Requirement Analysis Fundamentals

Peter Lo

Requirements Analysis

- The process of **Discovering**, **Refinement**, **Modeling** and **Specification** in a software project.
- Involve both the **Customer** and **System Engineer**
- Allowing system engineer to achieve a set of objectives:
  - Specify software function and performance
  - Indicate software interface with other system elements
  - Establish design constraints that the software must meet
- Provides the software designer with a representation of Information and Function that can be translated to Data, Architectural and Procedural Design.

Requirements Analysis

- Preparation of the **Software Specification**:
  - It is a formal document that specifies in precise terms the functional and performance requirements of the software.
  - Specification document in turn will allow the developer and customer to assess quality once the software itself has been built.
- The software developer performing Requirement analysis would often be known as the **Analyst**

Software Requirement Analysis

- Divided into 5 main areas:
  - Problem Recognition
  - Evaluation and Synthesis
  - Modeling
  - Specification
  - Review

Problem Recognition

- Recognize the basic problem elements perceived by user and customer
- Understand software in the system context
- Define software scope
- Establish contact with management and technical staff of the customer and software development organization

Evaluation and Synthesis

- Evaluate the flow and content of information
- Define and elaborate all software functions
- Understand software behavior in the context of events that affect the system
- Establish system interface characteristics and uncover design constraints
  - Emphasis on what must be done, not how it will be done
  - This step will continue until both the analyst and customer feels confident that software can be adequately specified for subsequent development steps

Modeling

- Create models of the system that will enable better understanding of data and control flow
- Describe the data and control flow, functional processing, behavioral operation, and information content
- Models serve a number of important roles:
  - Aids in understanding the information, function and behavior of a system
  - Makes requirement analysis task easier and more systematic
  - Serves as a basis for creating specification for the software
  - Becomes the focal point for review
  - Becomes the foundation for design

Specification

- A representation of software that can be reviewed and approved by the customer.
- Developed as a joint effort between the developer and the customer.

Specification Principles

1. Separate Functionality from Implementation
2. A Process-oriented Systems Specification Language is Required
3. Encompass the System of which the Software is Component
4. Encompass the Environment in which the System Operates
5. Cognitive Model
6. Operational
7. Tolerant of Incompleteness and Augmentable
8. Localized and Loosely Coupled

Separate Functionality from Implementation

- A specification is a description of what you want (i.e. you specify), as opposed to how it is realized (i.e. implementation).

A Process-oriented Systems Specification Language is Required

- In a dynamic environment, the behavior of the entity cannot be expressed as a mathematical function of its input
- A process-oriented description must be employed, in which the "what" specification is achieved by specifying a model of the desired behavior in terms of functional responses to various stimuli from the environment

Encompass the System of Software is Component

- System is made up of interacting components
- Specifications must be made in context of entire system and interaction between its parts

Encompass the Environment the System Operates

- The environment in which the system operates and how it interacts with must be specified

A Cognitive Model

- It means that the specification must describe a system as perceived by its user community, and should not be too abstract.

Operational

- The specification must be complete and formal enough such that it can be satisfy an implementation of some arbitrary test cases.

Tolerant of Incompleteness and Augmentable

- The specification must be robust enough to undergo changes, expansion.

Localized and Loosely Coupled

- Localized means to affect one piece only
- Loosely coupled means that parts can be removed or added easily
- The specification must be such that its content and structure should be able to accommodate dynamic changes, and that whatever information changes there are, it should affect one component only.

Review

- Review of analysis documents like specification
- Conducted at a macroscopic level.
- Conducted by customer and developer.
- Results in modifications:
  - Functions
  - Performance
  - Information representation
  - Constraints
  - Validation criteria

Responsibility of Analyst

- To analyze and define systems of optimum performance
  - i.e. an output that fully meets management objectives

Ability that Analyst should have

- Grasp abstract concept, partition them and generate solutions based on each division
- Understand implicit information, separate them and treat them individually
- Absorb pertinent facts from conflicting sources
- Understand the customer environment
- Apply hardware and/or software system elements to the customer environment
- Communicate well in written and verbal form

Problems in Requirement Analysis

- Requirement analysis is a communication-intensive activity.
- Problems like miscommunication and omission often occur between analyst and customer.
- Successful acquisition of information cannot be guaranteed because analyst have difficulties:
  1. In getting pertinent (appropriate) information
  2. Handling large and complex problems
  3. Accommodating changes that occur during and after analysis

Causes for the Problems

- Poor communication that makes information acquisition difficult
- Inadequate techniques and tools for developing specification
- Tendency to take short-cuts during requirements analysis tasks, leading to unstable design
- Failure to consider alternative solutions before software is specified on both parts of analyst and customer.

Communication Techniques

- Communication techniques are not fact-finding (information gathering) techniques,
- Two techniques are available to tackle these communication problems
  - Preliminary Meeting or Interview
  - Facilitated Application Specification Technique (FAST)

Preliminary Meeting or Interview

- Proposed by Gause and Weinberg.
- The commonly used analysis technique to bridge the communication gap between the customer and developer.
  - Though effective for a first meeting, it should not be used for subsequent meetings between the customer and software developer.

Preliminary Meeting or Interview

- The technique comprises of 3 different sets of questions
- These questions help to initiate the communication that is essential to successful analysis.
- This approach should be used as a first encounter only, and then replaced by a meeting format that combines elements of problem solving, negotiation and specification.

First Set of Questions

- These questions lead to a basic understanding of the problem
- Focuses on the customer and overall goals and benefits
- Example:
  - Who is behind the request for this work?

Second Set of Questions

- These questions enable the analyst to gain a better picture of the problem.
- Allow the customer to voice his perceptions about a solution.
- Example:
  - What problems will this solution address?

Third Set of Questions

- Focuses on the effectiveness of the meeting itself.
- These questions are often called “meta-questions”.
- Example:
  - Are you the right person to answer these questions?

Facilitated Application Specification Technique (FAST)

- A technique that is used after the first meeting is completed and basic understanding has been achieved.
- It proposes a meeting format that combines elements of Problem Solving, Negotiation, and Specification.
- The FAST mentality encourages the working together mentality rather than working individually.
- The technique is a team-oriented approach, the team jointly made up of customers and developers.

Goal of FAST

- The basic **Goals** of the FAST meeting are
  - Identify the problem
  - Propose elements of the solution
  - Negotiate different approaches
  - Specify a preliminary set of solution requirements.

Basic Guidelines of FAST

- There are different approaches to FAST, but all of them have the following basic guidelines:
  - Meeting is conducted at a neutral site
  - Rules for preparation and participation are established
  - An agenda is suggested to cover all the important points.
  - A "facilitator" is appointed to control the meeting

Fundamental Principles of Analysis Methods

- The information domain of a problem must be represented and understood
- The functions that the software is to perform must be defined
- The behavior of the software must be represented
- The models that depict information, function, and behavior must be partitioned in a manner that uncovers detail in a layered fashion
- The analysis process should move from essential information toward implementation detail

Davis's Guiding Principles for Requirements Engineering

- Understand the problem before you begin to create the analysis model
- Develop prototypes that enable a user to understand how human-machine interaction will occur
- Record the origin of and the reason for every requirement
- Use multiple views of requirements
- Prioritize requirements
- Work to eliminate ambiguity

Data Processing

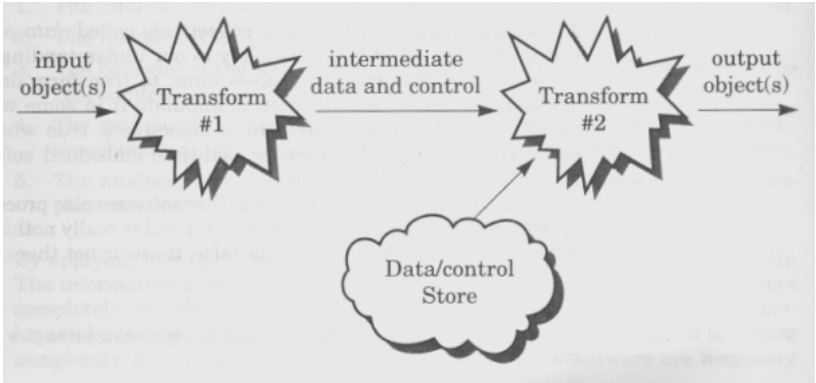
- Software is built to process data, to transform data from one form to another

Information Domain

- The information domain contains 3 different views of the data and control as a processed by a computer system:
  - Information Flow
  - Information Content
  - Information Structure

Information Flow

- Represents the manner in which data and control change as each moves through a system.
- The transformations that are applied to the data are functions or sub-functions that a program must perform.



Information Content

- Represents the individual data and control items that comprise some larger item of information that is transformed by the software.
- For example, the data object paycheck is a composite of a number of important pieces of data: the payee's name, the net amount to be paid, the gross pay, deductions, etc.

Information Structure

- Represents the internal organization of various data and control items

Modeling

- Models are created to gain a better understanding of the actual entity or object to be built.
- Make use of a graphic notation that depicts Information, Processing, System Behavior, and other characteristics as distinct and recognizable symbols.

Functional Models

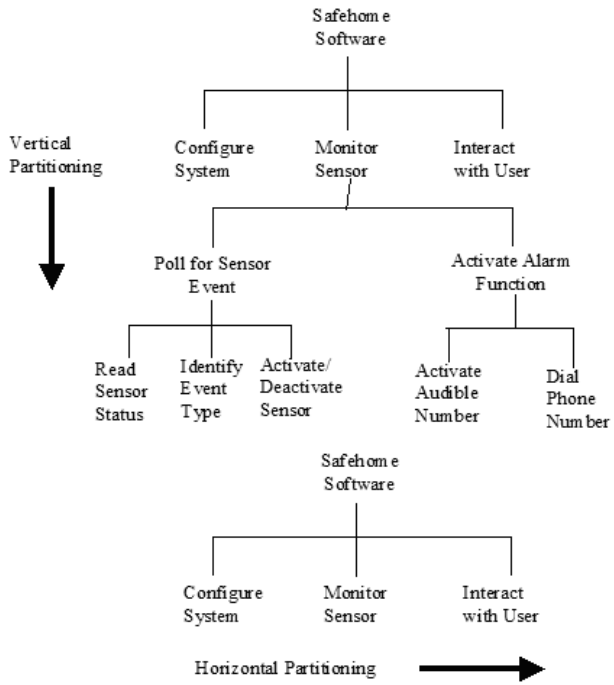
- Software transforms information, and in order to accomplish this
- It must perform at least three generic functions:
  - Input
  - Processing
  - Output.

Behavior Models

- Software responds to events from the outside world.
- This stimulus-response characteristic forms the basis of the behavioral model.
- Creates a representation of the different states that the software can exist in, and the events that would cause this software to change from one state to another.

Partitioning

- Problems are often too large and complex to be understood as a whole, so we partition them.
- The word partition means to divide into parts.
- Interfaces between these divided parts must be established so that the overall function can be accomplished.



Partitioning

- Horizontal partitioning
  - Greater detail as we move from left to right in the software structure.
- Vertical partitioning
  - Greater detail as we move from top to down in the software structure.

Essential and Implementation Views

- Essential View
  - Presents the functions to be accomplished and information to be processed without regard to implementation details.
- Implementation View
  - Presents the real world manifestation of processing functions and information structures. It is concerned with the physical how-is-it-going-to-be-done aspect.

Specification Review

- Review of a software requirements specification is conducted by both software developer and customer.
- Once the review is complete, a software requirements specification is “signed off” by both customer and developer.
- The specification becomes a “**Contract**” for software development.
- Requirements change after the specification is finalized will not be eliminated from consideration; but the customer should note that after-the-fact change is an extension of software scope and therefore can increase cost and/or protract the project schedule.

Specification Review

- Review of a software requirements specification is conducted by both software developer and customer.
- Once the review is complete, a software requirements specification is “signed off” by both customer and developer.
- The specification becomes a “contract” for software development.
- Changes in requirements requested after the specification is finalized will not be eliminated from consideration; but the customer should note that after-the-fact change is an extension of software scope and therefore can increase cost and/or protract the project schedule.