

Requirements Management: Necessary But Not Sufficient For System Engineering

James E. Long

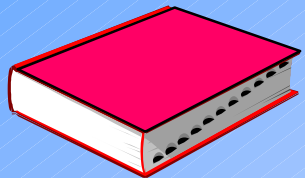
INCOSE/IEEE Meeting

September 25, 2001

Requirements Management Process: Its Inputs and Its Outputs

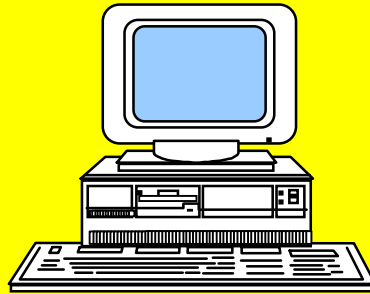


Keyboard Entry,
Data Extractors,
& Parsers



Source Documents/ Other
Requirements Sources

Requirements
Management
Process

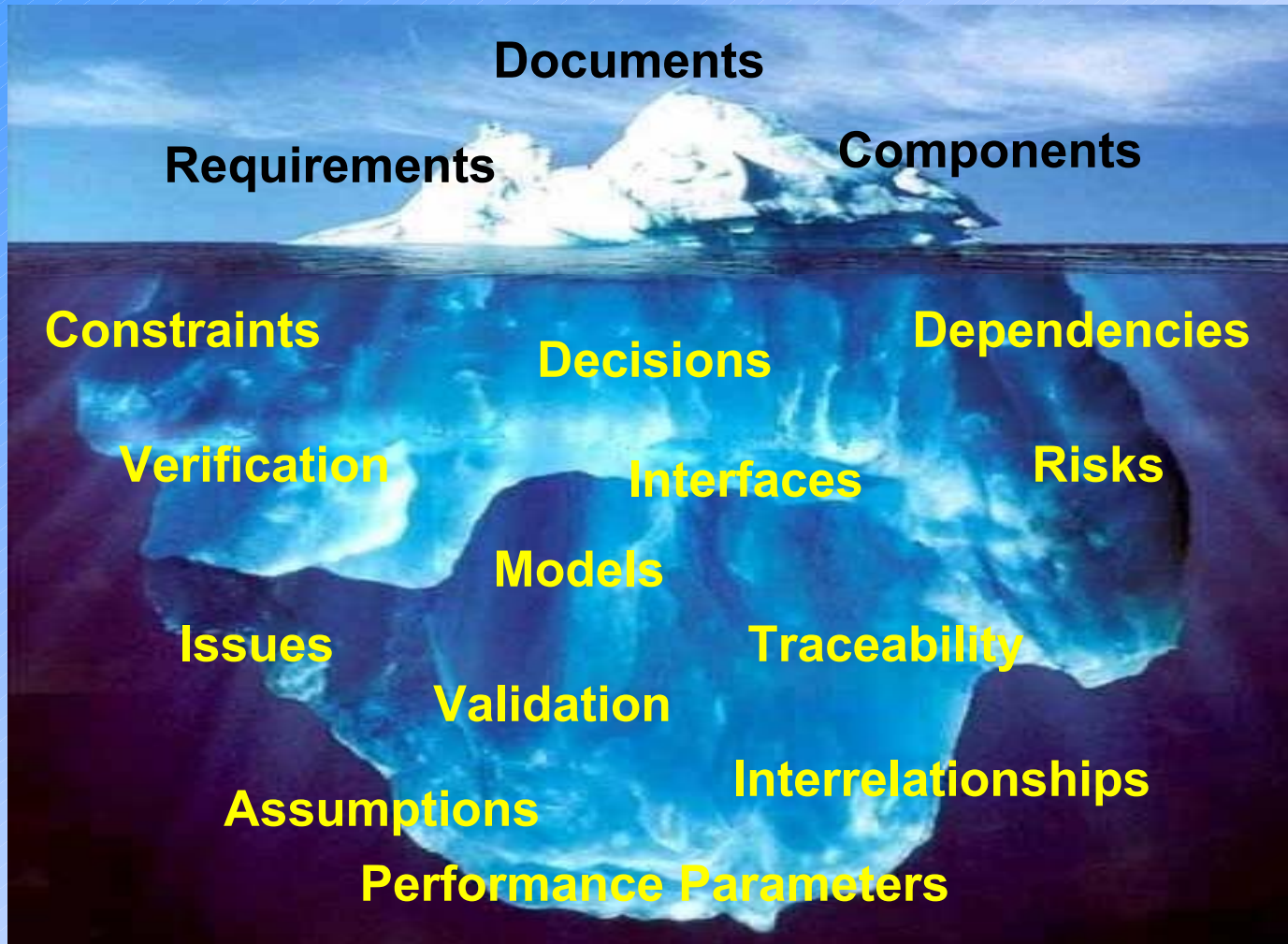


Data Repository

SYSTEM REQUIREMENTS

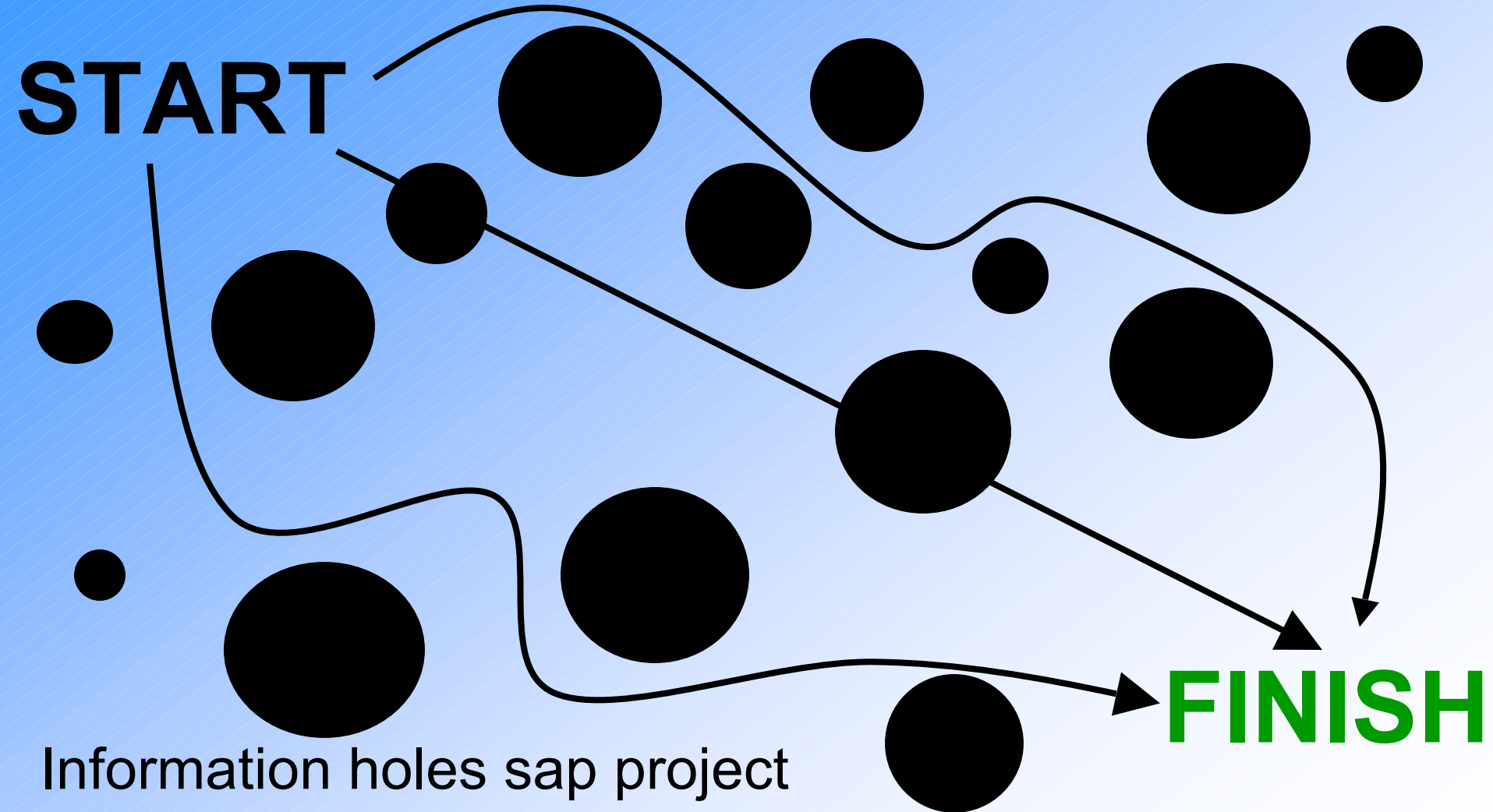
- Complete
- Consistent
- Concise
- Correct
- Feasible
- Traceable
- Unique
- Verifiable
- Unambiguous
- Understandable
- Design Independent

The System Development Challenge



100,000+ pieces of information!

Knowledge Gaps are Deadly



Information holes sap project schedule, resources, and quality

How Do We Find the Holes?

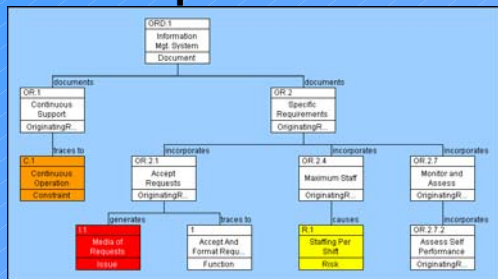
- Process
 - Model-Driven
 - Complete and traceable
 - Onion model
 - Verification of requirements attributes
- Visualization/Views
 - Behavior
 - Physical
 - Interfaces
- Database Queries

Proven Techniques for Assessment and Verification of Requirements

<u>DESIRED ATTRIBUTE</u>	<u>VERIFICATION METHOD</u>
Requirements are: Complete Consistent Testable	<ul style="list-style-type: none">•Behavior Analysis•Enhanced Function Flow Block Diagrams (EFFBDs)
Consistent Interface Requirements	<ul style="list-style-type: none">•EFFBDs•N2 (Interface) Diagrams
All Requirements Have Been Allocated	<ul style="list-style-type: none">•Requirements Traceability
Compatible Physical Architecture	<ul style="list-style-type: none">•Functional Simulation•Design Tradeoffs
Viability of Subsystem Design Concept	<ul style="list-style-type: none">•Functional Simulation

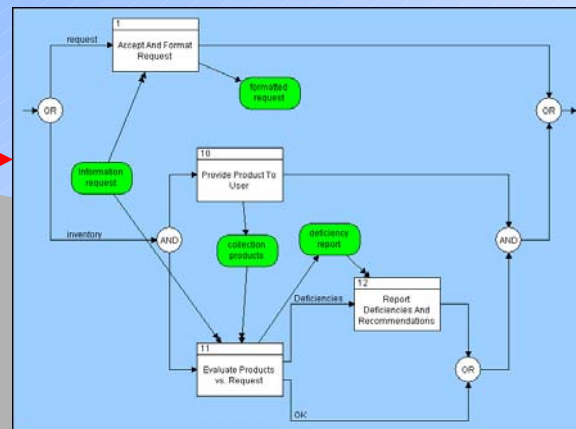
Consistency through Integrated Life-Cycle Knowledge Management

Source Requirements Domain



Originating requirements trace to behavior

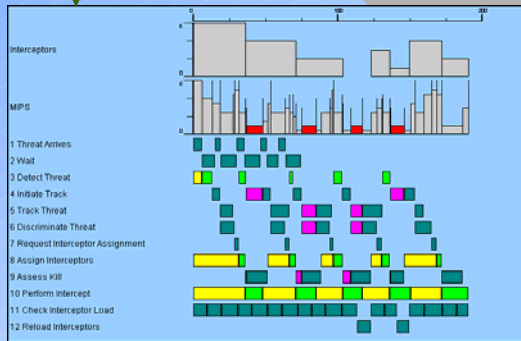
Behavior Domain



Behavior is allocated to physical components

System Development Repository

V&V Domain

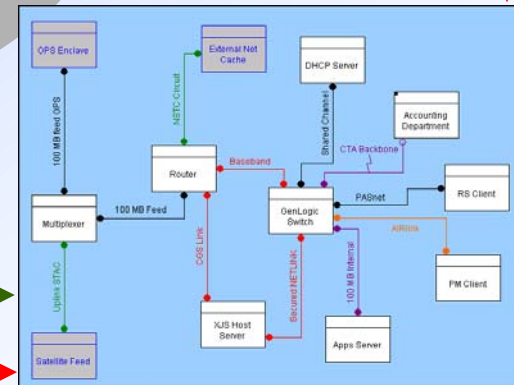


verified by

verified by

verified by

Architecture Domain

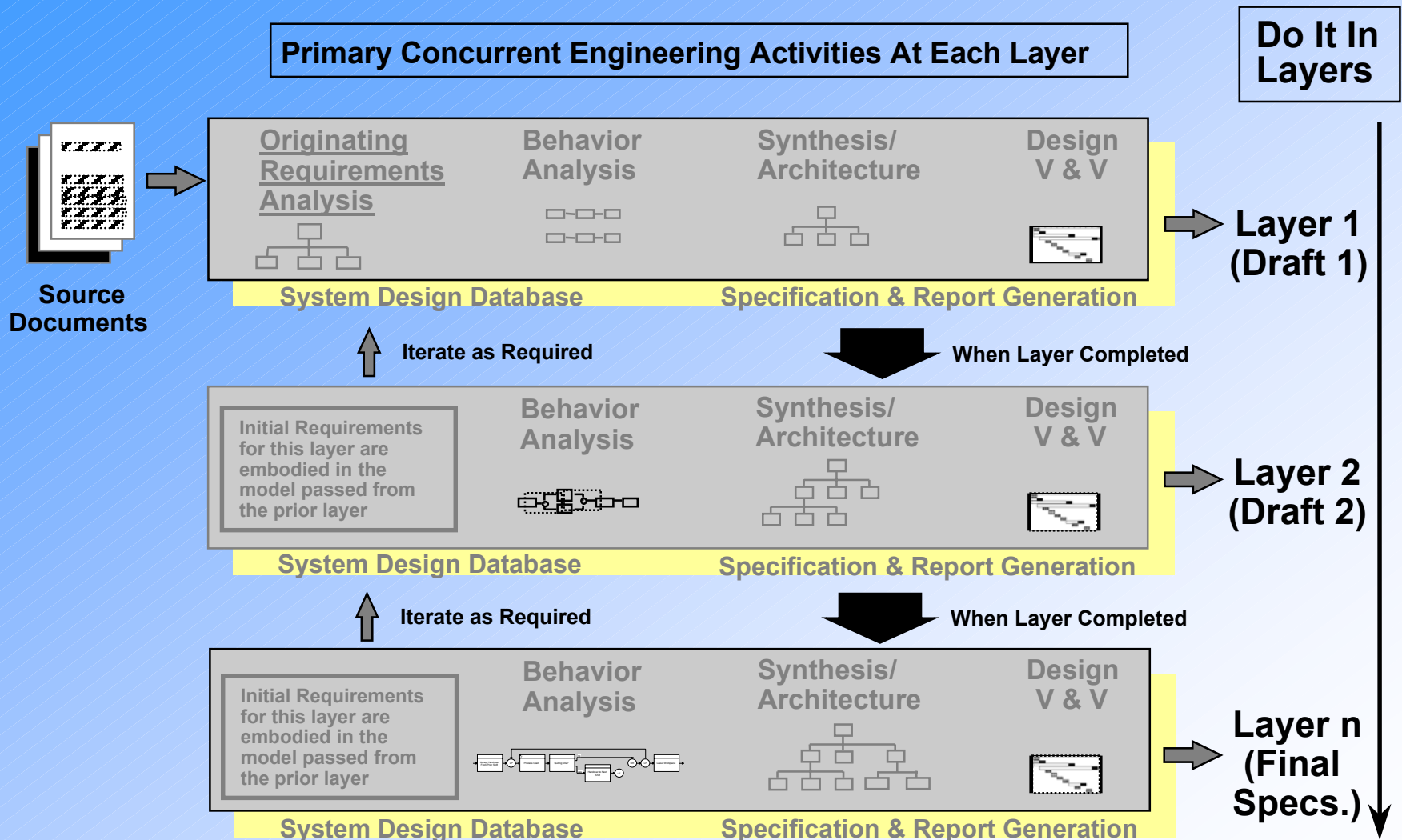


Originating requirements trace to physical components

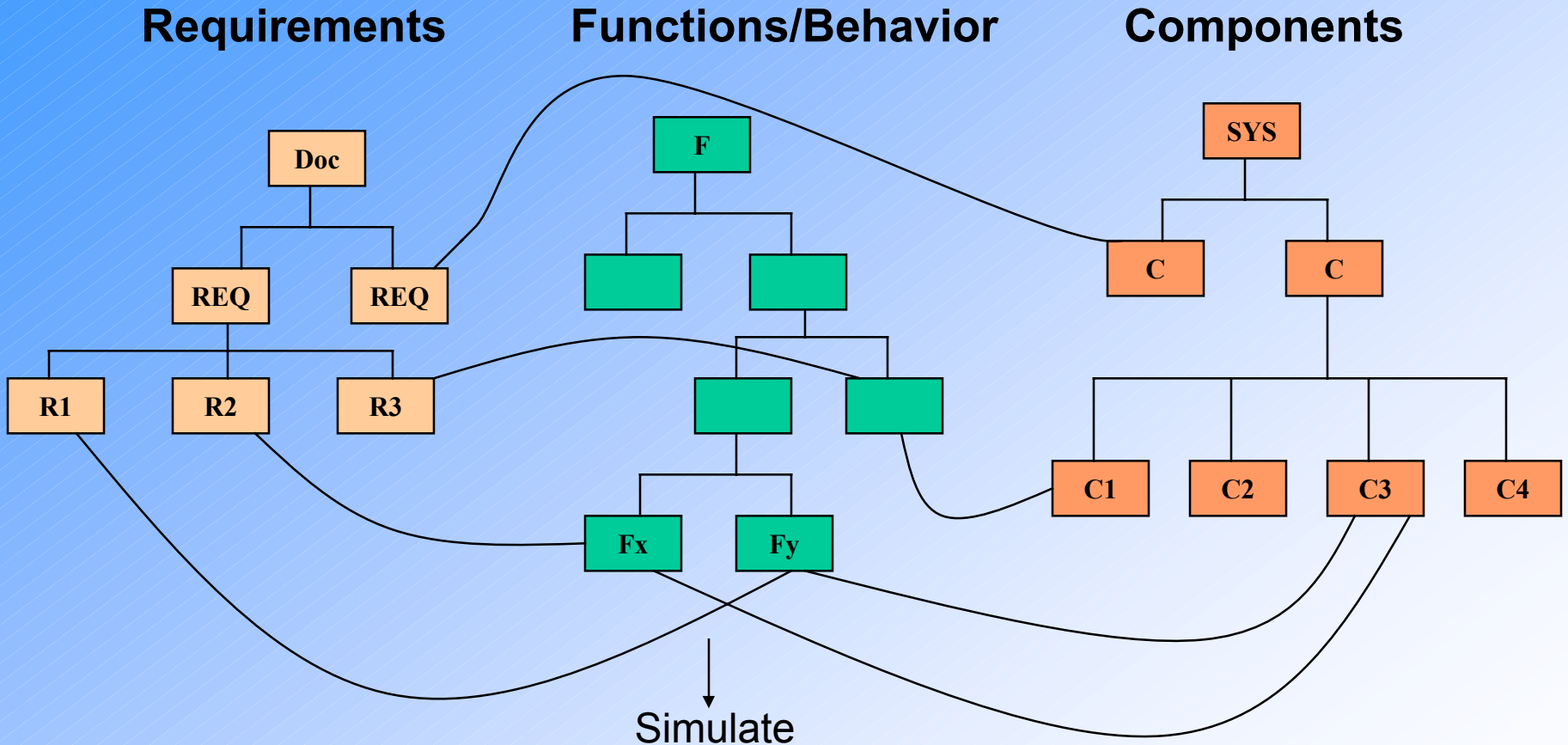
System Engineering Process – the Onion Model.

Do Your System Engineering in Increments/Layers.

Completeness and Consistency at Each Layer



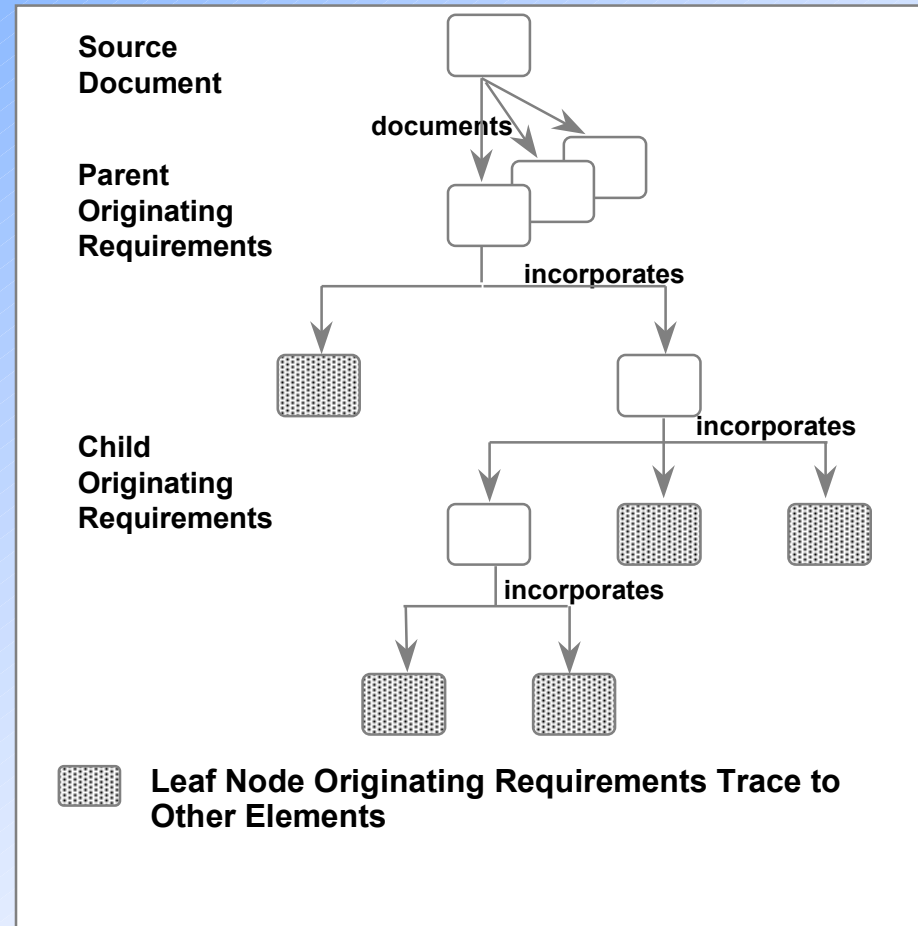
Traceability within Models



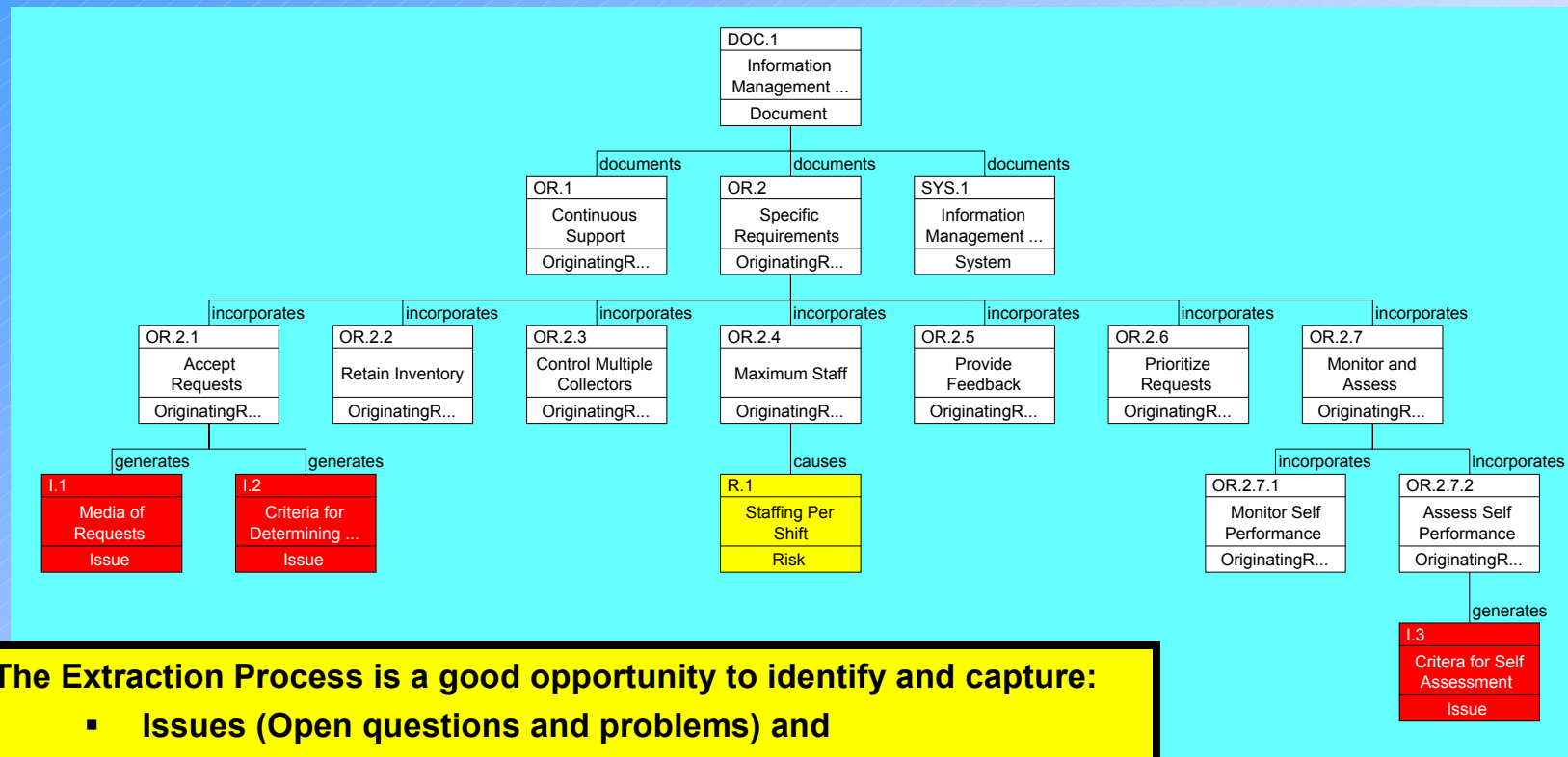
The combination of parent-child and peer-peer relationships provides complete design traceability.

A Requirements Capture Approach

- **Objective: Get to the leaf nodes – extract Originating Requirements into single, testable statements.**
- **Record source requirement statement in the description attribute of Originating Requirement.**
- **Obtain traceability from source, top level, and all child level originating requirements.**
- **Extract by parsing - using automatic, semiautomatic, or manual techniques.**
- **Be disciplined. In most cases you are trying to determine the true problem, not invent your own.**



Requirements Capture is an Opportunity to Incorporate Issues and Risk



The Extraction Process is a good opportunity to identify and capture:

- Issues (Open questions and problems) and
- Risks

and link them to the relevant Originating Requirements

An Originating Requirement is More Than a Text Statement (Attributes, Relationships, and Targets)

Accept Requests asERA (Demo)

File CORE Edit Diagram Settings Positions Views Help

OR.2.1
Accept Requests
OriginatingR...

generates

I.1
Media of Requests
Issue

generates

I.2
Criteria for Determining ...
Issue

incorporated in

OR.2
Specific Requirements
OriginatingR...

owned by

System Engineer
Engineer

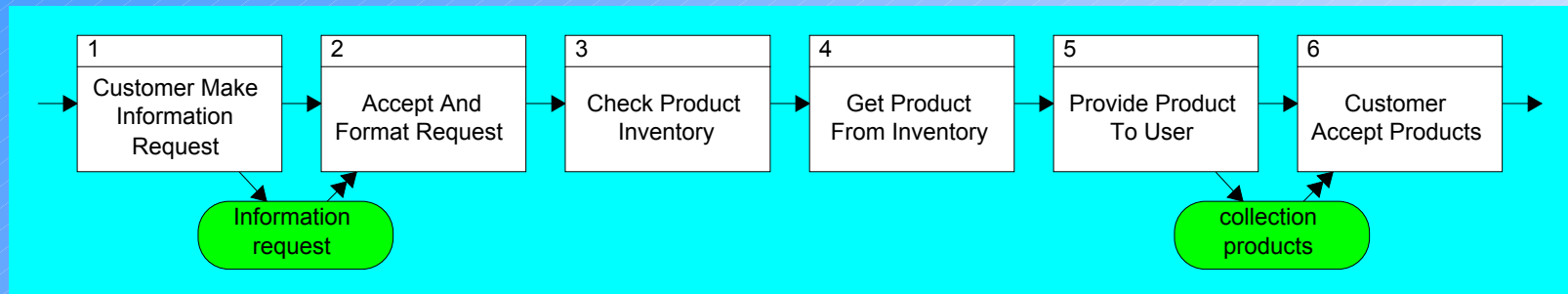
traces to

1
Accept And Format Requ...
Function

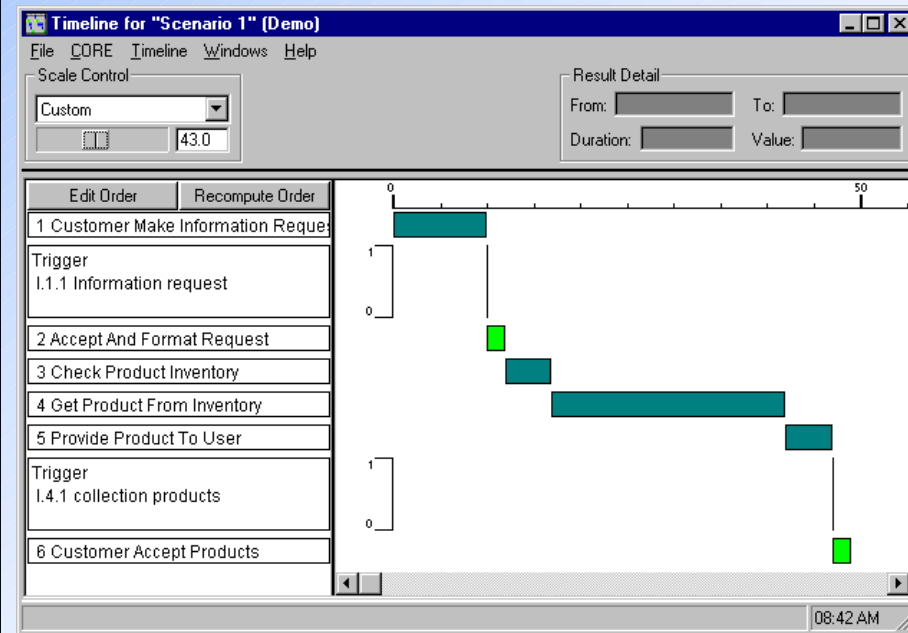
Name:	Accept Requests
Creator:	System Engineer
Created:	Sunday, August 03, 1997 at 11:26:48 AM
Number:	OR.2.1
Abbreviation:	
Description:	The system shall accept information requests from certified customers.
Weight Factor:	
Paragraph Title:	Specific Requirements
Paragraph	3.1
Line Number:	
Last Modified:	Thursday, January 22, 1998 at 09:59:18 AM

RWDA Thursday, January 22, 1998 at 09:59:18 AM 11:33 AM

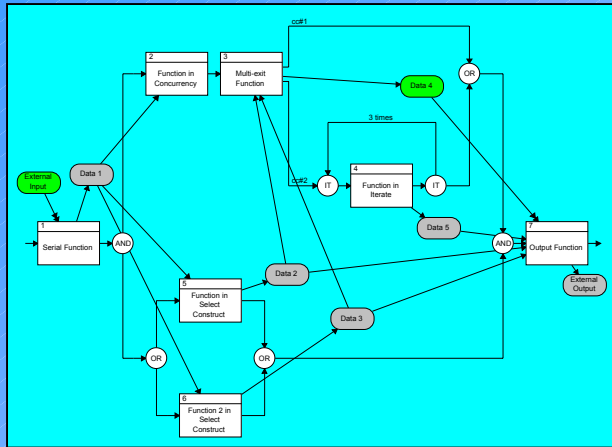
A Powerful Technique for Analyzing Originating Requirements is to Generate System Threads/Scenarios



- Scenario timelines can be analyzed automatically or derived and specified as derived requirements.
- Easy mechanism to identify functional holes and discontinuities. Capability to define functional requirements.
- N2 Chart identifies Interface inconsistencies.
- System level scenarios are the start of requirements validation and analysis by EFFBDs and simulation.

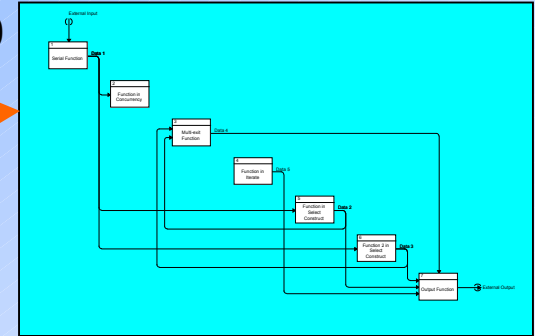


The Many Faces of Behavior

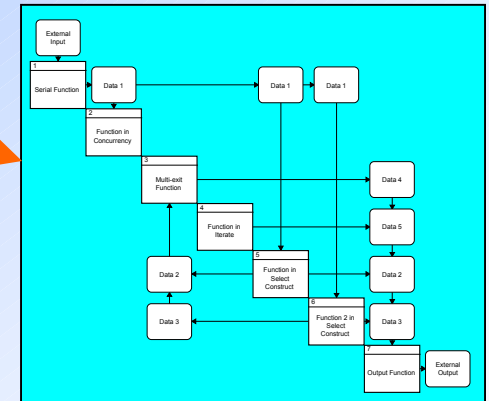


EFFBD

IDEF0



N2 Chart



Text

Constructs as Text (Default)

File Edit Element Target Views Options Help

Name: Constructs
 Number: 0
 Abbreviation: Con
 Description: This is a sample to show the different views available from CORE
 Doc. PUID: ABC.123
 Duration: Normal (μ: 10.0, stdDev: 1.0, stre Edk)

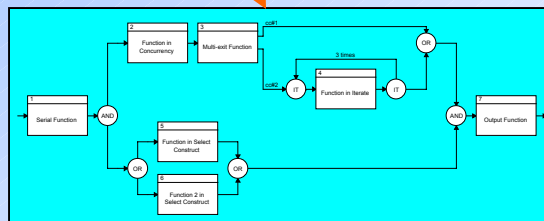
Relationships: causes, constrained by, consumes, decomposed by, decomposes, documented by, exhibits

Targets & Attributes: Function 1 Serial Function, Function 2 Function in Concur..., Function 3 Multi-exit Function

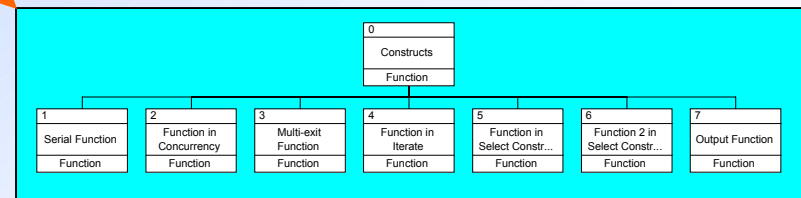
Sort: Numeric by class

RWDA August 20, 2001 at 07:01:36 AM 07:01 AM

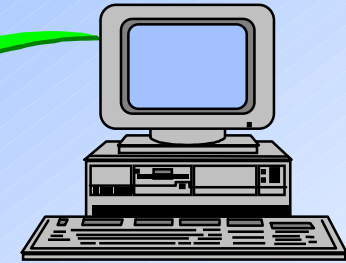
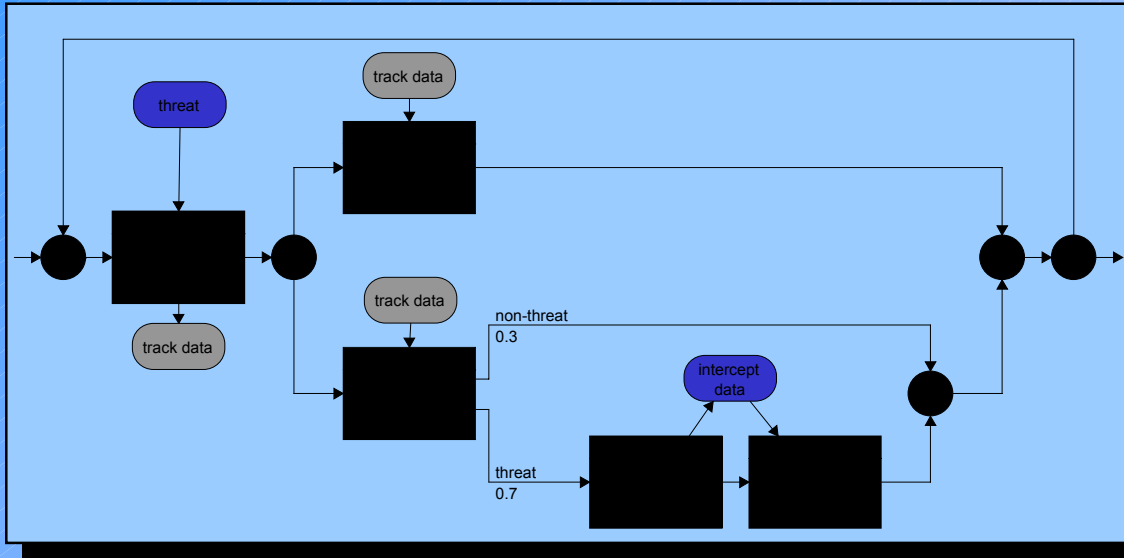
FFBD



Hierarchy



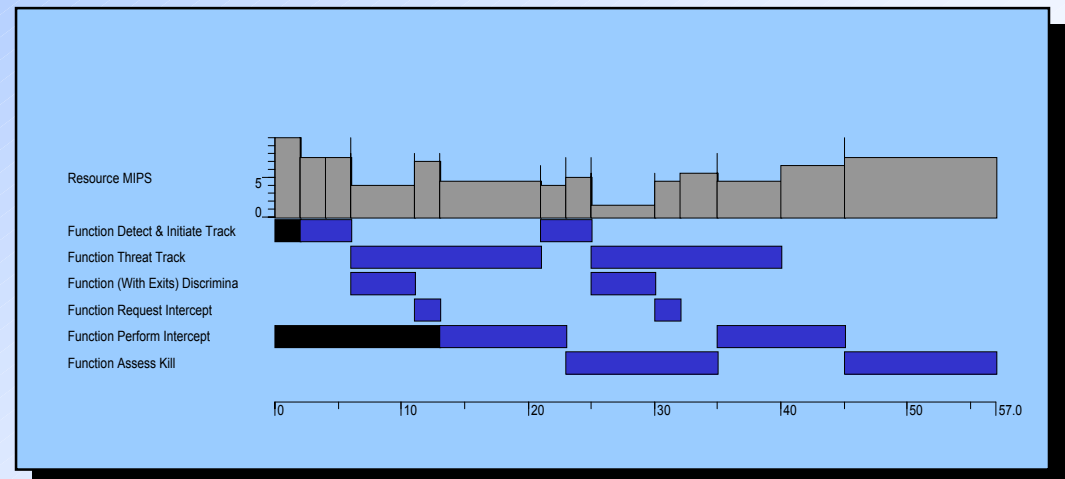
Executable Behavior: From Behavior to Timelines, Resources, Queues, and Flow Requirements



(Repository)

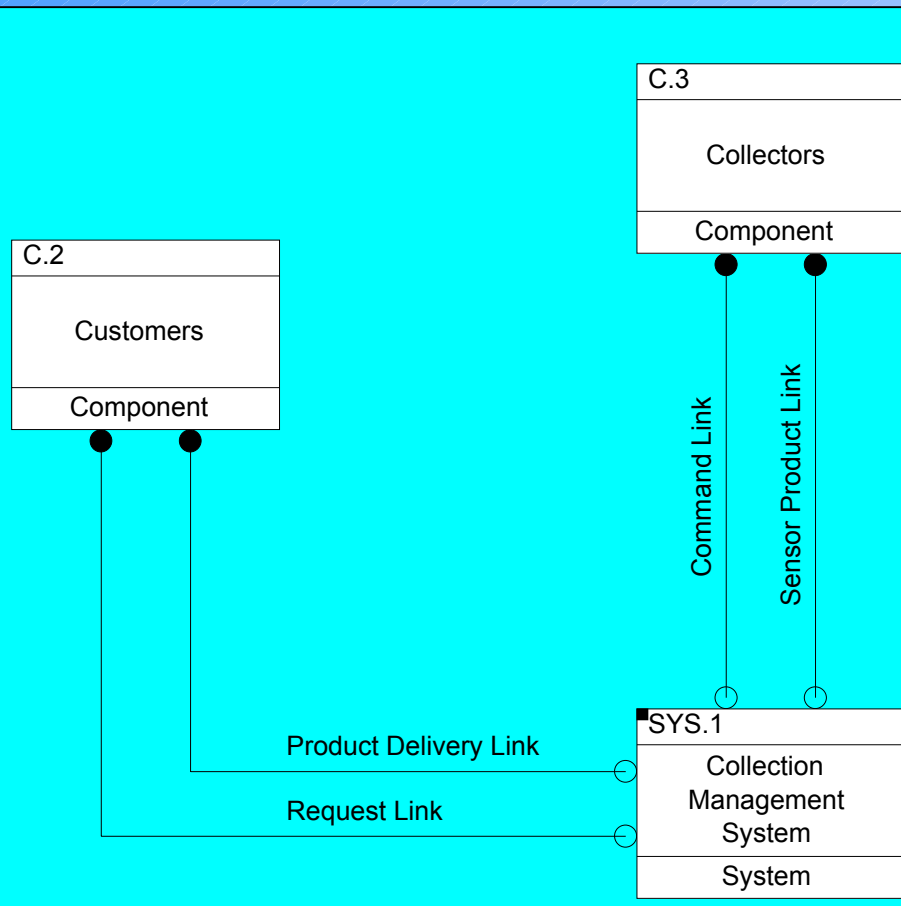


- **Assess dynamic consistency**
- **Analyze timelines**
- **Analyze resource models and dynamics**
- **Assess queuing dynamics**

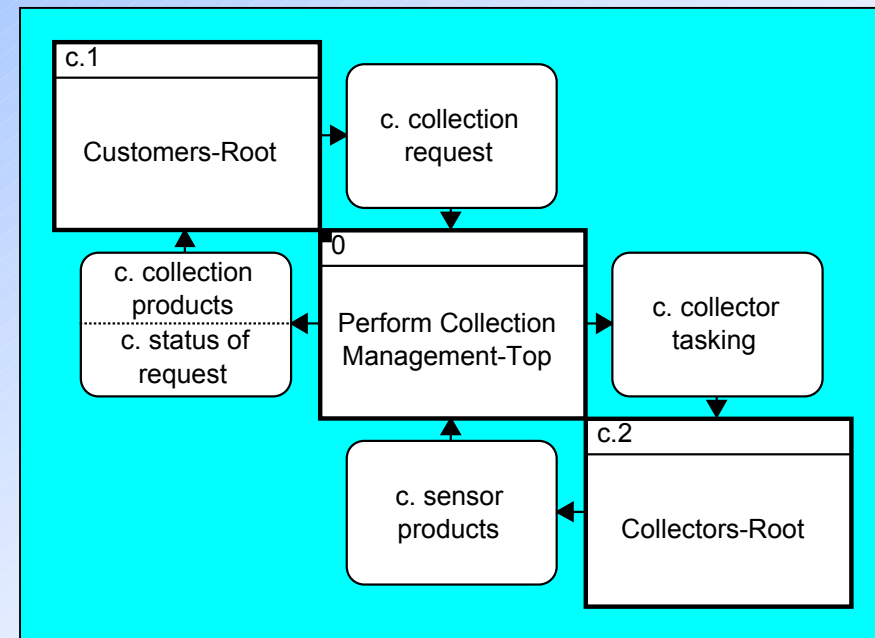


Physical and Interface Diagrams – Visual Indicators of Completeness

Physical Block Diagram

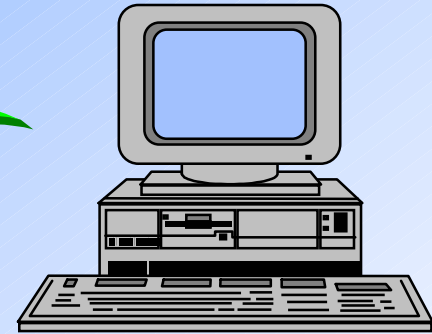
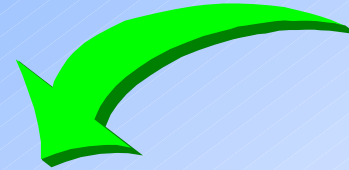


N2 Interface Diagram

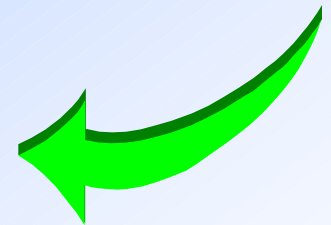


Database Queries Can Find Holes

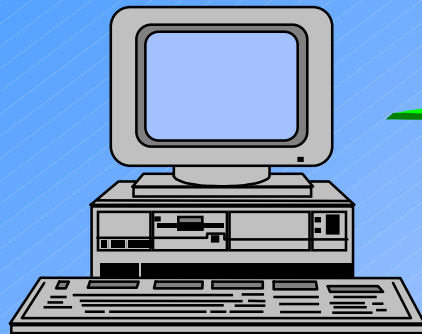
- Inputs with no source
- Outputs with no destination
- Unverified leaf-level requirements
- Unallocated leaf-level requirements
- Unallocated leaf-level functions
- Interfaces that don't carry anything
- Interfaces that aren't completely connected
- Issues with no decisions
-



(Repository)

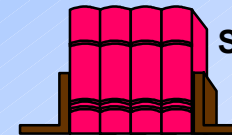


A Key Piece of Requirements Management is Communication



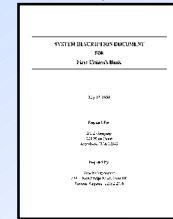
(Repository)

Data Interchange Files

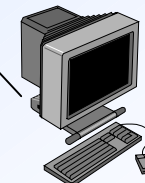
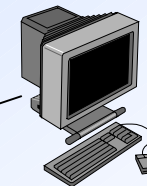


Formal Specifications

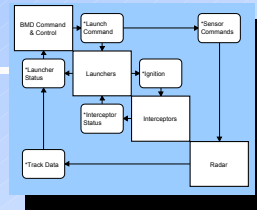
Custom Reports and Queries



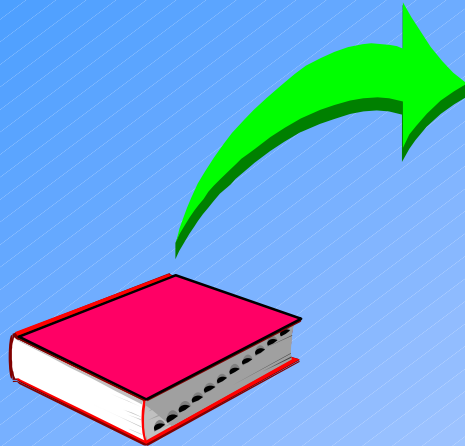
- Automatically extracted documents from the repository in standard format for soft and hard copies
- Integrated, distributed teams need web-based products



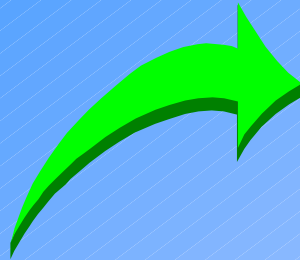
Web browser access for collaboration



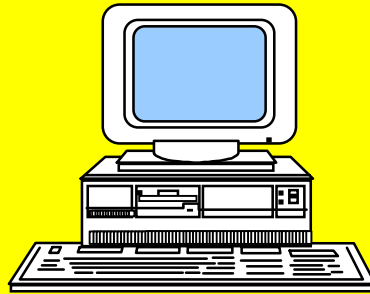
Overview: The Requirements Management Process



Source Documents/ Other
Requirements Sources



Requirements
Management
Process



Data Repository



SYSTEM REQUIREMENTS

- Complete
- Consistent
- Concise
- Correct
- Feasible
- Traceable
- Unique
- Verifiable
- Unambiguous
- Understandable
- Design Independent

Lastly, a Reminder:

**Requirements
Management**

≠

**System
Engineering**

Contact Information

James E. Long

jlong@vtcorp.com



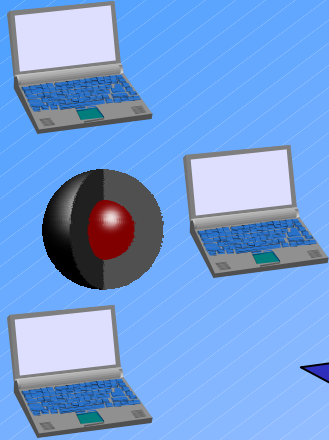
**2070 Chain Bridge, Ste 320
Vienna, VA 22182-2536
Phone: (703) 883-2270**

**FAX: (703) 883-1860
E-mail: info@vtcorp.com
Web: www.vtcorp.com**

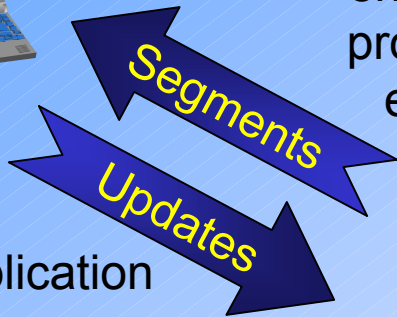
Additional Slides

IPT Deployment Architecture

Workstation

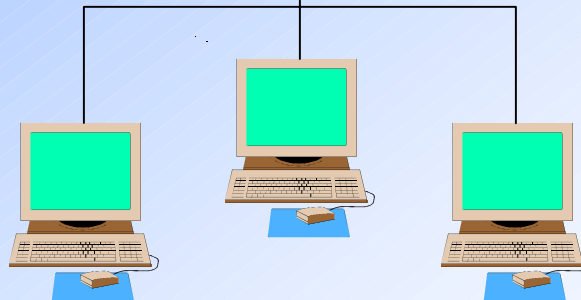
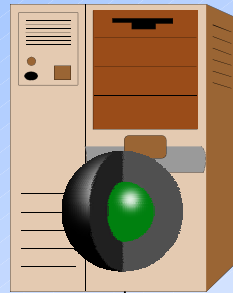


Integrated application for small teams or on-site modeling/review

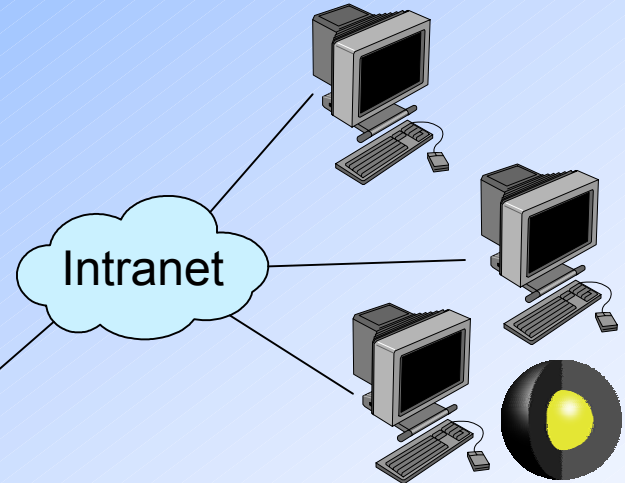


Enterprise

Collaborative environment at the product, project, or enterprise level



CORE2net



Web browser access enabling collaboration at the enterprise level and beyond

Document-Driven vs. Model-Driven Design

- Document-Driven Design
 - the current (most common) system engineering practice
 - the system model is informational not normative (i.e., a system model exists to provide a framework for discussing, identifying, tracking, or illustrating requirements; it is not mandatory)
 - the focus is on requirements document production
 - the system specifications are normally incomplete and inconsistent
 - the subsequent engineering teams reverse engineer the specification to derive the model (a primary waste of time and money)
- Model-Driven Design
 - a system engineering paradigm shift
 - the system model is normative (i.e., it is required)
 - the system model encompasses the system design and specification
 - the system specifications are complete and consistent
 - the model is provided to subsequent engineering teams