OBJECT-ORIENTED AND CLASSICAL SOFTWARE ENGINEERING

EIGHTH EDITION, WCB/MCGRAW-HILL, 2011

STEPHEN R. SCHACH

Edited by Hyungbae Park **University of Central Missouri**

CHAPTER 11

Requirements

Overview

- Determining what the client needs
- Overview of the requirements workflow
- Understanding the domain
- The business model
- Initial requirements
- Initial understanding of the domain: The MSG Foundation case study
- Initial business model: The MSG Foundation case study

Overview (contd)

- Initial requirements: The MSG Foundation case study
- Continuing the requirements workflow: The MSG Foundation case study
- Revising the requirements: The MSG Foundation case study
- The test workflow: The MSG Foundation case study
- The classical requirements phase
- Rapid prototyping

Overview (contd)

- Human factors
- Reusing the rapid prototype
- CASE tools for the requirements workflow
- Metrics for the requirements workflow
- Challenges of the requirements workflow

The Aim of the Requirements Workflow

- To answer the question:

What must the product be able to do?

11.1 Determining What the Client Needs

- Misconception
  - We must determine what the client *wants*
- The real objective of the requirements workflow is
  - To determine what the client *needs*
  - Problem
    - Many client do not know what they need
    - Even a client who has a good idea of what is needed may have difficulty in accurately conveying these ideas to developers
      - Most clients are less computer literate than the members of the development team

Determining What the Client Needs (contd)

- Determining what the client needs is straightforward
  - The members of the development team simply ask him or her
- However, this direct approach usually does not work very well
  - The client may not appreciate what is going on in his or her own organization
  - Software is complex (major reason)
    - A client often asks for the wrong software product
    - It is difficult enough for a software engineer to visualize a software product and its functionality
    - The problem is far worse for the client

Determining What the Client Needs (contd)

- A skilled systems analyst is needed to elicit the appropriate information from the client
- The client is the only source of this information

Determining What the Client Needs (contd)

- The solution:
  - Obtain initial information from the client
  - Use this initial information as input to the Unified Process
  - Follow the steps of the Unified Process to determine the client's real needs

    (Section 11.12)

11.2 Overview of the Requirements Workflow

- First, gain an understanding of the *application domain* (or *domain*, for short)
  - The specific environment in which the target product is to operate
  - e.g. banking, space exploration, automobile manufacturing
- Second, build a business model
  - Use UML diagrams
  - Model the client's business processes
  - Deeper understanding of the domain is gained
- Third, use the business model to determine the client's requirements
- Iterate the above steps
  - Until the team is satisfied with the set of requirements

Definitions

- Discovering the client's requirements
  - Re*quirements elicitation* (or *requirements capture*)
  - Methods include interviews and surveys
- Refining and extending the initial requirements
  - *Requirements analysis*

11.3 Understanding the Domain

- Every member of the development team must become fully familiar with the application domain
  - Correct terminology is essential
  - Use of an inappropriate word may lead to a misunderstanding, eventually resulting in a faulty product being delivered
- Construct a glossary
  - A list of technical words used in the domain, and their meanings
  - Updated whenever the members of the requirements team encounter new terminology

11.4 Business Model

- A *business model* is a description of the business processes of an organization
  - For example, business processes of a bank – include accepting deposits from clients, loaning money to clients, and making investment

- Why building business model?
  - The business model gives an understanding of the client's business as a whole
  - This knowledge is essential for advising the client regarding computerization

11.4 Business Model

- The systems analyst needs to obtain a detailed understanding of the various business processes
  - Different techniques are used, primarily interviewing

11.4.1 Interviewing

- The requirements team meet with the client and users to extract all relevant information

- There are two types of questions
  - *Close-ended* questions require a specific answer
    - e.g. How many salespeople the company employs?
    - e.g. How fast a response time is required?
  - *Open-ended* questions are posed to encourage the person being interviewed to speak out
    - e.g. Why is your current software product unsatisfactory?

Interviewing (contd)

- There are two types of interviews
  - In a *structured* interview, specific preplanned questions are asked, frequently close-ended
  - In an *unstructured* interview, questions are posed in response to the answers received, frequently open-ended
    - Interviewer may start with one or two prepared closed-ended question
    - Subsequent questions are likely to be open ended in nature to provide the interviewer with wide-ranging information
- Not a good idea if the interview is too unstructured
  - "Tell me about your business"

Interviewing (contd)

- Interviewing is not easy
  - The interviewer must be fully familiar with the application domain
  - The interviewer must remain open-minded at all times
- After the interview, the interviewer must prepare a written report
  - It is strongly advisable to give a copy of the report to the person who was interviewed

11.4.2 Other Techniques

- Interviewing is the primary technique

- Other techniques that may be used in conjunction with interviewing
  - Send a questionnaire to the relevant members of the client organization
    - Useful when the opinions of hundreds of individuals need to be determined
    - Carefully written answer from an employee of the client organization may be more accurate than an immediate verbal response

Other Techniques (contd)

- Examination the various forms used by the business
  - Documents such as operating procedures and job description
  - Software product – user manuals
- Direct observation of the employees while they perform their duties can be useful
  - Videotape cameras are a modern version of this technique
  - But, it can take a long time to analyze the tapes
  - Employees may view the cameras as an unwarranted invasion of privacy

11.4.3 Use Cases

- A model is a set of UML (Unified Modeling Language) diagrams
  - Represent one or more aspects of the software product to be developed
  - Primary UML diagram used in business modeling is the use case

Use Cases (contd)

- Use case diagram
  - Describes the externally observable behavior of system functions
  - Describes interactions between the software product itself (system) and the users of that software product (actors)
  - Example:

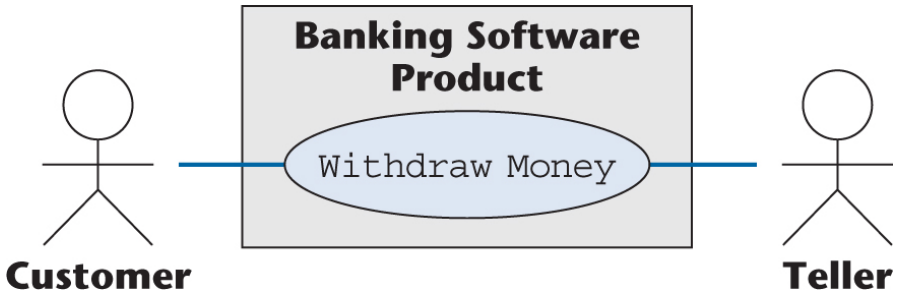
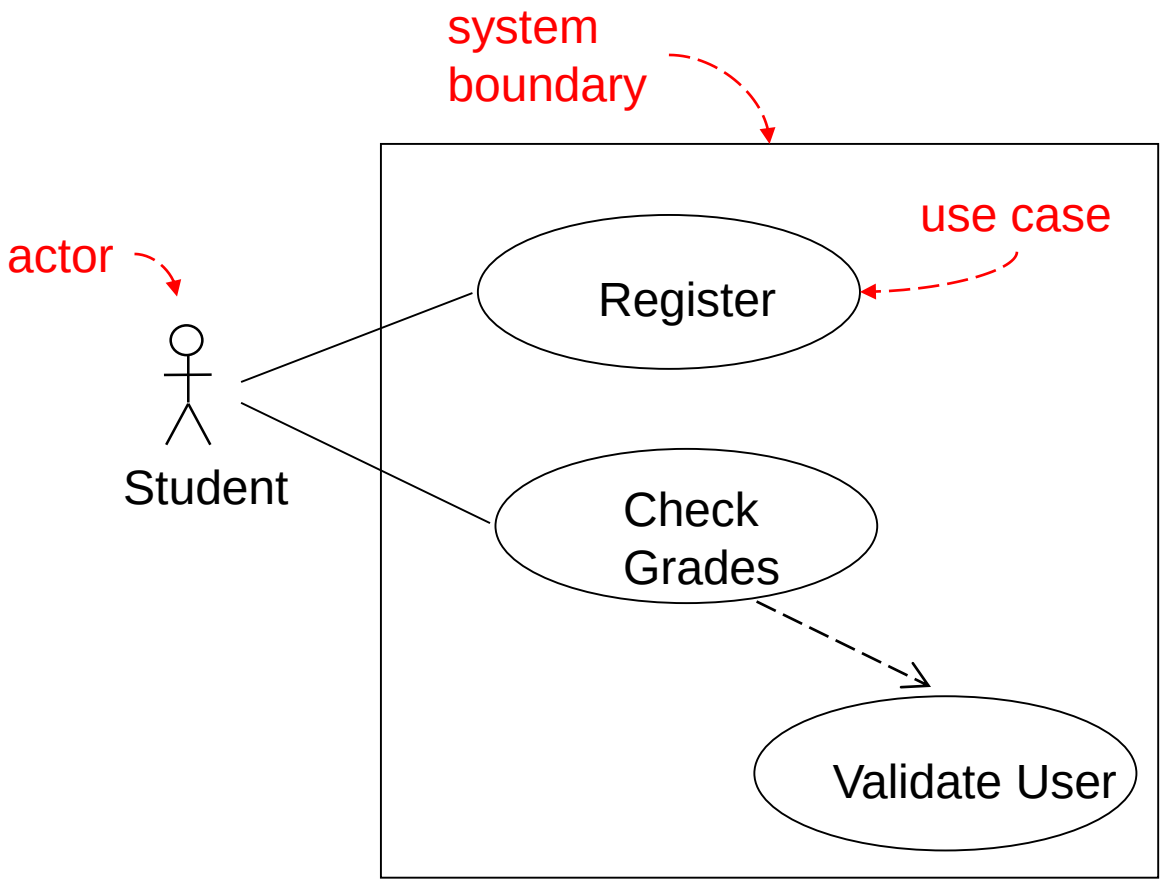


Figure 11.1

Example


system
boundary
use case
actor
Register
Student
Check
Grades
Validate User

Elements of Use Case Diagram

- Actor:
  - represents a role played by external entities that interact with the system
- Use case:
  - Describes what the system does (i.e., functionality)
  - Scenario: sequence of interactions between the actors and the system
- Relationship:
  - Extension (or generalization) among actors
  - Association between actors and use cases

Use Cases (contd)

- Example:

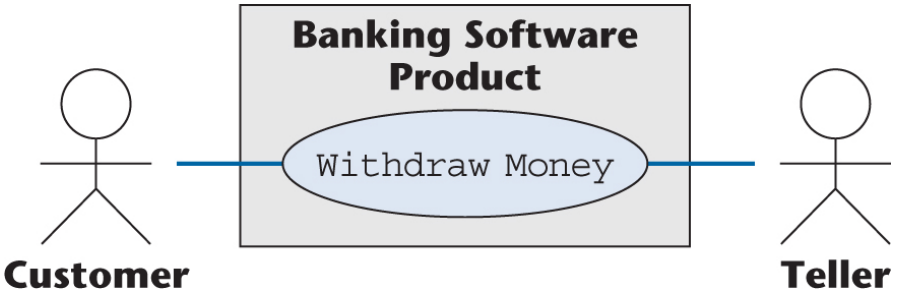


Figure 11.1

- Two actors – Customer, Teller
- Business activity (use case) – `Withdraw Money`

Use Cases (contd)

- An *actor* is a member of the world outside the software product
  - An actor is frequently *a user of the software product*

- In general, an actor plays a role with regard to the software product. This role is
  - As a user; or
  - As an initiator; or
  - As someone who plays a critical part in the use case

Use Cases (contd)

- A user of the system can play more than one role

- Example: A customer of the bank can be
  - A **Borrower** (when he or she takes out a loan) or
  - A **Lender** (when he or she deposits money in the bank)

Use Cases (contd)

- Conversely, one actor can be a participant in multiple use cases

- Example: A **Borrower** may be an actor in
  - The `Borrow Money` use case;
  - The `Pay Interest on Loan` use case; and
  - The `Repay Loan Principal` use case

- Also, the actor **Borrower** may stand for many thousands of bank customers

Use Cases (contd)

- An actor need not be a human being

- Example: An e-commerce information system has to interact with the credit card company information system
  - The credit card company information system is an actor from the viewpoint of the e-commerce information system
  - The e-commerce information system is an actor from the viewpoint of the credit card company information system

Use Cases (contd)

- A potential problem when identifying actors
  - Overlapping actors

- Example: Hospital software product
  - One use case has actor **Nurse**
  - A different use case has actor **Medical Staff**
  - Better:
    - Actors: **Physician** and **Nurse**

Use Cases (contd)

- Alternatively:
  - Actor **Medical Staff** with two specializations: **Physician** and **Nurse**

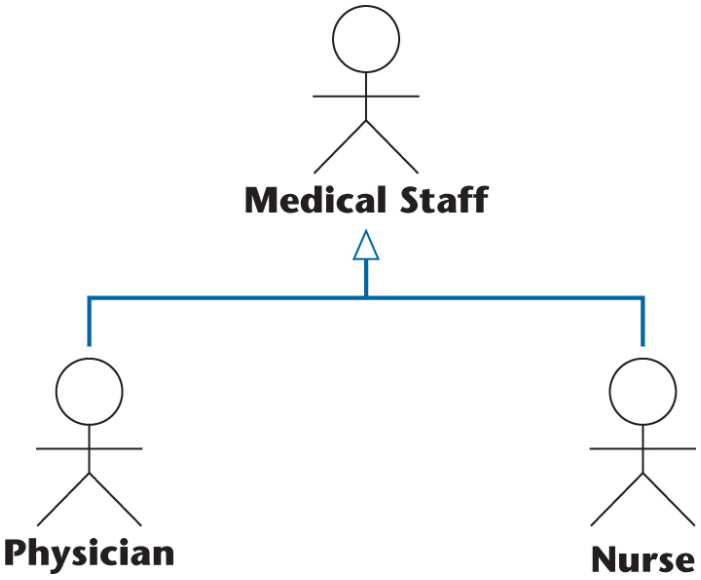


Figure 11.2

11.5 Initial Requirements

- To determine the client's requirements,
  - The initial requirements are drawn up based on the initial business model
- Then requirements are refined
- The requirements are *dynamic* — there are frequent changes
- A good way to handle frequent changes
  - Maintain a list of likely requirements,
  - Together with use cases of requirements approved by the client

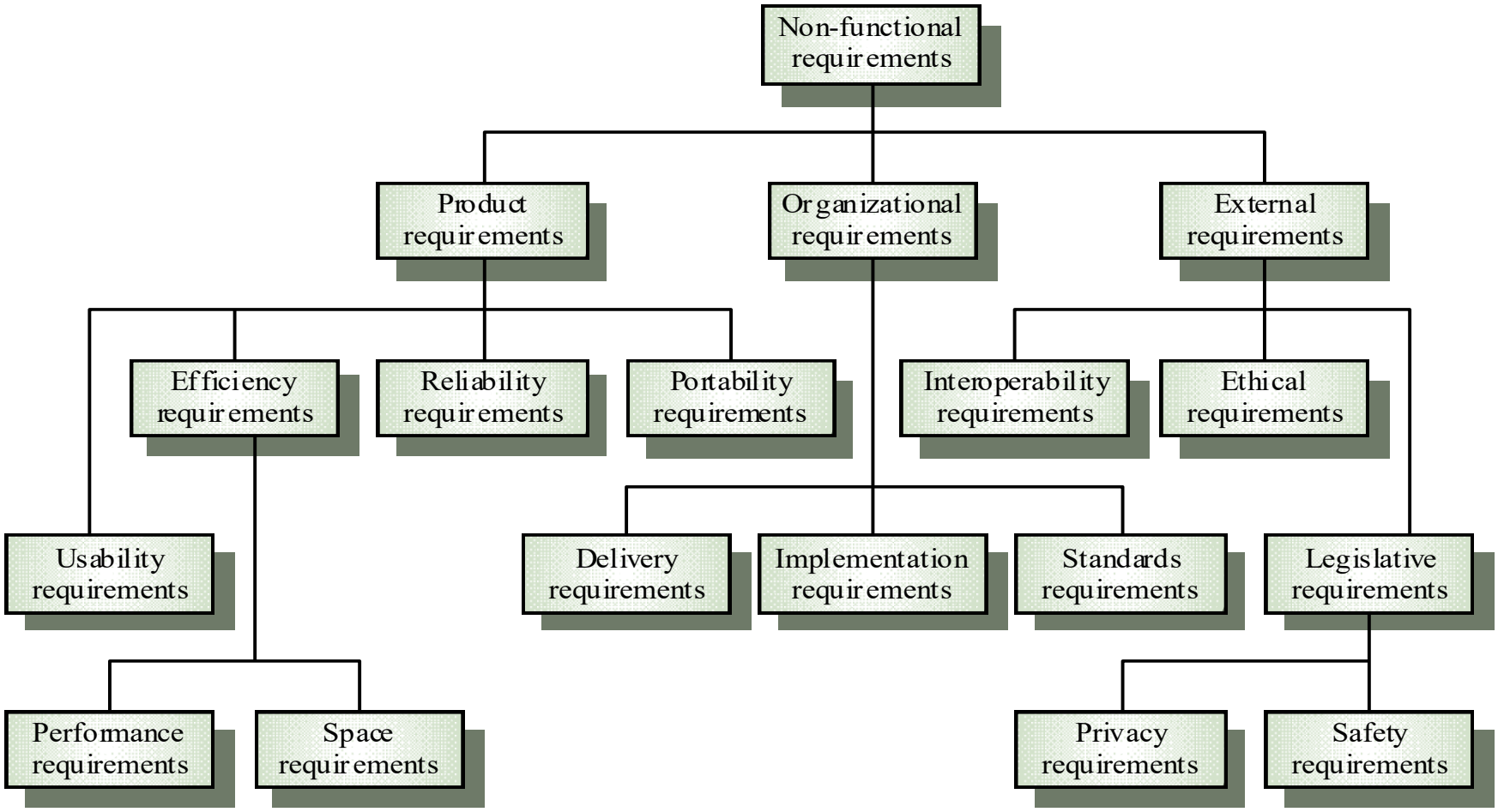
Initial Requirements (contd)

- There are two categories of requirements
  - A *functional requirement* specifies an action that the software product must be able to perform
    - Often expressed in terms of inputs and outputs
  - A *nonfunctional requirement* specifies properties of the software product itself, such as
    - Platform constraints
    - Response times
    - Reliability

Examples of Functional Requirements

- The user shall be able to search either all of the initial set of databases or select a subset from it.
- The system shall provide appropriate viewers for the user to read documents in the document store.
- Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storage area.

Non-functional Requirement Types



Non-functional Requirements Examples

- Product requirement
  - It shall be possible for all necessary communication between the APSE and the user to be expressed in the standard Ada character set
- Organizational requirement
  - The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95
- External requirement
  - The system *shall not disclose any personal information* about customers apart from their name and reference number to the operators of the system

Non-functional Requirements Measures

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K Bytes Number of RAM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Initial Requirements (contd)

- Functional requirements are handled as part of the requirements and analysis workflows
- Some nonfunctional requirements have to wait until the design workflow
  - The detailed information for some nonfunctional requirements is not available until the requirements and analysis workflows have been completed

11.13 Rapid Prototyping

- Hastily built ("rapid") software
  - Exhibits only key functionality of the target product
  - Imperfections can be ignored

- Example: Apartment complex management product
  - Must
    - Input screen that allows the user to enter details of a new tenant
    - Print an occupancy report for each month
  - Omit 'hidden' aspects
    - Error-checking capabilities, file-updating routines, and complex tax computations

Rapid Prototyping (contd)

- Rapid prototyping model – "*Rapid*"
  - Build the rapid prototype as quickly as possible

- Aim:
  - To provide the client with an understanding of the product
  - To enable the client and the developers to agree as quickly as possible on what the product is to do

Rapid Prototyping (contd)

- A rapid prototype is built for change
  - If the first version of the rapid prototype is not what the client needs,
  - Then the prototype must be transformed rapidly into a second version
  - Languages for rapid prototyping include 4GLs and interpreted languages
    - Smalltalk, Prolog, and Lisp, include HTML and Perl
    - Language is probably a good candidate, if the answer to both questions is *Yes*
      - Can a given language be used to produce a rapid prototype?
      - Can the rapid prototype be changed quickly?

11.14 Human Factors

- The client and *the future users of the product* must interact with the user interface

- User-friendliness
  - Vital objective for all software products
  - Ease with which human beings can communicate with the software product
  - Human-computer interface (HCI)
    - Menu, not command line
    - "Point and click"
    - Windows, icons, pull-down menus

Human Factors (contd)

- Human factors must be taken into account
  - Avoid a lengthy sequence of menus
  - Allow the expertise level of an interface to be modified
  - Uniformity of appearance is important
    - Users intuitively knowing how to use a screen that they have never seen before
  - Advanced psychology vs. common sense?
  - Benefits: reduced learning times and lower error rates
- Rapid prototype of the HCI of every product is obligatory

11.15 Reusing the Rapid Prototype

- After the rapid prototype has been built, it is discarded early in the software process
- Reusing a rapid prototype is essentially code-and-fix

- Problems
  - Changes are made to a working product
    - Expensive
  - Maintenance is hard without specification and design documents
  - Real-time constraints are hard to meet

Reusing the Rapid Prototype (contd)

- One way to ensure that the rapid prototype is discarded
  - Implement it in a different language from that of the target product
  - Example
    - Rapid prototype is implemented in HTML
    - Client may specify that the product must be implemented in Java

Reusing the Rapid Prototype (contd)

- Refine a rapid prototype, more specifically, portions of the rapid prototype
  - Portions may be used in the final product
  - Generated code can be reused

- We can safely retain (parts of) a rapid prototype if
  - This is prearranged
  - Those parts pass SQA inspections
  - However, this is not “classical” rapid prototyping

11.16 CASE Tools for the Requirements Workflow

- We need graphical tools for UML diagrams
  - To make it easy to change UML diagrams
  - The documentation is stored in the tool and therefore is always available
- Such tools are sometimes hard to use
- Overall, the strengths outweigh the weaknesses

CASE Tools for the Requirements Workflow (contd)

- Graphical CASE environments extended to support UML diagrams include
  - System Architect
  - Software through Pictures
- Object-oriented CASE environments include
  - IBM Rational Rose
  - Together
  - ArgoUML (open source)

11.17 Metrics for the Requirements Workflow

- Volatility and speed of convergence are measures of how rapidly the client's needs are determined
  - Useful metric – measure of requirements volatility
  - The number of requirements that change during the rest of the software development process
    - Requirements workflow should be thoroughly reviewed,

      if a large number of changes in requirements are initiated during the analysis, design workflows

Metrics for the Requirements Workflow (contd)

- Changes initiated by the client repetitively
  - Metric can be used to warn the client
    - Moving target problem can adversely affect the project

11.18 Challenges of the Requirements Phase

1. Essential to have the wholehearted cooperation of the potential users of the target product from the beginning of the process
   - Employees of the client organization often feel threatened by computerization
     - Members of the client organization with whom they interact in all probability are deeply concerned about the potential impact of the target software product on their jobs
2. The requirements team members must be able to negotiate
   - The client's needs may have to be scaled down

Challenges of the Requirements Phase (contd)

3. Key employees of the client organization may not have the time for essential in-depth discussions

4. Flexibility and objectivity are essential
   - Making premature assumptions regarding the requirements is dangerous;
   - Making any assumptions during the requirements workflow regarding the software product to be built can be disastrous

The steps of the requirements workflow

- The chapter 11 concludes with Box 11.1

How to Perform the Requirements Workflow Box 11.1

- **Iterate**

  Obtain an understanding of the domain.

  Draw up the business model.

  Draw up the requirements.

- **Until** the requirements are satisfactory.