Requirements Analysis is Hard

“The hardest single part of building a software system is deciding what to build. No part of the work so cripples the resulting system if done wrong.”

“The seeds of major software disasters are usually sown in the first three months of commencing the software project.”

Requirements Analysis is Hard

Requirements Engineering

Requirements engineering is accomplished through the execution of seven major tasks:

- **Inception**
  - Roughly define scope

- **Elicitation**
  - Define requirements

- **Elaboration**
  - Further define requirements

- **Negotiation**
  - Reconcile conflicts

- **Specification**
  - Create analysis model

- **Validation**
  - Ensure quality of requirements

Requirements Management (Umbrella Activities)

Requirements Process

Collecting the user’s requirements

Understanding and modeling the desired behavior

Documenting the behavior of the proposed system

Checking that specification matches user’s requirement

Software Requirements Specification
Elicitation is the Hardest Part!

- Problems of scale
  - Explicit and implicit requirements abound
- Problems of scope
  - System boundaries are ill-defined
  - Customers will provide irrelevant information
- Problems of understanding
  - Customers never know exactly what they want
  - Customers don’t understand capabilities and limitations
  - Customers have trouble fully communicating needs
- Problems of volatility
  - Requirements always change

The Demons of Ambiguity (Harry Robinson)

What are We Talking About?

- Why does the sign use the plural “children” instead of the singular “child”?
- Who qualifies as a “child”?
- What does it mean to be “present”?
- Does this rule apply only when school is in session?
  - What about weekends? Holidays? Nighttime?
- And what if these situations are combined?

Stakeholders

- Clients (who’s paying)
- Customers (who’s buying)
- Users (who’s using)
- Domain experts (who knows about problem)
- Market researchers (will anyone buy this)
- Software engineers (who will build it)

Types of Requirements

- Functional: behavior, features
- Non-functional or quality: characteristic
- Design constraint: choice of platform, etc.
- Process constraint: resources, techniques, …
Requirements Documents

• Requirements Definition
  – Complete listing of everything customer wants
  – Described in terms of the operating environment

• Requirements Specification
  – Spec of how the proposed system will behave
  – System boundary clearly identifies what is in the system from what is outside (env)

Requirements Management Tools

• With so many requirements to manage, we need tools to deal with the complexity
  – Features traceability tables
  – Source traceability tables
  – Dependency traceability tables
  – Subsystem traceability tables
  – Interface traceability tables
  – Requirements traceability matrix (RTM)
    • One of the most popular tools in practice

The Requirements Traceability Matrix (RTM)

An RTM is a set of tables that links requirements to system modules, and system modules to test cases.

Here is an example from an open source project:
http://yaktrack.sourceforge.net/yaktrack_docs/a2332.html

Project Inception

• During the initial project meetings, the following tasks should be accomplished
  – Identify the project stakeholders
    • These are the folks we should be talking to
  – Recognize multiple viewpoints
    • Stakeholders may have different (and conflicting) requirements
  – Work toward collaboration
    • It’s all about reconciling conflict
  – Ask the first questions
    • Who? What are the benefits? Another source?
    • What is the problem? What defines success? Other constraints?
    • Am I doing my job right?
Collaborative Elicitation

- One-on-one Q&A sessions rarely succeed in practice; collaborative strategies are more practical
- Collaborative elicitation should result in several work products.
  - A bounded statement of scope
  - A list of stakeholders
  - A description of the technical environment
  - A list of requirements and constraints
  - Any prototypes developed
  - A set of use cases
    - Characterize how users will interact with the system
    - Use cases tie functional requirements together

Narrative Use Case

- A use case describes a sequence of steps in an interaction.
- Buy a Product Scenario:
  - The customer browses the catalog and adds desired items to the shopping basket. When the customer wishes to pay, the customer describes the shipping and credit care information and confirms the sale. The system checks the authorization on the credit card and confirms the sale both immediately and with a follow-up e-mail.

The Analysis Model

(Specification)

- Purpose
  - Precisely describes desired function and performance
  - Precisely describes all relevant constraints
  - Serves as the foundation for all subsequent activities
- Structure
  - Consists of many different views
  - Views range from informal to semi-formal to fully-formal

Validating the Analysis Model

- Before design activities can proceed, the analysis model must be validated
  - Is the model intellectually manageable?
  - Does the model reflect the system to be built?
  - Are the requirements consistent?
  - Is each requirement bounded and unambiguous?
  - Is each requirement achievable?
  - Is each requirement really necessary?
  - Is each requirement testable?
  - Does each requirement have attribution?
The Analysis Model

The analysis model consists of a wide variety of diagrammatic forms used to bridge an important gap.

- System information
- System function
- System behaviors

Purpose:
- Describe what the customer wants built
- Establish the foundation for the software design
- Provide a set of validation requirements

Some Rules of Thumb

- Make sure all points of view are covered
- Every element should add value
- Keep it simple
- Maintain a high level of abstraction
- Focus on the problem domain
- Minimize system coupling

Analysis Modeling Approaches

Structured Analysis
- Models data elements
  - Attributes
  - Relationships

Object-Oriented Analysis
- Models analysis classes
  - Data
  - Processes

- Models processes that transform data
- Models class collaborations

Techniques from both approaches are typically used in practice.

Data Objects

A data object is a domain element that will be manipulated by the system.

Characteristics:
- Plays a necessary role
- Characterized by attributes
- Uniquely identifiable

Examples:
- Roles
- Events
- Places
- External entities
- Structures
- Other things

Object: car
- Attributes:
  - VIN #
  - Make
  - Model
  - Price
Relationships, Cardinality, Modality

- **Relationships**
  - Define connections between objects

- **Cardinality**
  - Defines the number of items on either end of a connection

- **Modality**
  - Defines the necessity of a connection

Entity-Relationship Diagram (ERD)

Person → Car (owns)
Person → Trailer (insured)
Car → Trailer (attached)