Analysis Principles

- Operational Principles
  - Information Domain must be represented and understood
  - Functions must be defined
  - Behavior (stimulated by external events) must be represented
  - Models must be partitioned in layered fashion
  - Process must move from Essential Information to Implementation Detail

Analysis Principles

- **Guiding 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 Origin and Reason for every requirement**
  - **Use multiple views of requirements**
  - **Prioritize requirements**
  - **Work to eliminate ambiguity**

The Information Domain

- Data Processing
  - Software is built to process Data
- Event Processing
  - Software also processes Events
  - Control
- Views
  - info content and relationships
  - info flow
  - info structure

The Information Domain

- Information Content
  - Attributes
  - Relationships
- Information flow
  - Input
  - Storage
  - Output
  - Transformations

The Information Domain

- **Information Structure**
  - **Data structure refers to design and implementation of information structure**
  - **How is Information related to other information**
  - **Is all information in one or many structures?**
  - **How do structures relate?**

Modeling

- **Functional**
- **Behavioral**

Model Roles

- Important Roles
  - Aids Analyst to Understand
    - Information
    - Function
    - Behavior
  - Focal Point for Review
    - Completeness
    - Consistency
    - Accuracy
  - Foundation for Design

Partitioning

- Partition
  - Divide problems that are too large and complex into parts that can be understood as a whole
- Hierarchical Representation
  - Move vertically
    - expose increasing details
  - Move horizontally
    - decompose the problem

Essential and Implementation Views

- **Essential View**
  - **presents functions without regard to implementation details**
- **Implementation View**
  - **presents the "real world" of processing functions and information structures**

Software Prototyping

- **Prototyping**
  - **Constructed for customer and developer assessment**
  - **Used to help derive requirements when other methods may fail**

Selecting the Prototyping Approach

- Closed-Ended
  - Throwaway prototyping
  - Serve as a rough demonstration of requirements
  - Software is engineered using a different paradigm
- Open-Ended
  - Evolutionary prototyping
  - Continue into design and construction

Selecting the Prototyping Approach

- Questions to Ask
  - Is the application domain understood by customer and developer?
  - Does problem lend itself to modeling?
  - Is customer certain of basic system requirements?
  - Are requirements established and stable?
  - Are any requirements ambiguous?
  - Are requirements contradictory?

Prototyping Methods and Tools

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

Representation

- Representation format and content should be relevant to problem
- Information contained within specification should be nested
- Diagrams should be restricted in number and consistent
- Representations should be revisable

The Software Requirements Specification

- Outline
  - Introduction
  - Information Description
  - Functional Description
  - Behavioral Description
  - Validation and Criteria
  - Bibliography
  - Appendix

Specification Review

- Review is first conducted at a macroscopic level
  - ensure that the specification is complete, consistent, and accurate
- Then focus at detailed level
  - uncover hidden problems
- Signed off by both customer and developer
  - change will not be eliminated
  - extension of scope
  - increase cost
  - protract schedule