Chapter 5: Analysis Concepts and Principles

Complied by Heng Sovannarith
heng Sovannarith@yahoo.com
Introduction

• Software requirements engineering is a process of discovery, refinement, modeling, and specification.

• Both the software engineer and customer take an active role in software requirements engineering—a set of activities that is often referred to as analysis.
Introduction (cont.)

• The customer attempts to reformulate a sometimes nebulous system-level description of data, function, and behavior into concrete detail.

• The developer acts as interrogator, consultant, problem solver, and negotiator.
Requirements Analysis

- **Requirements analysis** is a software engineering task that bridges the gap between system level requirements engineering and software design.
Requirements Analysis

• Requirements engineering activities result in:
  1. The specification of software’s operational characteristics (function, data, and behavior).
  2. Indicate software's interface with other system elements
  3. Establish constraints that software must meet.
Requirements Analysis (cont.)

• **Requirements analysis** allows the software engineer (analyst) to **refine** the software allocation and build models of the **data, functional, and behavioral domains** that will be treated by software.

• Requirements analysis provides the software designer with a **representation of information, function, and behavior** that can be **translated to data, architectural, interface, and component-level designs**.

• Finally, the requirements specification provides the developer and the customer with the **means to assess quality** once software is built.
Requirements Analysis (cont.)

• Software requirements analysis may be divided into five areas of effort:
  1. Problem recognition,
  2. Evaluation and synthesis
  3. Modeling,
  4. Specification
  5. Review.
Requirements Elicitation for Software

• Before requirements can be analyzed, modeled, or specified they must be gathered through an elicitation process.
• A customer has a problem that may be amenable to a computer-based solution. A developer responds to the customer's request for help.
• Communication has begun.
Requirements Elicitation for Software (cont.)

- **Initiating the Process**
  - The most commonly used requirements elicitation technique is to conduct a *meeting or interview*.
  - The first meeting between a *software engineer (the analyst)* and the customer can be likened to the *awkwardness* of a first date between two adolescents.
  - Gause and Weinberg suggest that the analyst start by asking *context-free questions*. That is, a *set of questions* that will lead to a *basic understanding of the problem*, the people who want a solution, the nature of the solution that is desired, and the effectiveness of the first encounter itself. The first set of context-free, questions focuses on the *customer*, the *overall goals*, and the *benefits*. 
Requirements Elicitation for Software (cont.)

• For example; the analyst might ask
  • Who is behind the request for this work?
  • Who will use the solution?
  • What will be the economic benefit of a successful solution?
  • Is there another source for the solution that you need?
• These questions help to identify all stakeholders who will have interest in the software to be built. In addition, the questions identify the measurable benefit of a successful implementation and possible alternatives to custom software development.
Requirements Elicitation for Software (cont.)

• Facilitated Application Specification Techniques
  – A number of independent investigators have developed a team-oriented approach to requirements gathering that is applied during early stages of analysis and specification.
  – Called *facilitated application specification techniques* (FAST), this approach encourages the creation of a joint team of customers and developers who work together to identify the problem, propose elements of the solution, negotiate different approaches and specify a preliminary set of solution requirements.
Requirements Elicitation for Software (cont.)

**FIGURE 11.2.**
The FAST meeting
Basic guidelines for FAST:

- Meeting is arranged at a neutral site and attended by both s/w engineers and customers.
- Rules for preparation and participation are established.
- Agenda is set.
- Appoint a facilitator to control the meeting (can be developer, customer or outside expert.)
- Participants should not criticize and debate.

Goal should be to identify problems, propose solution and identify a preliminary set of requirements.
Requirements Elicitation for Software (cont.)

FAST session preparation:

• Initial meetings between customer and developer occur where scope is established and an overall perception of a solution is determined.

• ½ page product request is documented.

• Meeting place, time, date for FAST are selected and facilitator is chosen.

• Product request is distributed to all attendees before meeting date.
Requirements Elicitation for Software (cont.)

Every FAST attendee is asked to make:
1. List of objects.
2. List of services.
3. List of constraints.
4. List of performance criteria.

Activities of FAST session.
• Each participant presents his/her list of constraints.
• Combined list is prepared eliminating redundant entries and adding new ideas.
• List is finalized by the facilitator.
A microprocessor based home security system needs to be built that would protect against a variety of undesirable situations such as smoke, fire, illegal entry etc. The product would use appropriate fire and smoke detectors, window and door sensors and an alarm that would be set when an untoward situation occurs and automatically telephone a monitoring agency.

**List of objects**: Fire detector, smoke detector, window and door sensors, telephone, alarm.

**List of services**: Setting the alarm, monitoring the sensors, dialing the phone etc.

**List of constraints**: Manufacturing cost less than 80 $, must be user friendly, must interface directly to the standard phone line.

**Performance Criteria**: Sensor event should be recognized within 1 sec.
What is requirement?

• What is a Requirement?
  – “A condition or capability needed by a user to solve a problem or achieve an objective.”
  – “A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.”
Kinds of Requirements

• Functional requirements
  – capture the intended behavior in terms of services, tasks or functions the system is required to perform. [Malan et al. 1999]
  – Problem: if too general, ambiguity reigns; if too specific, design is stifled and leads to a large document;

• Techniques for writing them
  – Use cases,
  – Requirements List—“Shall” statements,
Kinds of Requirements (cont.)

• Non-functional requirements (or system qualities)
  – capture required properties or qualities of the system
  – often means: how well some behavioral or structural aspect of the system should be accomplished [Malan et al. 1999]

• two categories:
  – Observable at runtime, e.g., performance, security, reliability, availability, usability, etc.
  – Not observable at runtime, e.g., extensibility, portability, reusability, etc.
Other Things that Need to be Taken into Account

• Project constraints
  – define how the eventual system must fit into the world and what rules must be followed in its development.
    • Organizational constraints, e.g., deadlines, budget, process standards, business rules;
    • Operational constraints, e.g., mandated technologies, interfaces to hardware and other software;
    • Legislative and Ethical constraints, e.g., safety, privacy, health regulations/standards.
Other Things that Need to be Taken into Account (cont.)

• Project issues
  – define the ideas, concerns, and issues related to the project:
  – Open issues
  – Installation and transition issues
  – Risks
  – Estimated cost
  – Change Cases
  – Ideas for solutions and off-the-shelf options
Example Table of Contents for Requirements Document

1. The Purpose of the Document
2. The Purpose of the System
3. Stakeholders of the System
4. Naming Conventions, Definitions, and Assumptions
5. Project Constraints
6. The Scope of the Work and System
7. Functional Requirements
8. Non-functional Requirements
9. Project Issues

Use Cases are found here
Use Case

• A use case is a methodology used in system analysis to identify, clarify, and organize system requirements.

• The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal.
Use Case (cont.)

• Use case strengths are
  – That they work well as an analytical tool
  – That the notation is simple and easy to pick up
  – That they are easy to understand, both for the business and from the technological aspect
  – It is a widely recognised market standard
  – That customer and supplier – or operators and technicians – can jointly work out and understand the operational functionality
  – They bring structure, and ensure complete analysis
Use Case (cont.)

• Definition:
  o diagram which provides an overview of system functionality
  o Shows which use cases the individual actor uses

• Purpose:
  o To analyse the functionality the system must include
  o To give an overview of the functionality and how it is linked
  o To analyse how the actors should use the system

• Challenges:
  o To simplify the complex
Actor:

An actor is a person, organization, or external system that plays a role in one or more interactions with your system.

System boundary:

System boundary indicates the scope of your system. Anything within the box represents functionality that is in scope, and anything outside the box is not.

Use case:

A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.