

Requirements Engineering

21 April 2004

Regina M. Gonzales, Ph.D.
Sandia National Laboratories

rgonzal@sandia.gov

(505) 844-7238

Fred Brooks in *No Silver Bullet*, Wrote:

“The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is so difficult as establishing the detailed technical requirements.... No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.”
[IEEE Computer, 1987]

Semantic Gap

- ◆ Systems and software development models do not address requirements or pre-requirements consistently
- ◆ Diverse stakeholders
 - Clients or customers
 - Developers from many domain areas
 - Multiple users with diverse needs
 - Production, manufacturing, distribution, and support staff
 - Standards, regulatory agencies, and existing infrastructure

Some Notions on Requirements

“Aren’t requirements the job of marketing. Shouldn’t engineers design?” (A Faculty Colleague)

“You mean I don’t just sit in front of the word processor and type requirements?” (A Sandia Engineer)

“It sounds like your trying to put a clamp around jello.”
(College Advisor)

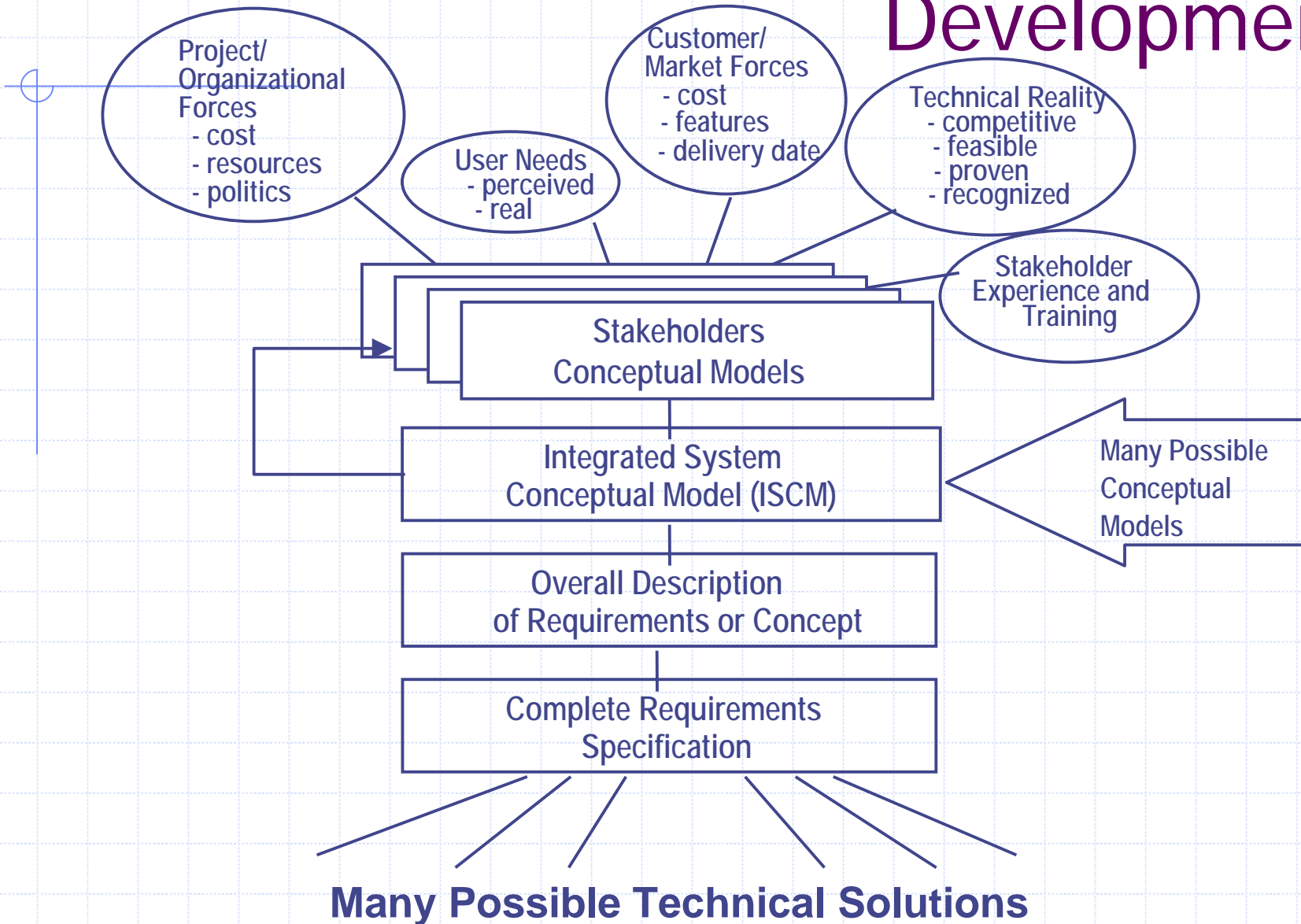
Requirements Engineering

- ◆ Elicit or capture – obtaining the raw material from which to analyze and formulate the actual requirements; understanding the problem
- ◆ Analysis – model, partition, organize information captured in order to understand fundamental elements of scope, ontology, behavior, and abstraction level
- ◆ Specify – writing the requirements in a form that is unambiguous, verifiable, and manageable
- ◆ Manage – update, use, and trace requirements
- ◆ Verify and validate – ensure that the requirements are being met at each step in the process of use
 - Either as the requirement is further specified or as it is realized in the actions of a system

Accepted and Maturing Approaches to Requirements Elicitation

- ◆ Win-Win Theory W (Boehm)
- ◆ Soft Systems and Participatory Design (Checkland)
- ◆ Scenario and Use Case Based Methods (Filippidou, Larmen)
- ◆ Knowledge Elicitation Techniques (Cooke)
- ◆ Facilitator-based Methods (Mcauley, Wood & Silver)
- ◆ Viewpoint Analysis (Darke & Shanks)
- ◆ Ethnography (Hughes)
- ◆ Problem Frames (Jackson)
- ◆ Domain Modeling (Sztipanovits)

Context for Requirements Development



The Role of the Requirements Engineer

- ◆ Incumbent on engineers to close the semantic gap
 - Take responsibility for the acquisition and formalization of requirements
- ◆ Systems integrator at the conceptual level
- ◆ Responsible for representing stakeholder information in terms of “what” needs to be done
- ◆ Stay in an ‘ask and listen only’ mode during the beginning steps of the process
- ◆ Must convey the importance of the stakeholder’s unique viewpoint and must train the stakeholder to actively participate
- ◆ Primary role is to build the system conceptual model

Recommended Steps in the Elicitation

- ◆ Identify stakeholders or stakeholder groups
 - System context
 - Project and product lifecycle
 - Operational scenario
 - Who pays the bills?
- ◆ Elicit individual stakeholder viewpoints
 - Formalize using a consistent notation – models help
 - Conduct stakeholder follow-up for questions
 - Run interference on potential conflicts or gaps
- ◆ Integrate viewpoints
 - Integrated System Conceptual Model (ISCM)
- ◆ Iterate ISCM with the stakeholders

Mental Models & Conceptual Models

- ◆ Extensive psychological literature on mental models mostly applied to existing systems
- ◆ Norman [1983, 1988] describes them as a cognitive representation which “evolves naturally from interaction with the target system... need not be technically accurate ... need not be complete... is functional... used to explain the observable”
- ◆ Conceptual models are an aggregation of mental models and perhaps existing models for systems
- ◆ Used extensively in systems engineering community
- ◆ Wieringa states, “The hallmark of conceptual models is that they are conceptual structures used as a framework for communication between people”

Trends in Requirements Elicitation

- ◆ Goal-oriented Requirements
 - Goes back further in the life cycle to system objectives
 - Formalizes the postulation of these objectives
 - Less formal application of goal and sub-goals of stakeholders can be applied
 - ◆ Assists in evolving a Use Case architecture from the viewpoint of the stakeholders
- ◆ Specialization of Requirements Elicitation
 - As maturity in the area of elicitation occurs there is an effort to understand the uniqueness of requirements
 - ◆ For example performance versus constraint requirements
 - Another similar trend is understanding of elicitation methods that best apply to the different sources of requirements
 - ◆ I developed a repeatable method to go from written text to an object model, I use it as part of domain modeling based on stakeholder transcripts or other written text

Requirements Analysis

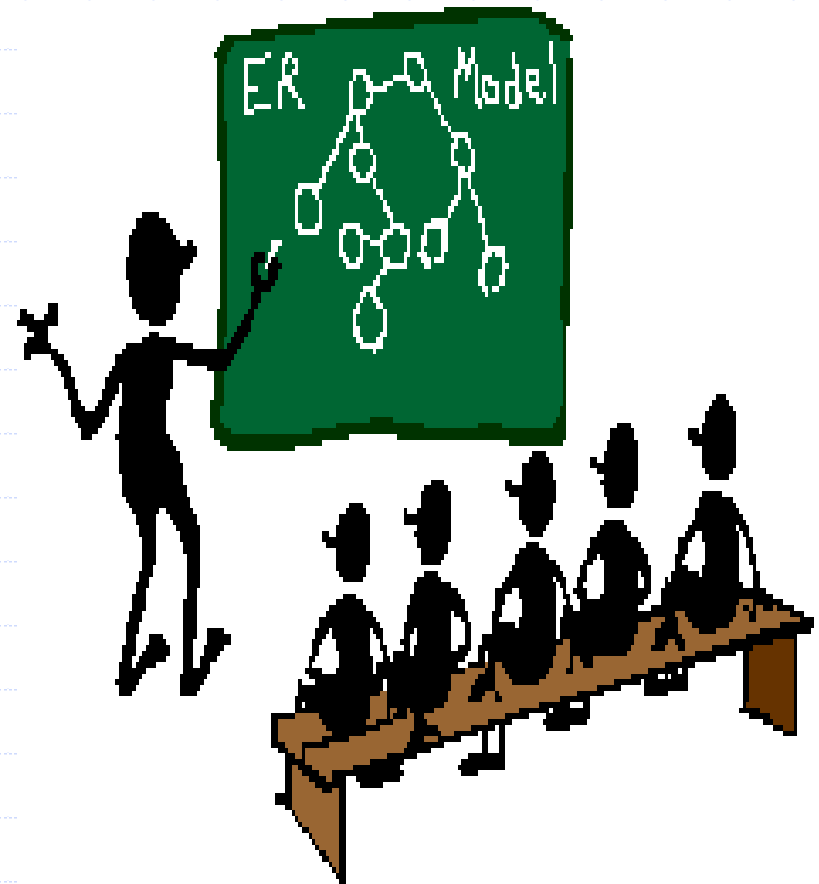
- ◆ Different Approaches
 - Formal
 - Informal but Rigorous
 - Ad Hoc
- ◆ 'Leveling' of System, Decomposition
- ◆ System Boundaries
- ◆ Determining Interconnectedness
- ◆ Analyzing Technical Constraints
- ◆ Use Methods
 - Object-oriented, Structured Analysis, Prototyping

Using Requirements Analysis for Understanding Domain Ontology

- ◆ Ontology
 - More than terminology, taxonomy, definitions ...
 - Closer to a semantic network
 - What are the “things” discussed and recognized by the people in the domain and how do those “things” relate to one another
- ◆ During elicitation and analysis developing a shared ontology is crucial
 - Most of the time the people in the domain don't have a shared ontology this requires negotiating a working ontology
- ◆ Having a defined and understood ontology simplifies specifying and managing requirements

Model-based Requirements Analysis

- ◆ Models and modeling methods help formalize the static and behavioral views of a problem
- ◆ Models are created first from the perspective of the problem domain and the enterprise in which the solution will fit
- ◆ Models are used as part of analysis because they help achieve a shared understanding prior to specifying requirements
- ◆ UML has created a common discourse language for modeling human systems, processes, systems, and software even if imperfect



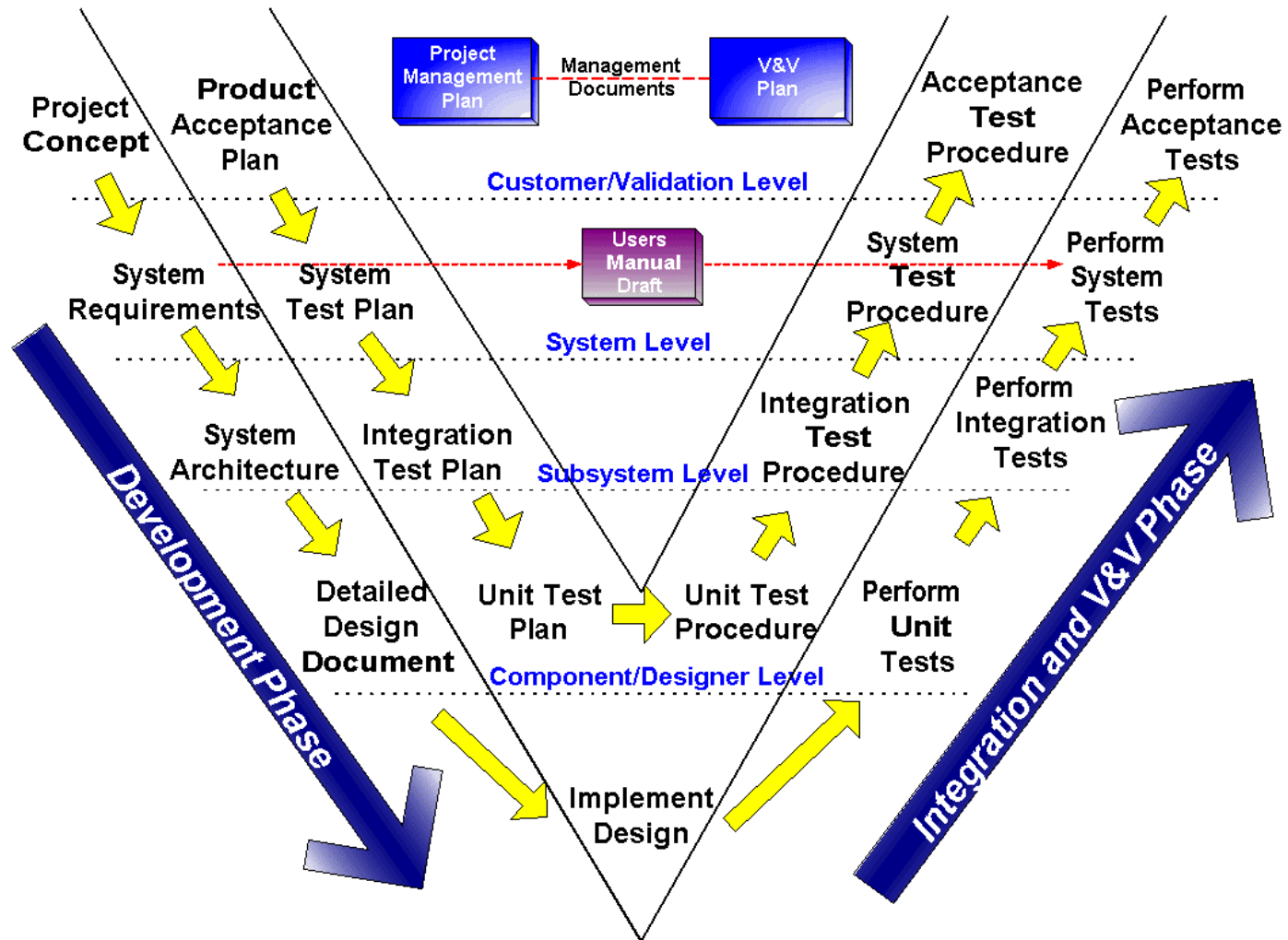
Requirements Analysis Based on Models

- ◆ Two main attributes of “things” defined in requirements
 - Static – relationships, partitioning, abstraction, generalization, composition
 - Behavior – roles, responsibilities, process, transactions, state-change, timing, order of events
- ◆ Assigning behavior clearly and concisely requires understanding the static view
 - Eliminates redundancy, inconsistency, and makes it more manageable
 - Example: *Product Definition* is a composition that includes *Drawings*, of which *Level 1* is an instance. When assigning behavior, what behavior belongs to one versus the other

A Vehicle for the Elicitation and Domain Analysis

- ◆ Establishing a Shared Vision with a Concept Document
 - If you don't know where you are any road will get you further lost
- ◆ A commercial necessity before requirements
 - Marketing documents don't cut it
 - Project Plans main focus on the project not product
 - Requires a rigorous approach – lifecycle
- ◆ Keystone in the process
 - Sets up the full definition of requirements
 - Basis for Validation
 - Basis for V&V Plan
 - Input to iteration of Project Plan

Process Dependencies



Requirements Specification

- ◆ There is a certain amount of Analysis that happens during specification
- ◆ The process of writing down requirements in a logical, readable fashion
- ◆ It contains functional and non-functional requirements
- ◆ Most important attribute is that they be verifiable
- ◆ Natural Language Specifications (IEEE 830)
- ◆ Formal Languages
 - Z (zed), Larch, Marvin, Prolog

Trends in Requirements Specification

- ◆ Specifying as much of the behavior for a system using use cases
 - Practically speaking some functionality for systems has to be specified separate from use cases
- ◆ Using a database of requirements as your specification for the system
- ◆ Compromise between formal specifications and natural language specifications
 - Narrow the lexicon for specifying requirements and restrict the sentence construction
 - Allows for human readability and use of sophisticated natural language processing techniques to perform axiomatic and logical analysis of requirements

Attributes of an Requirements Specification

- ◆ The 'what' not the 'how'
 - Apply filter to stakeholder input
- ◆ Correct
 - According to current system knowledge
- ◆ Unambiguous - Clarity
 - Conflict
- ◆ Complete ***
- ◆ Consistent ***
- ◆ Modifiable
 - Organized
 - Non-redundant
- ◆ Ranked for importance and/or stability
 - Have stakeholders apply filters later
- ◆ Verifiable
 - This takes engineering judgment when writing specific requirements
- ◆ Traceable
 - Backward
 - Forward

Requirements Management

- ◆ A tool is very helpful
 - Bookkeeping function for managing requirements
- ◆ Assign attributes to each requirement to track status, priority, changes, etc.
- ◆ Making Traceability Links
- ◆ Maintain archive of all changes as a metrics
- ◆ All changes to the product throughout it's lifecycle should be evaluated for impact on requirements
- ◆ Some degree of requirements analysis here as well

Examples Of Attributes Include

- ◆ Priority
 - High, Medium, Low
- ◆ Stability
 - How subject to change is this requirement
- ◆ Difficulty
 - High, Medium, Low
- ◆ Risk
 - High, Medium, Low
- ◆ Revision
- ◆ Author
- ◆ Current Status
 - TBD - Not defined yet
 - TBR - Preliminary value not reviewed yet
 - Defined - Final value
 - Approved - Review by team
 - Verified - By inspection, analysis, demonstration, or test
 - Deleted - Requirement no longer applies to system

Making Traceability Links

- ◆ Need to trace changes made throughout development life-cycle back to requirements
- ◆ Parent-child relationships
 - Establishing dependencies
 - Directional link between requirements
 - Linking requirements to other documents
- ◆ Traceability matrix is very helpful
 - Traced from and Traces to relationships
 - Visual queue of suspect links is very helpful

Trends in Requirements Management

- ◆ More powerful tools
- ◆ Tools that allow modeling and facilitation of the Requirements Management Process in the context of Organizational Processes
- ◆ Incorporation into database of design, implementation, and test artifacts for traceability of and to requirements

Requirements Verification and Validation

- ◆ Ability to simulate models at higher levels of abstraction
 - Organizational agreement processes
 - Interfaces to the system
 - Domain processes
 - ◆ Use cases/ operational processes
 - ◆ Allows for better understanding of the problem
- ◆ Automated metrics at a higher level contribute to quality of requirements
 - Completeness of Conceptual Models
 - Completeness of Requirements Specifications
- ◆ Real-time monitors of system behavior during test
 - Automatically traces back to requirements specification
 - ◆ Requires formalized specification of requirements

Metric Example: Concept Model Completeness

- ◆ Indication of requirements completeness
- ◆ How well resolved are the models?
- ◆ Are the static and behavioral views consistent?
- ◆ Not making a judgment about the contents of the stakeholder models
 - Requirements engineer is not the domain expert
 - Based on sorting and organizing to evolve an architecture of the system information provided

Broad Requirements Issues

- ◆ Product families and/or System-of-Systems
 - How does the requirements elicitation, analysis, specification, management, and verification & validation effort change?
- ◆ COTS and requirements
 - Proceeding top-down in requirements development while incorporating large-scale COTS technology
- ◆ Technology transfer of requirements engineering techniques
 - Scope and organizational issues
- ◆ Standards analyzed and specified as requirements

Standards As Requirements

- ◆ People reference standards because they need to comply with expected behavior
 - Example: a hard drive needs to comply with expected SCSI standards, those standards become requirements for the designers of the drive
- ◆ Standards documents need to be specified as well-written requirements documents
 - They impact the organization, which is a system
 - Understanding them should not be an impediment
 - Standards that result in repeatable behavior should be the goal of the standards designer
- ◆ Ontology is critical for standards
 - When most people read standards the expert is not there
 - When differences in wording exist they assume there is a reason
- ◆ Begin standards development with models as you would systems development