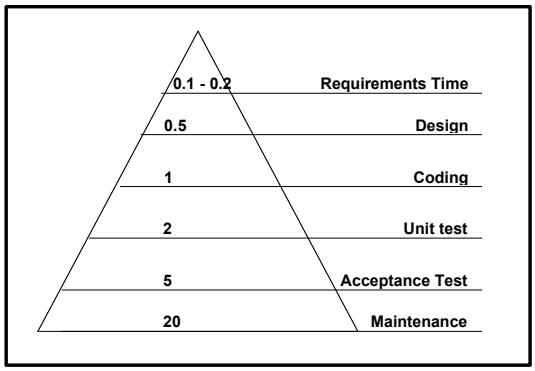
Requirements Engineering

- Bridge from customer needs to design and construction
  - "Computers are useless. They can only give you answers." -- Pablo Picasso
- Requirements engineering tasks
  - Inception - starting a project
  - Elicitation - getting requirements
  - Elaboration - building initial models
  - Negotiation - balancing/prioritizing
  - Specification - making reqts specific
  - Validation - are these the right requirements?
  - Requirements management
    - How will they change and be organized?

Why are requirements important?

- Relative cost to repair a defect discovered at different phases of development



Requirements Principles

- Information domain must be represented and understood
- Functions of the software must be defined
- Behavior with respect to external events must be represented
- Models must be partitioned in a manner that uncovers detail in a layered or hierarchical fashion
- Analysis should move from essential information toward implementation detail
  - "It is better to get the right output looking ugly than to get the wrong output looking just fine." -- Bjarne Stroustrup

Inception

- Origin of project
  - Business need identified
  - Marketing opportunity identified
- Identify stakeholders
  - Anyone who benefits or is affected directly or indirectly by the system being developed
- Recognizing multiple views of the system
- Working toward collaboration

Elicitation process

- Collaborative requirements gathering
  - Conduct meetings with stakeholders, facilitator, low tech tools
- Goals are:
  - Understand the system as-is and its context
  - Identify the problems and opportunities calling for a new system
  - Discover the real needs of stakeholders with respect to the new system
  - Explore alternative ways the new system can address the needs
- Understand priorities
  - Normal requirements - goals and functions of the system
  - Expected requirements - implicit requirements
  - Exciting reqts - cool and interesting beyond current expectations
- User scenarios (use cases)
- Prototyping
- Role playing

Products of Elicitation

- A statement of need and feasibility
- A bounded statement of scope for the system
  - Key step to ensuring that the product developed is the product requested by the customer.
  - A way to determine when the project ends (i.e., when they will get paid!).
  - For any process model, overall problem (scope) will be decomposed along functional lines into smaller subproblems.
- A list of stakeholders
- Description of technical environment
- List of requirements and their constraints
- Set of use cases
- Prototypes that help understand requirements

Use Case Model

- Identify who, what and how of system behavior
- Focus is on what system "does" for the users
- Use case describes a sequence of actions a system performs that yields a result of some value to a particular actor
- Use case model consists of:
  - All the actors of the system
  - All of the various cases by which actors interact with the system
- Limitations
  - May end up focusing on specific stakeholders, excluding others
  - Not very useful for determining non-functional requirements (reliability, performance, etc.)
  - May lead stakeholders to get bogged down in details

Guidelines

- Use stakeholders' own language
- Let user interaction help determine user interface
- Think about use cases as helping guide testing and validation
- Remember some use cases involve more than one actor
- Be prepared to identify additional actors or adjust system boundary with care!
- Develop several use cases with each stakeholder
  - Typical or normal uses of the system
  - Exceptional or infrequent uses of the system

Example Use Case

USE CASE
Use Case Name: Buy Items
Actors: Customer (initiator), Cashier
Purpose: Capture a sale and its payment
Overview: A customer arrives at a checkout with items to purchase. Cashier records purchase items and collects a payment.
Cross-references: R1.1, R1.2, R2.2, ...

Typical Course Of Events:

Actor Action	System Response
1. Customer arrives at checkout with items to purchase	
2. Cashier records each item	3. Determines item price, adds item info to running sales transaction
If there is more than one of an item, Cashier can enter quantity.	
	Description & price of current item presented
4. On completion of item entry, the Cashier indicates entry is complete	5. Calculates and presents the sale total
6. Cashier tells Customer the total	

Elaboration

- Build analysis model
  - Describe information, functional and behavioral aspects of the system domain
- Expect model to change!
  - Stable elements represent general agreement and understanding
  - Volatile elements represent things not yet understood
- Analysis "model" is usually a collection of models capturing different points of view

Elements of Analysis Models

- Scenario-based elements
  - User point of view
  - Use cases
- Class-based elements
  - Objects in the domain
- Behavioral elements
  - Represent changes to system state over time
- Flow-oriented elements
  - Information transformations as "flow" through the system

Analysis Concepts and Principles

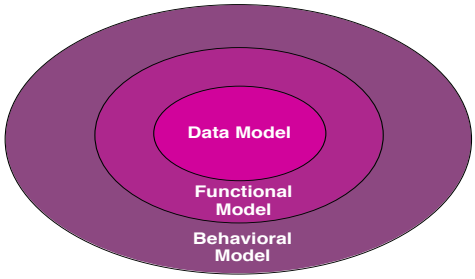
- Problems during requirements analysis
  - Customer doesn't completely understand what is desired
  - Final result is hazy
  - Analyst cannot understand customer's world
  - Personality conflicts between analyst and customer
  - Too many customers
- Analysis focus
  - Work with the customer to identify and refine information, functional, and behavioral requirements to solve customer's problems.

Validation of requirements

- Are the requirements right?
- Examine for
  - Consistency
  - Completeness
  - Ambiguity
- At right level of abstraction?
- Really necessary or a "frill"?
- Are they partitioned?
- Are they feasible?

Requirements Analysis

- Software requirements analysis bridges the gap between system requirements engineering and software design
- Software requirements analysis focuses on the process of modeling the data, function, and performance of the software to be developed



Analysis Goals

- Build working models of the information, functional characteristics, and behavioral performance of a software product
- Don't go straight to the implementation!
  - First consider the essential view
- Analysis Areas of Effort
  - Problem recognition
  - Evaluation and synthesis
  - Modeling
  - Specification
  - Review

Requirements Analysis Steps

- Identify the "customer" and work together to negotiate "product-level" requirements
- Build an analysis model
  - Focus on data
  - Define function
  - Represent behavior
- Prototype areas of uncertainty
- Develop a specification that will guide design
- Conduct formal technical review

Alan Davis' Principles

- 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.

Requirements Specification

- Not generally possible to completely separate software requirements specification from design
- Specification principles
  - Separate functionality from implementation
  - Develop model of desired system behavior that encompasses both data and functional responses of system to stimuli
  - Establish context in which software operates
  - Define environment in which system operates
- More Specification principles
  - Create cognitive model, not design/implementation model
    - Cognitive model describes a system as perceived by its user community
  - Recognize that specifications must tolerate incompleteness
    - Specification is always a model (abstraction) of some real (or proposed) thing
  - Establish content and structure of specification so it can be designed

Specification Review

- The purpose of the specification review is to make sure that the customer and developer agree on details of the software requirements (or prototype) before beginning the major design work
- Reviewers attempt to ensure that the specification is complete, consistent and accurate
  - Watch for vague words such as
    - some, sometimes, often, usually, ordinarily, most or mostly

Analysis Modeling Steps

- Generic analysis model
  - An entity-relationship diagram (data model)
  - A data flow diagram (functional model)
  - A state transition diagram (behavioral model)
- Data dictionary
  - a repository that contains descriptions of all data objects that consumed or produced by the software
- Acquire scope statement
  - A statement of scope can be acquired from:
    - the FAST working document
    - A set of use-cases

Analysis Modeling Steps (cont.)

- The statement of scope must be “parsed” to extract data, function and behavioral domain information
  - Define “objects” by underlining all nouns in the written statement of scope
    - Producers/consumers of data
    - Places where data are stored
    - “composite” data items
  - Define “operations” by double underlining all active verbs
    - Processes relevant to the application
    - Data transformations
  - Consider other “services” that will be required by the objects
- All icons must be labeled with meaningful names

Questions to consider...

- What are the three basic models in requirements and why is each of them important?
- What is a stakeholder? How do you know if you've identified all the stakeholders for a system?
- Why is it important to identify actors and a system boundary?
- What is a statement of scope and what is its purpose in developing software?
- Why do we distinguish the normal flow and exceptional flow in a set of use cases?
-