Requirements Analysis Concepts & Principles

Instructor: Dr. Jerry Gao
Requirements Analysis Concepts and Principles

- Requirements Analysis
- Communication Techniques
  - Initiating the Process
  - Facilitated Application Specification Techniques
- Analysis Principles
  - Information Domain
  - Modeling
  - Partitioning
- Software Prototyping
  - Selecting the Prototyping Approach
  - Prototyping Methods and Tools
- The Software Requirements Specification
  - Specification Principles
  - Representation
Requirements Analysis

Requirements analysis
-> A process of discovery, refinement, modeling, and specification.
During the process, both the developers and customers take an active role.

Focus on: “what” instead of “how”

Input of the requirements analysis process:
- Software Project Plan
- System specification (if one exists)

Output: Software requirements specification document
- provides the software engineer with models that can be translated in to data, architectural, interface, and procedure design.
- customer and developer can check the quality of the software and provide the feedback.

Who perform requirements analysis: system analysts
Requirements Analysis

Major tasks and efforts:
- Problem recognition (or system understanding)
  - Discover and understand the system requirements
  - Refine the requirements
- Evaluation and synthesis:
  - what are the alternative solutions
  - focus on what solution should be selected or used instead of how to implement a solution.
- Modeling: to represent the various aspects of the system
  - the required data
  - information and control flow
  - operation behavior
- Specification of:
  - software functions, and performance
  - interfaces between system elements
  - system constraints
Requirements Engineering Process

- Feasibility study:
  - Identify and estimate to see if user needs can be satisfied using current techniques and technologies.

- Requirements analysis:
  - The process of deriving the system requirements through observation of existing systems, discussion with users and customers, task analysis, and so on.

Requirements definition:
  - Translating the information into a REQ. document.

Requirements specification:
  - Define system requirements using a consistent precise, and complete way.
  - Using some requirements specification method
Requirements Engineering Process

Feasibility Study

Requirements analysis

Requirements definition

Requirements specification

System models

Definition of requirements

Requirements documents

Specification of requirements
Requirements Analysis Process

- Domain understanding:
  - Understanding of application domain.

- Requirements collections:
  - The process of interacting with customers, users to discover the requirements for the system.

- Requirements classification:
  - Group and classify the gathered requirements.

- Conflict resolution:
  - Resolve the conflict requirements.

- Prioritization:
  - Identify and list requirements according to their importance.

- Requirements validation:
  - Check and validate the gathered requirements to see if they are complete, correct, and sound.
Requirements Analysis Process

- Requirements collection
- Domain understanding
- Classification
- Prioritization
- Conflict resolution

Requirements validation

Requirements definition and specification
Communication Techniques

Initiating the Process:

Q1 set: Context free questions to lead the basic understanding of the problem

Who is behind the solution?
Who will use the solution?
…..

Q2 set: Questions to gain a better understanding of the problem and the customer’s perceptions about a solution.

How would you characterize “good” output that would be generated by a successful solution?

What problems will this solution address?

Q3 set: Meta-questions focus on the effectiveness of the meeting.

Are you the right person to answer these questions? Are your answers “official”? 
Customers and software engineers often have an unconscious “us and them” mind set. This may cause: misunderstandings, miss important information, …. To solve the problem, FAST approach is proposed.

FAST encourages the creation of a joint team of customers and developers. They work together
- to identify the problem and proposed and
- to negotiate the different elements of solutions and approaches

The basic guidelines of FAST:
- hold a meeting at a neutral site
- establish rules for preparation and participation
- have a formal meeting agenda
- control the meeting by a “facilitator”
- use a “definition mechanisms”
- have a common goal to
  - identify the problem
  - propose elements of solutions and requirements
  - negotiate different approaches
**Quality Function Deployment**

*Quality function deployment (QFD) is a quality management technique*

- Translate the needs of the customer into technical requirements for software.

QFD identifies three types of requirements:

- **Normal requirements:**

  **Objectives and goals:**

  examples: types of graphic displays, specific system functions

- **Expected requirements:**

  implicit requirements:

  examples: ease of human-machine interaction

  ease of software installation

- **Exciting requirements:**

  Features go beyond the customer’s expectations
Analysis Principles

Each analysis method has a unique point of view.

All analysis methods are related by a set of operational principles:

- represent and understand the information domain
- define the functions that the software
- represent the behavior of the software
- use models to depict information, function, and behavior
  --> uncover the details in a layered fashion.
- move from essential information toward to details

A set of guidelines for requirement engineering:

- understand the problem before beginning to create the analysis model
- develop prototypes to help user to understand how human-machine interactions
- record the origin of and the reasons for every requirement
- use multiple views of requirements
- prioritize requirements
- work to eliminate ambiguity
The Information Domain

Software is built to process data, to transform data from one form to another.

Software also process events.

The first operational analysis principle requires to exam the information domain.

Information domain contains three different views of the data and control:

- information content and relationship:
  information content --> represent the individual data and control objects

  there are different relationships between data and objects

- information flow:
  represents the manner in which data and control change as each moves through a system. Data and control moves between two transformations (functions)

- information structure:
  represent the internal organization of various data and control items
    - data tree structure
    - data table (n-dimension)
Modeling

During software requirements analysis, we create models of the system to be built. The models focus on:

- what the system must do, not how it does it.

The models usually have a graphic notation to represent:

- information, processing, system behavior, and other features

The second and third operational analysis principles require:

- build models of function and behavior

- Functional models

  Software transforms information. Three generic functions:

  - input, processing, output

- Behavior models

  Most software responds to events from the outside world
  A behavior model creates a representation of the states of the software and events that cause software to change state

Important roles of models:

- The model aids the analyst in understanding the information, function, and behavior of a system.
- The model becomes the focal point for review in the aspects of completeness, consistency, and accuracy of the specification.
- The model becomes the foundation for design, providing the designer with an essential representation of software.
Partitioning

Partitioning decomposes a problem into its constituent parts.

Establish a hierarchical representation of information (or function):
- exposing increasing detail by moving vertically in the hierarchy
- decomposing the problem by moving horizontally in the hierarchy.

Horizontal partition

Vertical partition
Software Prototyping

In some cases, it is possible to apply operational analysis principles and derive a model of software from which a design can be developed.

Selecting the prototyping approach:

The closed-ended approach is called throwaway prototyping.
   - a prototype serves only as a rough demonstration of requirements.

The open-ended approach is called evolutionary prototyping.
   - a prototype serves as the first evolution of the finished system.

Figure 11.7 selecting the appropriate prototyping approach.

Prototyping Methods and Tools:

- Fourth Generation Techniques
- Reusable Software Components
- Formal Specification and Prototyping Environments
Software Specification

Specification principles:
- Separate functionality from implementation
- Develop a model of the desired behavior of a system
- Establish the context in which software operates
- Define the environment in which the system operates
- Create a cognitive model rather than a design or implementation model
- Specification is an abstract model of a real system
- Establish the content and structure of a specification (easy to be changed)

Guidelines for representation:
- Representation format and content should be relevant to the problem
- Information contained within the specification should be nested
- Diagrams and other notational forms should be restricted in number and consistent in use.
- Representations should be revisable

Software requirements specification standard:
IEEE (standard No. 830-1984) and U.S. Department of Defense

In many cases, a preliminary user’s manual should be provided to present the software as a black box.