**Requirement**: a feature the system must have or a constraint it must satisfy to be accepted by the client

**Activities** (not necessarily in this order, repeat as needed):

1. **Requirements elicitation or gathering**: find out what clients **needs** (not just what he wants)
   - Ex. wants fast car, needs brakes
   - Rolling Stones: "you can't always get what you want, but sometimes you can get what you need"

2. **Requirements analysis**: develop models of current and proposed system

3. **Document requirements**: produce deliverable (Requirements Document)

4. **Testing**
   - Validation: Are we building the right product?
     - From client's point of view: Is the RD complete? Correct?
   - Verification: Are we building the product right?
     - From developer's point of view: Is the RD consistent? Non-ambiguous? Is the proposal feasible?
Main problem in requirements elicitation:
Communication
  • developers don't know client's environment and cannot communicate well with non-computer experts
  • client cannot express needs or does not know what he needs

Requirements elicitation techniques:

1. Interview client
   • **Unstructured** - ask open-ended questions about problem to be solved and work activities to be modeled
   • **Structured** - ask focused questions - plan q in advance - not normally good for initial interview - better for clarification
   • Developers should write-up interview for review by client - client may add information and correct errors

2. Questionnaires
   • Only use when too many/diverse users to interview
   • Disadvantage: not interactive, hard to ask good questions, answers may not be meaningful
3. Examine client's forms and documentation:
   • examine existing forms used by client to find out how he does business
   • examine client's manuals on operating procedures

4. Observation of client's operations
   • how users do tasks currently
   • Video or audio recording optional
   • Act of observation may influence behavior of user (Hawthorne effect: named after 1939 study of workers in Hawthorne, Ill.)
   • Informal or use HCI techniques:
     • Field studies
     • Controlled lab studies

5. rapid "throw-away" prototyping
   • system built as quickly as possible
   • only user-visible features implemented (Wizard-of-Oz)
   • must be able to modify user-visible features quickly
   • informal or HCI techniques to observe users with it
   • must throw it away after it has served its purpose
Object-oriented Requirements Analysis

1. **Identify scenarios**: description of using system from a specific users' point of view to achieve a specific goal

2. **Identify Use Cases**: generalization of 1 or more scenarios, to describe all possible scenarios compactly

**Scenario**
- like a story: tells how a *single* actor achieved a specific goal
- from point of view of a single "actor", but other supporting actors may be involved in story
  - "actor": an external entity that interacts with system
  - can be human or another system (e.g. Internet)
  - a user can play role of different actors
- Purpose:
  - help client and developer communicate
  - to help construct use cases (next step)
Example: 3 informal scenarios for on-line mail-order catalog company (such as L.L. Bean):

- Customer Nancy wants to buy hiking boots: she finds the boots she wants in the catalog, places her order, gives her credit card number and shipping address
  - **Primary scenario** (also called normal or Happy Day scenario)
  - everything goes right
  - most likely interaction
  - always at least one primary scenario

- What if Nancy gives incorrect credit card number?
- What if hiking boots Nancy ordered are not in stock?
  - These are called **Secondary** (Abnormal) scenarios
  - describe error conditions (incorrect credit card) or less common alternatives (out of stock)

**How to develop formal scenarios:**

1. **Identify all actors** - can use questions
   - Who will use the system to perform his job? (ex. warehouse person who will fill order)
   - Who will use the system's main functions? (ex. shopper)
• Who will use secondary features of system? (ex. accounting dept. will use data)
• What external hardware or software will interact with it? (ex. boot supplier's system, credit card company's system)

2. **Identify all scenarios:** can use questions to help
   • What are all the tasks that each actor wants the system to perform? (ex. customer wants to order items, wants to see what styles of boots, etc.)
   • What information does the actor access? (ex. customer looks at on-line catalogue)
   • Who creates that information? (ex. sales dept. adds items to the catalogue)
3. **Write up primary and secondary scenarios**
   - Use application domain's terms
   - Specify name of scenario (underlined), participating actors (names underlined), then flow of events:
     - steps describing what happens from actor's point of view
     - **no branching steps or alternatives** allowed
   - To Identify secondary scenarios: Go through primary scenario line by line and ask:
     - Are there alternative actions that could be done here?
     - Something that could go wrong at this point?
     - Is there something that could go wrong at any time?

4. **Validate scenarios with client**
   - Are they complete? - do they describe all possible cases, including exceptions and errors
   - Are they correct from client's view of system?

Developing scenario is first step towards developing **use cases**
Scenario Name: BuyBoots  
Participating actor: Nancy: Shopper, Dilbert: PhoneOrderTaker  
Flow of events:
1. Nancy wants to order a pair of size L hiking boots that she sees in a printed catalog and so she calls the toll free phone number for ordering.
2. Dilbert answers the phone and asks her what she wants to order.
4. Dilbert enters the item information into the system and the system displays the information that the item is available.
5. Dilbert tells Nancy that the item is available and asks her for her credit card.
6. Nancy tells him that it is PLASTIC999999999.
7. Dilbert enters the credit card number into the system and the system displays the information that Nancy’s credit is good and that her address is 123 Nice Place, Any City, NC.
8. Dilbert asks Nancy if she wants the order shipped to 123 Nice Place, Any City, NC.
10. Dilbert tells the system to submit the order and the system confirms that the order has been placed.
11. Dilbert tells her that the order has been placed and she should get the boots in 7 days.

Scenario Name: InvalidCreditCard  
Participating actor: Nancy: Shopper, Dilbert: PhoneOrderTaker  
Flow of events:
1 – 6 is same as in BuyBoots scenario.
7. Dilbert enters the credit card number into the system and the system displays the information that Nancy’s credit card is no longer valid.
8. Dilbert informs Nancy and asks if she wants to use another credit card.
9. She says no thank you and hangs up.
10. Dilbert enters the information into the system that the customer did not wish to make the order with another credit card.

Scenario Name: ItemOutOfStock  
Participating actor: Nancy: Shopper, Dilbert: PhoneOrderTaker  
Flow of events:
1 – 3 is same as in BuyBoots scenario.
4. Dilbert enters the item information into the system and the system displays the information that the item is NOT available.
5. Dilbert tells Nancy and asks her if she wants to order another item.
6. Nancy says no thank you and hangs up.
7. Dilbert enters the information into the system that the customer did not wish to make another order.
Example of a Primary Scenario
Fall 2001 - CSC640 - Written by Dr. Green

Dr. Lea needs information for her research on software architectures for operating system design. She has saved a collection of articles in html and ascii that she has downloaded into two directories. Some of the articles may contain information that she can use. However, she has no idea which of the saved articles to read, or which directory they are in, so she starts the Inquiry program:

1. Dr. Lea gives the name of one of the directories: C:\MyDocuments\papers\Spring00.
2. Inquiry confirms that it can access that folder and that it contains 20 files.
3. Dr. Lea then describes the papers of possible interest to her:
   a. Highest (1) priority:
      • Contains exact phrase "operating system" and either 1. the keyword 'architecture' or 2. the keyword 'design'
   b. Next (2) priority:
      • contains at least one of the keywords: 'client', 'server', 'pipe', 'filter', 'repository'
4. Inquiry reports a syntax error in the information given in 3a.
5. Dr. Lea discovers that she left out a quotation mark and retypes the search criteria.
6. Inquiry confirms that the search criteria were entered correctly.
7. Dr. Lea then specifies some desired global characteristics of the search criteria to Inquiry:
   a. Perform case insensitive matching
   b. Use any synonyms of the keywords (which she defined in a previous session) as additional search criteria
   c. Include plural forms of the keywords in the search (e.g. system/systems)
8. Inquiry confirms that it can do that
9. Dr. Lea requests Inquiry to list results in this format: for each file name, the number of matches at each priority (1 and 2); also, the files should be sorted in descending order by total number of matches.
10. Inquiry performs the search and lists the results:
    | File name | number of priority 1 matches | number of priority 2 matches |
    |-----------|-----------------------------|-----------------------------|
    | Mmm.txt   | 20                          | 50                          |
    | AAA.txt   | 30                          | 10                          |
    | CCC.txt   | 2                           | 20                          |
11. Dr. Lea requests that Inquiry show her the context in which each of the matches occurred in file AAA.txt
12. Inquiry shows her this:
    Context of matches found in AAA.txt:
    … design of operating system software
    … architectures for distributed operating systems
    … client-server architecture
13. Dr. Lea decides to print AAA.txt so she prepares to exit Inquiry
14. Inquiry asks whether to save the search results for later use.
15. Dr. Lea says yes, in case AAA.txt turns out not to be what she was looking for.
16. Inquiry saves them before ending.
Use cases

- Purpose: to compactly describe all possible scenarios (too many scenarios, possibly infinite)
- Generalization of multiple scenarios (but still from user's point of view)
- Scenario is instance of a Use Case
- Scenarios-Use Cases similar to Object-Class distinction

How to develop use cases:

1. Conceptually group related scenarios into one use case - example:
   - Primary scenario: Nancy orders hiking boots at regular price
   - Alternative scenario: Sue orders gloves that are on-sale

2. Describe use case
   - Name
   - Brief Description
   - Preconditions: what must be true before use case can start
   - Main flow of events:
     - not necessarily sequence of steps
     - can use if-then, while loops, for loops
• **Alternative flow**: steps outside of the main flow, e.g.
  "At any time before selecting SUBMIT, the customer can select CANCEL"
• **Postconditions**: what must be true after the use case ends

**Example of Use Case**

**Name:** Place order  
**Brief Description:** The customer orders items from the printed copy of the catalog and charges the items to his credit card.  
**Actors:** customer, credit card company  
**Preconditions:** items ordered are in catalog and in stock  
**Main flow of events:**  
1. For each item that the customer wants, he enters the item's catalog number, quantity, and other required information (such as size, color, etc.).  
2. The system confirms that the items have valid catalog numbers and are in stock in the requested quantity, size, color, etc.  
   a. If the item is on sale, then the system adds the sale price to the bill  
   b. Otherwise the system adds the regular price to the bill  
3. The customer gives his credit card number  
4. The system contacts the credit card company to confirm that the credit card number is valid  
5. The credit card company confirms the number is valid  
6. The system charges the sale to the credit card  
7. The system informs user that order has been placed
**Alternative flow:** In step 5, if the credit card company informs the system that the card number is not valid, the system informs the customer and requests another number. If the customer enters another credit card number then flow resumes at step 4; otherwise the system terminates the order.

**Postconditions:** The order has been placed and the credit card has been charged.

3. **Refine use cases**
   - Goal: reduce complexity and increase understandability of use cases
   - Divide complex use cases into separate cases related by *include and extends relations* (to be discussed shortly)
   - Rewrite the use case descriptions

4. **Draw UML Use Case Diagram:**
   - shows relationships among all the use cases and actors
   - links show which actors interact with use case

5. **Validate use case model with user:**
   - Complete?
   - Correct?