATM System (2)

2. Identify actors
   - Who interacts with the system?
     - Who or what uses the system?
       - For what goals?
       - What roles do they play?
     - Who installs the system?
     - Who or what starts and shuts down the system?
     - Who maintains the system?
     - Who or what gets and provides information to the system?
     - Does anything happen at a fixed time?
   - Check for possible actor generalization

Example ATM with scope 2

Bank Customer

Installation Technician

Administrator

Administrator

Bank

Weekly Reports

Use Case Development – Example: ATM System (3)

2. Identify actors (cont'd)
   - Choose actors' names carefully
   - Actors give value, get value from system or both
   - Should reflect roles rather than actual people
     - An actor specifies a role an external entity adopts when it interacts directly with your system
     - People / things may play multiple roles simultaneously or over time
   - Use right level of abstraction

Poor actor name	Good actor name
Clerk	Pension Clerk
Third-level supervisor	Sale supervisor
Data Entry Clerk #165	Product accountant
Eddie "The Dawg" Taylor	Customer service representative

Use Case Development – Example: ATM System (4)

3. Identify use cases
   - Identify and refine actors' goals
     - Why would actor AAA use the system?
   - Identify actors' tasks to meet goals
     - What interactions would meet goal GGG of actor AAA?
   - Chose appropriate use case names
     - Verb-noun construction
     - No situation-specific data
     - Not tied to organization structure, paper forms, computer implementation, or manual process (e.g., enter form 104-B, complete approval window, get approval from immediate supervisor in Accounting Department)

Example actor: Customer

Goal: Access account 24/7 for regular banking operations (withdrawal, deposit, statement...) in a timely and secure way.

To be refined into sub-goals.

From goals we can identify tasks: Withdraw Cash, Make Deposit, Print Account Statement....

Use Case Development – Example: ATM System (5)

3. Identify use cases (cont'd)
   - Develop a brief description in narrative form
   - e.g., description of use case Withdraw Cash

The Customer identifies herself to the system, then selects the cash withdrawal operation. The system asks for the amount to withdraw. The Customer specifies the amount. The system provides the requested amount. The Customer takes the amount and leaves.

4. Identify preconditions
   - e.g., preconditions of use case Withdraw Cash
     - System is in operation, cash is available

Use Case Development – Example: ATM System (6)

5. Definition of primary scenario
   - Happy day story where everything goes fine
6. Definition of secondary scenarios (alternatives/exceptions)
   - A method for finding secondary scenarios is to go through the primary scenarios and ask:
     - Is there some other action that can be taken at this point? (alternate scenario)
     - Is there something that could go wrong at this point? (exception scenario)
     - Is there some behavior that could happen at any time? (alternative scenario unless it is an error, then it would be an exception scenario)

Example use case: Withdraw Cash

Not enough cash available

Incorrect identification

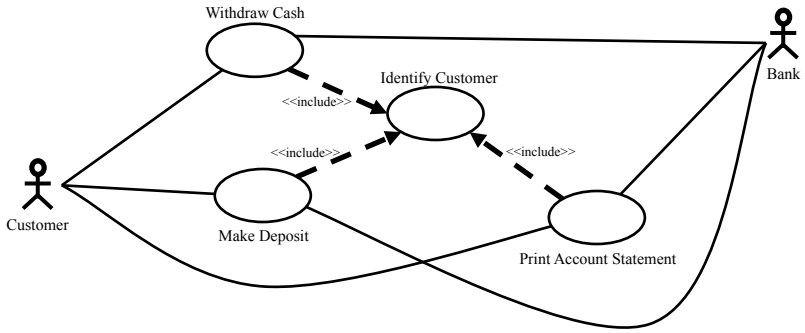
Customer forgets card in card reader

Incorrect amount entered

Customer forgets cash in dispenser

Use Case Development – Example: ATM System (7)

7. Structure use case diagram
   - Identify commonalities and specify included use cases
   - Identify variants and specify extended use cases



Use Cases Development (1)

Heuristics for finding use cases

- Select a narrow vertical slice of the system (i.e., one scenario)
  - Discuss it in detail with the user to understand the user's preferred style of interaction
  - Could target high value or high risk first
- Select a horizontal slice (i.e., many scenarios) to define the scope of the system
  - Discuss the scope with the user
- Use illustrative prototypes (e.g., mock-ups) as visual support
- Find out what the user does
  - Task observation (preferable to questionnaires)

Use Cases Development (2)

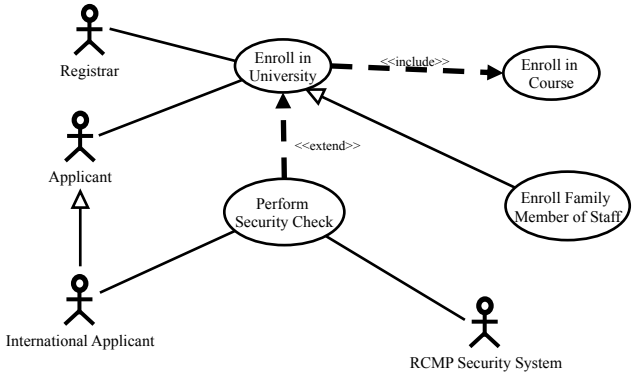
Alternative ways of identifying use cases (Ham, Larman)

- Identify external events to which the system must respond, and then relate these events to participating actors and specific use cases
- Express business processes in terms of specific scenarios, generalize the scenarios into use cases, and identify the actors involved in each use case
- Derive likely use cases from existing functional requirements – if some requirements do not trace to any use case, consider whether you really need them

Use Cases Development (3)

- Often one use case (or a very small number) can be identified as central to the system
  - The entire system can be built around this particular use case
- There are other reasons for focusing on particular use cases:
  - Some use cases will represent a high risk because for some reason their implementation is problematic
  - Some use cases will have high political or commercial value
- Approach is iterative
  - System scope and boundaries may change as more information is known about actors, their goals, and use cases

Example: University Registration System (1)



Example: University Registration System (2)

- Name: Enroll in University
- Identifier: UC 19
- Goal: Enroll applicant in the university
- Preconditions:
  - The Registrar is logged into the system.
  - The Applicant has already undergone initial checks to verify that they are eligible to enroll.
- Postconditions:
  - The Applicant will be enrolled in the university as a student if they are eligible.

Example: University Registration System (3)

Main Flow:

1. An applicant wants to enroll in the university.
2. The applicant hands a filled out copy of form UI13 University Application Form to the registrar.
3. The registrar visually inspects the forms.
4. The registrar determines that the forms have been filled out properly.
5. The registrar selects to Create New Student.
6. The system displays the Create Student Screen.
7. The registrar inputs the name, address, and phone number of the applicant.

[Extension Point: XPCheck]

8. The system determines that the applicant does not already exist within the system.
9. The system determines that the applicant is on the eligible applicants list.
10. The system adds the applicant to its records.
11. The registrar enrolls the student in courses via use case UC 17 Enroll in Course.
12. The system prepares a bill for the applicant enrollment fees.

Alternate Flows:

- 4.a The forms have not been adequately filled…

Example: University Registration System (4)

- Name: Perform Security Check
- Identifier: UC 34

At Extension Point XPCheck:

1. The registrar asks for security check results about applicant.
2. The system asks RCMP Security System for applicant security check results.
3. The RCMP Security System responds that applicant has been cleared.

3.a The Security System responds that applicant has not been cleared

Example: University Registration System (5)

- Name: Enroll Family Member of Staff
- Identifier: UC 20
- Inherits From: UC 19

Main Flow:

1. An applicant family member wants to enroll in the university.
2. The applicant hands a filled out copy of form UI15 University Application Form for Family Members to the registrar.

…

12. The system prepares a bill for the applicant enrollment fees based on staff family members rate.

Alternate Flows: …

Example: Point of Sale Application

Fully Dressed Example: Process Sale

Fully dressed use cases show more detail and are structured; they are useful in order to obtain a deep understanding of the goals, tasks, and requirements. In the NextGen POS case study, they would be created during one of the early requirements workshops in a collaboration of the system analyst, subject matter experts, and developers.

The usecases.org Format

Various format templates re available for fully dressed use cases. However, perhaps the most widely used and shared format is the template available at www.usecases.org. The following example illustrates this style.

Please note that this is the book's primary case study example of a detailed use case; it shows many common elements and issues.

Use Case UC1: Process Sale

Primary Actor: Cashier
Stakeholders and Interests:
- Cashier: Wants accurate, fast entry, and no payment errors, as cash drawer short ages are deducted from his/her salary.
- Salesperson: Wants sales commissions updated.
- Customer: Wants purchase and fast service with minimal effort. Wants proof of pur chase to support returns.
- Company: Wants to accurately record transactions and satisfy customer interests. Wants to ensure that Payment Authorization Service payment receivables are recorded. Wants some fault tolerance to allow sales capture even if server compo nents (e.g., remote credit validation) are unavailable. Wants automatic and fast update of accounting and inventory.
- Government Tax Agencies: Want to collect tax from every sale. May be multiple agen cies, such as national, state, and county.
- Payment Authorization Service: Wants to receive digital authorization requests in the correct format and protocol. Wants to accurately account for their payables to the store.

Preconditions: Cashier is identified and authenticated.
Success Guarantee (Postconditions): Sale is saved. Tax is correctly calculated. Accounting and Inventory are updated. Commissions recorded. Receipt is generated. Payment authorization approvals are recorded.

Main Success Scenario (or Basic Flow):
1. Customer arrives at POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.

Cashier repeats steps 3-4 until indicates done.

5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.
8. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory).
9. System presents receipt.
10. Customer leaves with receipt and goods (if any).

Extensions (or Alternative Flows):

*a. At any time, System fails:
To support recovery and correct accounting, ensure all transaction sensitive state and events can be recovered from any step of the scenario.
1. Cashier restarts System, logs in, and requests recovery of prior state.
2. System reconstructs prior state.
2a. System detects anomalies preventing recovery:
1. System signals error to the Cashier, records the error, and enters a clean state.
2. Cashier starts a new sale.

3a. Invalid identifier:
1. System signals error and rejects entry. 3b. There are multiple of same item category and tracking unique item identity not important (e.g., 5 packages of veggie-burgers):
1. Cashier can enter item category identifier and the quantity.

3-6a: Customer asks Cashier to remove an item from the purchase:
1. Cashier enters item identifier for removal from sale.
2. System displays updated running total.

3-6b. Customer tells Cashier to cancel sale:
1. Cashier cancels sale on System.

3-6c. Cashier suspends the sale:
1. System records sale so that it is available for retrieval on any POS terminal. 4a. The system generated item price is not wanted (e.g., Customer complained about something and is offered a lower price):
1. Cashier enters override price.
2. System presents new price.

5a. System detects failure to communicate with external tax calculation system service:
1. System restarts the service on the POS node, and continues. 1a. System detects that the service does not restart.
1. System signals error.
2. Cashier may manually calculate and enter the tax, or cancel the sale.

5b. Customer says they are eligible for a discount (e.g., employee, preferred customer):
1. Cashier signals discount request.
2. Cashier enters Customer identification.
3. System presents discount total, based on discount rules.

5c. Customer says they have credit in their account, to apply to the sale:
1. Cashier signals credit request.
2. Cashier enters Customer identification.
3. Systems applies credit up to price=0, and reduces remaining credit.

6a. Customer says they intended to pay by cash but don't have enough cash:
1a. Customer uses an alternate payment method.
1b. Customer tells Cashier to cancel sale. Cashier cancels sale on System.

7a. Paying by cash:
1. Cashier enters the cash amount tendered.
2. System presents the balance due, and releases the cash drawer.
3. Cashier deposits cash tendered and returns balance in cash to Customer.
4. System records the cash payment.

7b. Paying by credit:
1. Customer enters their credit account information.
2. System sends payment authorization request to an external Payment Authorization Service System, and requests payment approval.
2a. System detects failure to collaborate with external system:
1. System signals error to Cashier.
2. Cashier asks Customer for alternate payment.
3. System receives payment approval and signals approval to Cashier.
3a. System receives payment denial:
1. System signals denial to Cashier.
2. Cashier asks Customer for alternate payment.
4. System records the credit payment, which includes the payment approval.
5. System presents credit payment signature input mechanism.
6. Cashier asks Customer for a credit payment signature. Customer enters signature.

7c. Paying by check...
7d. Paying by debit...
7e. Customer presents coupons:
1. Before handling payment, Cashier records each coupon and System reduces price as appropriate. System records the used coupons for accounting reasons.
1a. Coupon entered is not for any purchased item:
1. System signals error to Cashier. 9a.

There are product rebates:
1. System presents the rebate forms and rebate receipts for each item with a rebate.

9b. Customer requests gift receipt (no prices visible): 1.
Cashier requests gift receipt and System presents it.

Special Requirements:
- Touch screen UI on a large flat panel monitor. Text must be visible from 1 meter.
- Credit authorization response within 30 seconds 90% of the time.
- Somehow, we want robust recovery when access to remote services such the inventory system is failing.
- Language internationalization on the text displayed.
- Pluggable business rules to be insertable at steps 3 and 7.

Technology and Data Variations List:
3a. Item identifier entered by bar code laser scanner (if bar code is present) or keyboard.
3b. Item identifier may be any UPC, EAN, JAN, or SKU coding scheme.
7a. Credit account information entered by card reader or keyboard.
7b. Credit payment signature captured on paper receipt. But within two years, we predict many customers will want digital signature capture.

Source: Craig Larman "Applying UML and Patterns"

Use Case Development – Rules of Thumb (1)

Be careful not to over specify behavior

- Keep it short, keep it simple: main flow should fit on a single page
- Focus on what, not how:
  - Focus on externally visible behavior
  - Are you specifying a sequence of events, in which the sequence does not really matter?
    - Example: Order of entering data for new customer
  - Do you specify which elements from a set are selected, when any arbitrary element is needed?
    - Example: Selecting new arbitrary phone number

Use Case Development – Rules of Thumb (2)

- Be careful not to under specify behavior
  - Do not forget variations on basic flow
  - Do not forget exceptions
    - For example, what happens to a phone call if there are no resources to allocate to it?
- Specify behavior for all possible inputs, both valid and invalid input

Use Case Development – Rules of Thumb (3)

- Keep names, data at an abstract level suitable for users
  - For example, input and output events should have intuitive names
  - Avoid data definition in use cases
- Use cases need to be understandable by users
  - They must validate them
- Avoid functional decomposition
  - Do not try to structure the use cases as nested functions with «includes»
  - Avoid deep hierarchies with «includes»

Use Case Development – Rules of Thumb (4)

- Think of use cases before use case diagram
- Do not spend too much time on the use case diagram – the textual description is the most important part
- Avoid too much use of "extends" and "includes" in use case diagrams
- Do not describe the user interface
- You do not want too many use cases; if you have too many, you have probably included too much detail ("If in doubt, leave it out")
  - Do not attempt to describe everything – too many variations – too many things that can go wrong
  - The requirements specification captures a more complete picture

Benefits of Use Case-Based Software Development

- They can help to define the scope of the system
- They are often used to plan the development process
- They are used to both develop and validate the requirements
  - Simple, easy to create
  - All stakeholders understand them
  - Often reflect user's essential requirements
  - Separates normal behavior from exceptional behavior
- They can form the basis for the definition of test cases
- They can be used to structure user manuals

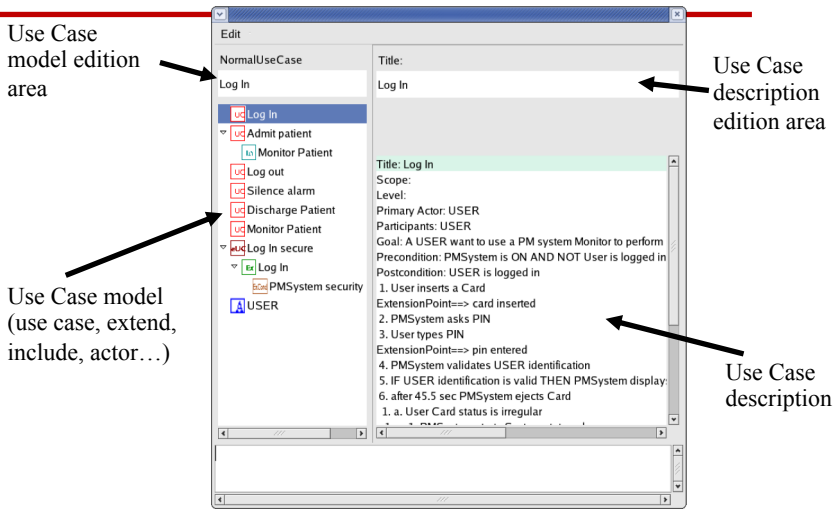
Use Cases are Not a Panacea…

- The use cases themselves must be validated
  - Using the requirements validation methods
  - Question/observe many types of users
- There are some aspects of software that are not covered by use case analysis
  - How to integrate nun-functional requirements?
- Innovative solutions may not be considered
- Scalability and maintainability
- Others discussed by Stephen Ferg in “What's Wrong with Use Cases”

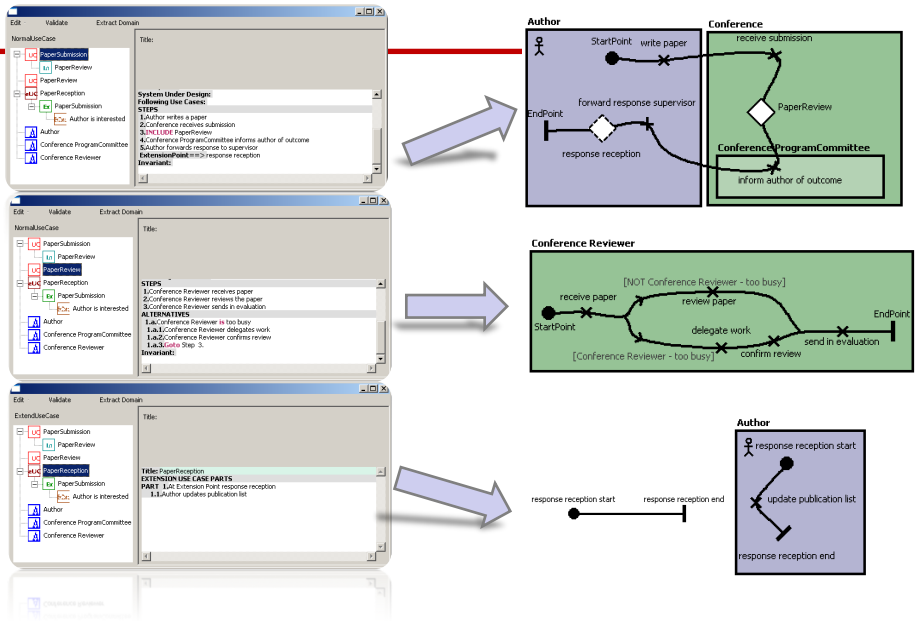
Tools

- Many UML tools support use case diagrams, without really supporting use cases well
- UCEd tool (Prof. Somé), to help capture/validate use cases
  - Use Case edition (structured English)
  - Domain model edition (and automatic extraction)
  - Scenario edition
  - Use Case & Domain model validation
  - Use Cases combination in state models
  - Simulation of executable model derived from Use Cases
  - Scenario generation
  - http://www.site.uottawa.ca/~ssome/Use_Case_Editor_UCEd.html

Use Case Edition with UCEd



UCEd Model Import in jUCMNav



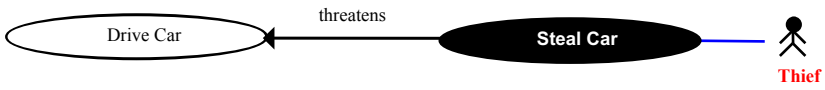
Don’t be Negative, But…

- Be ready to face the music!
  - In business, some people might wish to see you fail…
  - There are unforeseen events in any project
  - Open systems are subject to various threats
  - Software contains various kinds of bugs
- Remember Murphy’s Law…
  - “Anything that can go wrong will go wrong.”

Think about Negative Scenarios?

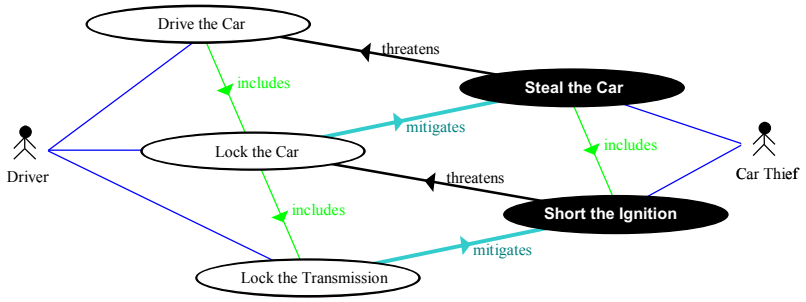
- A negative scenario is a scenario whose goal is
  - Undesirable from the business organization's viewpoint
  - Desirable from a hostile agent's viewpoint (not necessarily human)
- Consider them beforehand
  - As well as potential solutions
- Or…
  - Wait until it is too late to react…

Misuse Case



- Proposed by Sindre et Opdahl (2000)
  - Capture use cases that a system must be protected against
  - Goal is to threaten the system
  - Obvious applications for security and risk analysis
- The misuse case's misactor is a hostile agent
- The colors of the ellipse are inverted

A MiniMax[1] for Security



Use Cases for 'Car Security'

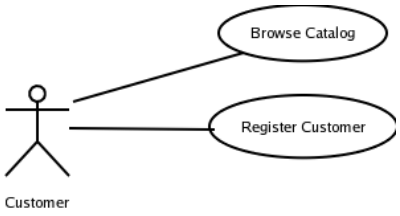
- Similar to a chess match…
  - White's best move is to find Black's best move and to counter it
- New relations:
  - threatens (from misuse case to use case)
  - mitigates (from use case to misuse case)

[1] Minimax: minimizing the maximum possible loss

Finding Misuse Cases – Step 1

- Start from a normal use case diagram
- Find misactors (hostile roles)
  - Who are the misactors, who want:
  - To harm the system, its stakeholders, or their resources intentionally

    or
  - To achieve goals that are incompatible with the system's goals



Competitor

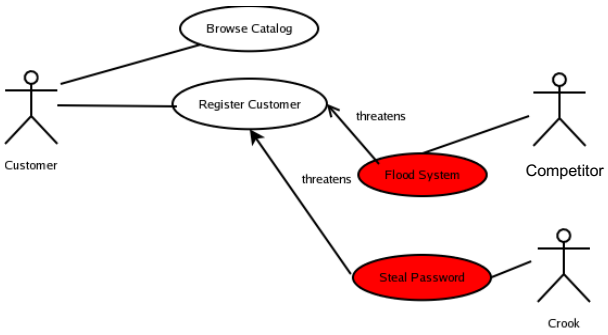
Crook

…

Finding Misuse Cases – Step 2

Find misuse cases

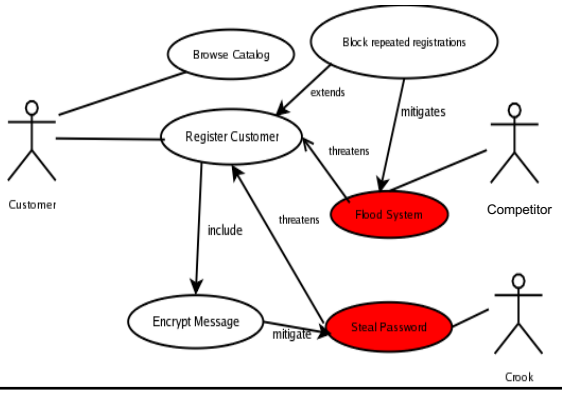
- Ask what would a misactor do to harm system
- Express goals of misactors (if needed elaborate with scenarios)
- Add relationships (threaten)



Finding Misuse Cases – Step 3

Mitigate misuse cases

- Ask what would neutralize the threats
- New included use case, new extension use case, or new secondary scenario to existing use case might be added



Benefits and Risks of Misuse Cases

- Benefits
  - Elicitation of security and safety requirements
  - Early identification of threats, mitigations, and exceptions that could cause system failure
  - Early identification of test cases
  - Documentation of rationales
- Risks
  - Get into premature design solutions in step 3 (mitigation)
    - Goal should be to find requirements (safety, security…)
  - Missing misactors and threats in a partial view

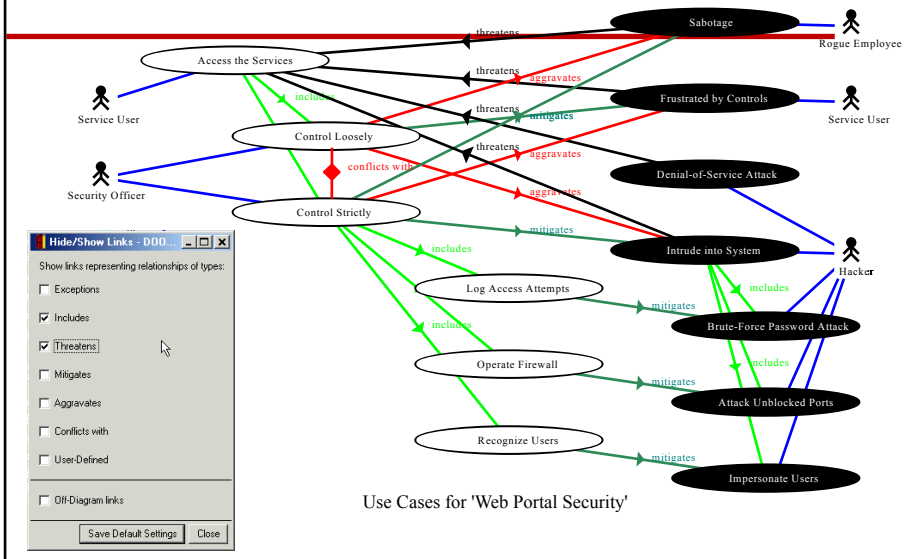
Tool: DOORS Plug-in

- Scenario Plus (for Telelogic DOORS)
- Textual / Graphical output (HTML)
- Automatic links, metrics, etc.
- Upon referencing: automatic creation of use/misuse cases

ID	Use Cases	Links to Included Use/Misuse Cases
UC-30	**2.1.3 Lock the Car**	
UC-31	**2.1.3.1 Primary Scenario**	
UC-35	System automatically Locks the Transmission to prevent the Car thief from Stealing the Car.	UC-49 Lock the Transmission UC-70 Steal the Car

- Automatic creation of links between misuse and use cases, by searching for underlined use case names with simple fuzzy matching

Conflict and Trade-off Analysis



Use Cases for 'Web Portal Security'