Characteristics of Model-Based Systems Engineering

Warren B. Smith
wsmith@vitechcorp.com
480.560.2655 Cell
Vitech Corporation

A Roadmap for Today

- A Brief Introduction to Systems Engineering
- Model-Based Systems Engineering – MBSE

Please raise questions and offer perspectives as they occur!

A (very) Brief Introduction to Systems Engineering

System Essentials

> A system is a construct or collection of different elements that together produce results not obtainable by the elements alone.
>
> INCOSE

- A **System** *'is a thing that contains interconnected smaller things, interacts with other things in a larger thing, and does something.'*
  - has structure, in a larger structure ('physical' context)
  - performs purposeful actions
    - time sequence of collective actions = behavior
  - behavior observed at interfaces ('functional' context)

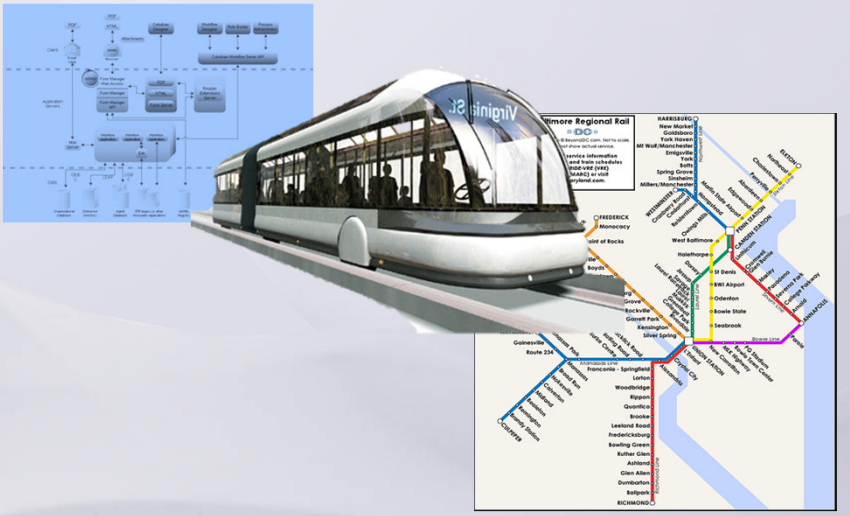
What is Systems Engineering?

Systems Engineering is an engineering discipline whose responsibility is creating and executing an interdisciplinary process to ensure that the customer and stakeholder's needs are satisfied in a high quality, trustworthy, cost efficient and schedule compliant manner throughout a system's entire life cycle.

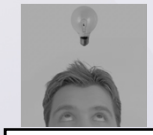
INCOSE

Model-Based Systems Engineering

Three Systems of Interest



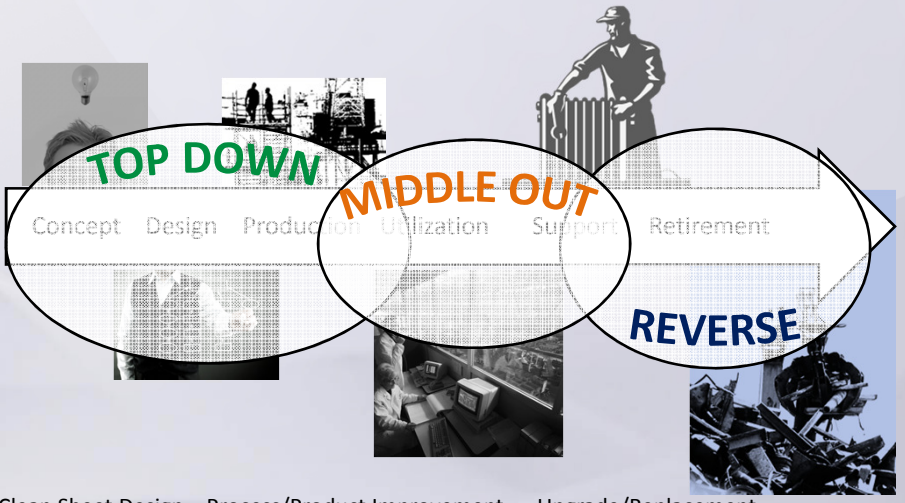
System Life Cycle and Problem Classes



Concept Design Production Utilization Support Retirement

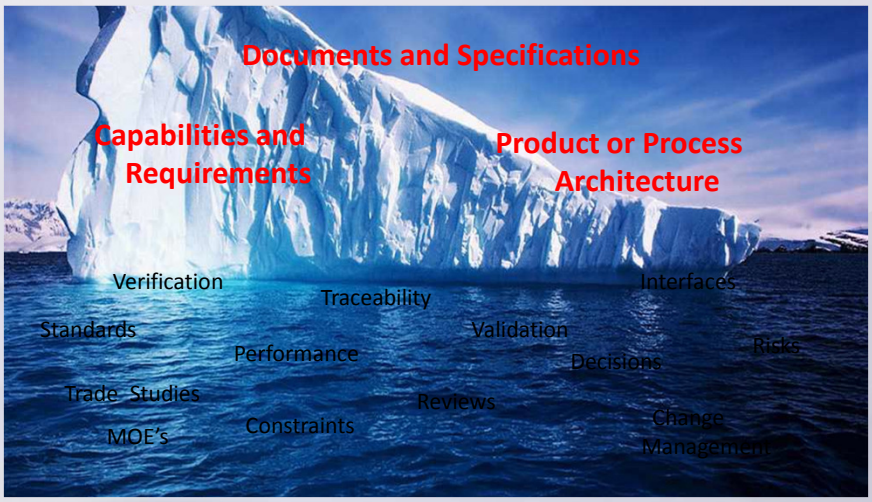
Clean Sheet Design Process/Product Improvement Upgrade/Replacement

System Life Cycle and Problem Classes

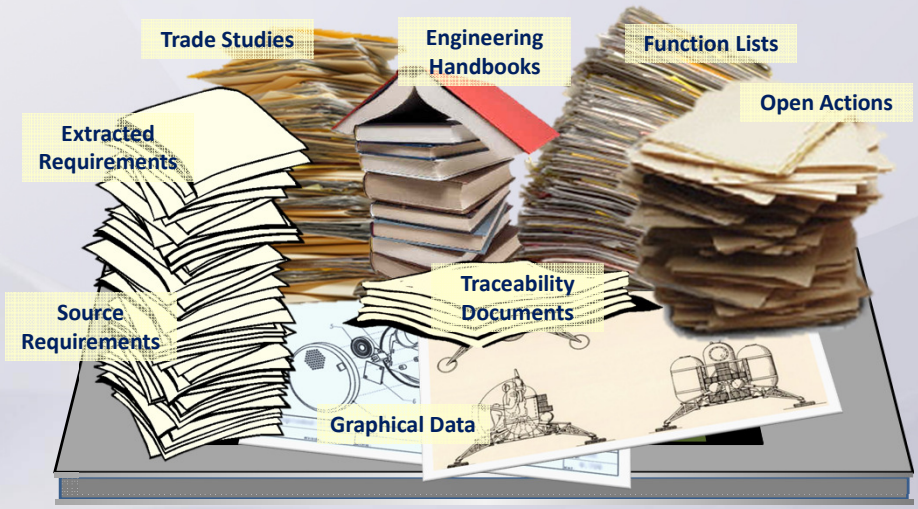


Clean Sheet Design Process/Product Improvement Upgrade/Replacement

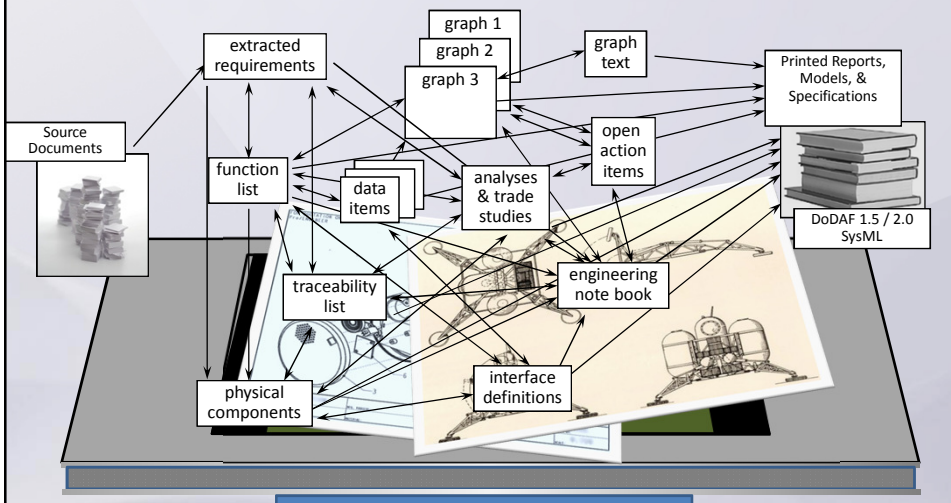
The Hidden Complexity of System Engineering



Systems Engineer's Dilemma: Complexity and Synchronization



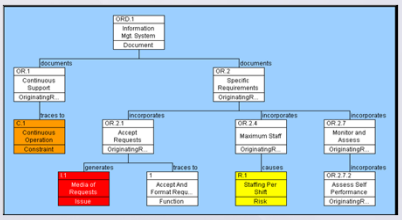
Systems Engineer's Dilemma: Complexity and Synchronization



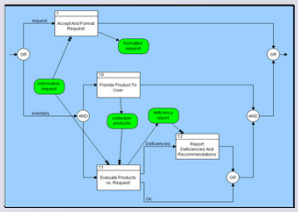
Any change will affect something else

Setting the Context: The Four Primary SE Activities

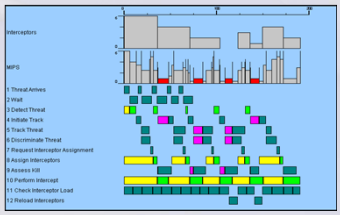
Source Requirements Domain



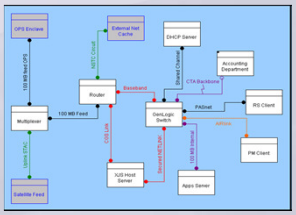
Behavior Domain



Test & Evaluation Domain

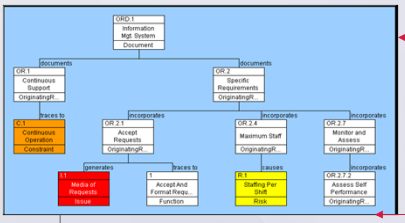


Architecture Domain



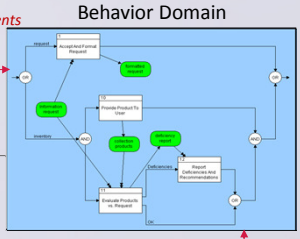
Setting the Context: The Four Primary SE Activities

Source Requirements Domain



Originating requirements trace to behavior

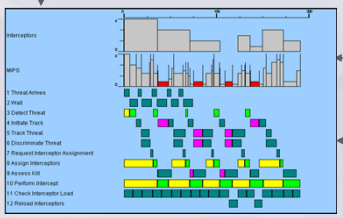
Behavior Domain



verified by

Originating requirements trace to physical components

Test & Evaluation Domain

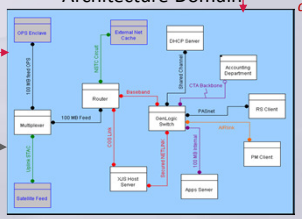


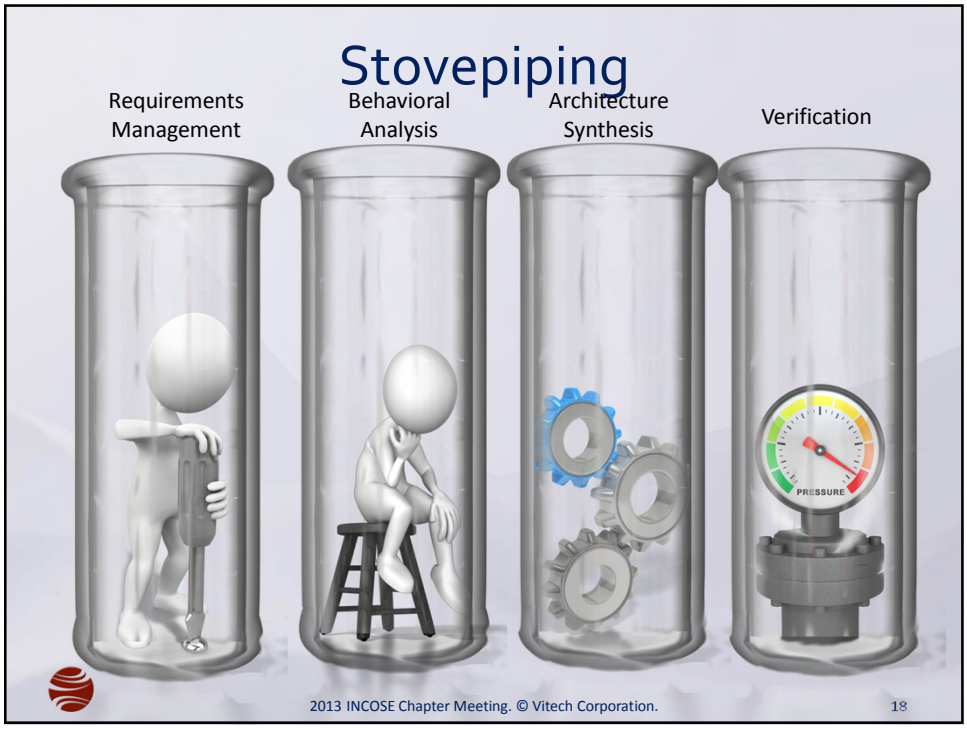
verified by

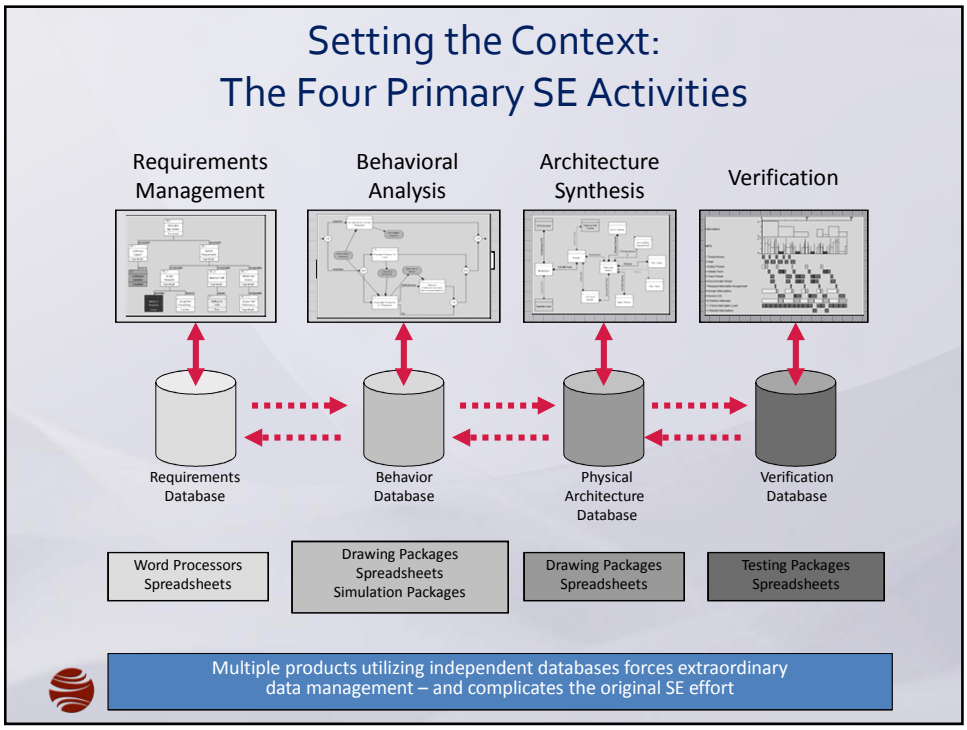
verified by

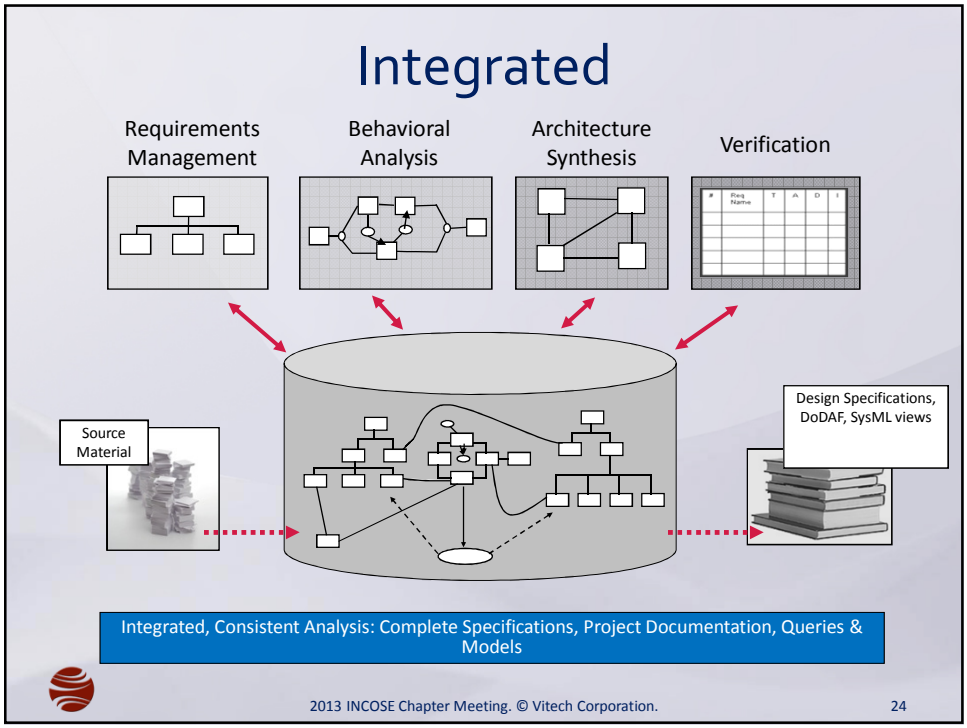
Behavior is allocated to physical components

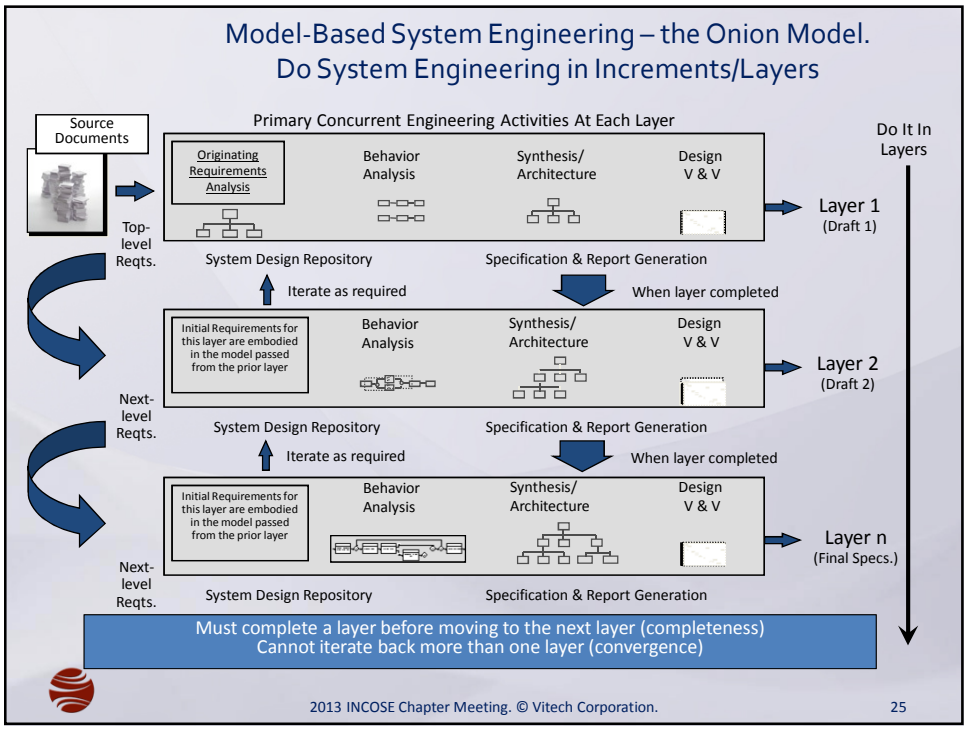
Architecture Domain




Stovepiping
Requirements Management
Behavioral Analysis
Architecture Synthesis
Verification
PRESSURE
2013 INCOSE Chapter Meeting. © Vitech Corporation.
18

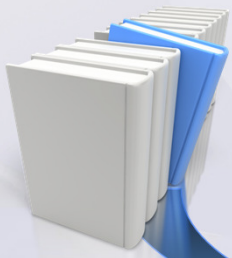

Setting the Context:
The Four Primary SE Activities
Requirements Management
Behavioral Analysis
Architecture Synthesis
Verification
Requirements Database
Behavior Database
Physical Architecture Database
Verification Database
Word Processors
Spreadsheets
Drawing Packages
Spreadsheets
Simulation Packages
Drawing Packages
Spreadsheets
Testing Packages
Spreadsheets
Multiple products utilizing independent databases forces extraordinary data management – and complicates the original SE effort


Integrated
Requirements Management
Behavioral Analysis
Architecture Synthesis
Verification
Source Material
Design Specifications, DoDAF, SysML views
Integrated, Consistent Analysis: Complete Specifications, Project Documentation, Queries & Models
2013 INCOSE Chapter Meeting. © Vitech Corporation.
24


Model-Based System Engineering – the Onion Model.
Do System Engineering in Increments/Layers
Source Documents
Primary Concurrent Engineering Activities At Each Layer
Do It In Layers
Originating Requirements Analysis
Behavior Analysis
Synthesis/ Architecture
Design V & V
Layer 1
(Draft 1)
Top-level Reqts.
System Design Repository
Specification & Report Generation
Iterate as required
When layer completed
Initial Requirements for this layer are embodied in the model passed from the prior layer
Layer 2
(Draft 2)
Next-level Reqts.
Layer n
(Final Specs.)
Must complete a layer before moving to the next layer (completeness)
Cannot iterate back more than one layer (convergence)
2013 INCOSE Chapter Meeting. © Vitech Corporation.
25

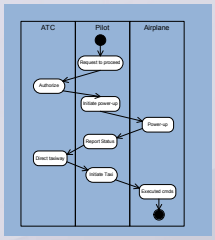
Systems Engineering: A Practice in Transition

Traditional



- **Specifications**
- **Interface requirements**
- **System design**
- **Analysis & Trade-off**
- **Test plans**

Future



Moving from document-centric to model-centric

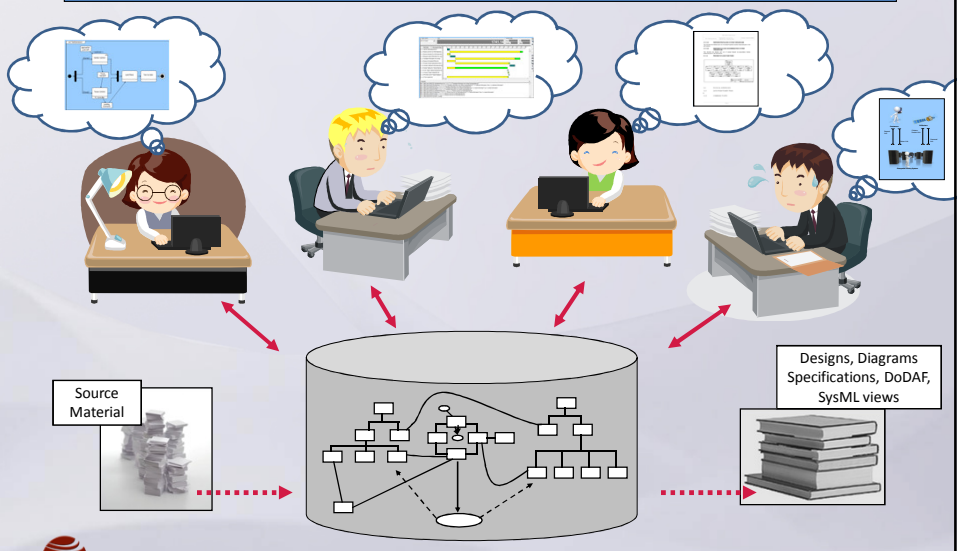
Reprinted from INCOSE Model-Based Systems Engineering Workshop, February 2010

Characteristics of Model-Based Systems Engineering

- **Model-Centric, not Diagram-Centric**
  - An Underlying Model of the System
  - Common Repository
  - Shared with team members
  - One version of the Truth
- **Unambiguous Notation**
  - One version of the Truth
- **Views generated from the model**
  - Consistency maintained as Changes occur
  - Views are tailorable to Audience needs
- **Complete, Query-able, Virtual System Prototype**
  - Handles all aspects of Systems Engineering
  - Provides “Defect-Finding” capabilities
  - Act as a Virtual Prototype of System

Model-Centric, not Diagram-Centric

Integrated, Consistent Analysis: Complete Specifications, Project Documentation, Queries & Models

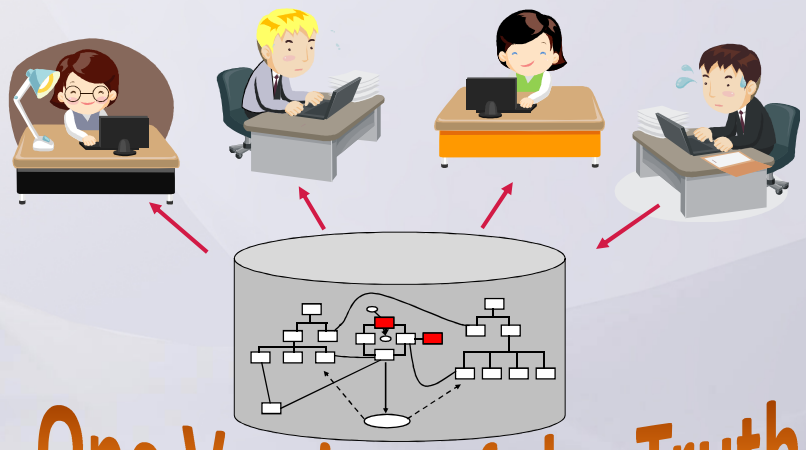


...But don't we draw Diagrams?

- Yes! And the model is populated when Diagrams are created or modified.
- The model is *also* populated from updated:
  - Property Sheets
  - Tables
  - Imported data
- All changes are instantly propagated to the team
- With SySML (and other views) *generated* from the model (more on this later)

Model-centric, not Diagram-centric

Integrated, Consistent Analysis: Complete Specifications, Project Documentation, Queries & Models



One Version of the Truth

Ambiguous Notation

The Plague of Vague

- To manage complexity, Systems Engineers own Interfaces and Relationships
- But yet… Ambiguous Notation remains!
  - Ambiguity wastes time and money
  - Humans easily misunderstand each other
  - Interpreted "Truth"
  - Some notation standards do exist, *but…*
    - Part of
    - Kind of
    - Depends on
    - Associated with
    - Extends
    - Includes
  - … common ambiguous examples abound:
    - Requirements *are linked to* something
    - Elements *are associated with* Elements

Ambiguous Notation
The Plague of Vague

- To manage complexity, Systems Engineers own Interfaces and Relationships
- … Ambiguous Notation remains!
  - Ambiguity wastes time and money
  - Humans easily misunderstand each other
  - Interpreted "Truth"
  - Some notation standards do exist, *but…*
    - Part of
    - Kind of
    - Depends on
    - Associated with
    - Extends
    - Includes
  - … common ambiguous examples abound:
    - Requirements *are linked to* something
    - Elements *are associated with* Elements

MBSE Brings Clarity

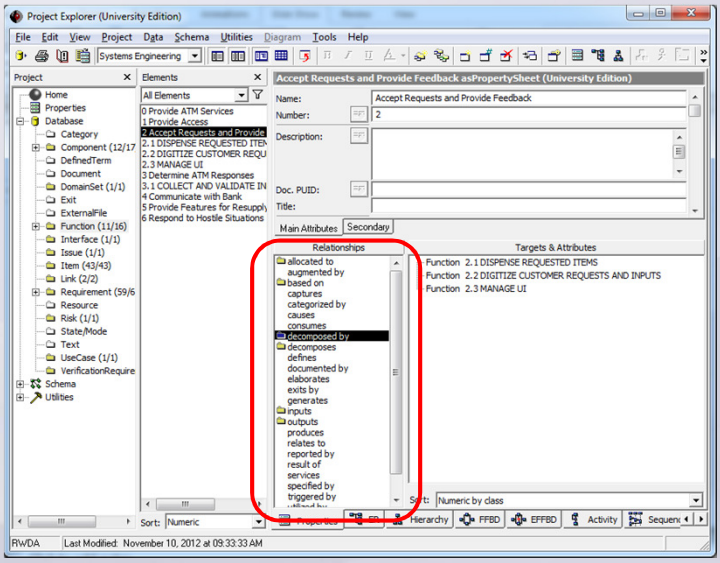
Continuity, not Ambiguity

- Models ensure clarity among the Design Team via an explicit **S**ystem **D**efinition **L**anguage
- Explicitly define relationships:
  - Use Case *extends/includes* Use Case
  - Requirements *refine* Requirements
  - Requirements *specify* Functions
  - Requirements are *verified by* Verification methods
  - Functions are *allocated to* Components
  - Functions *decompose* Functions
  - Functions *consume* Resources
  - Assemblies are *built from* Components
  - Components are *joined by* Interfaces

Continuity, not Ambiguity

- Systems Engineers deal with much more:
  - Interface *is comprised of* links
  - Links *transfers* Data
  - Links *are serviced by* Functions
  - Risk, issue or concern *is assigned to* an Organization
  - Risk is *caused by* a Component, Function or I/F
  - Risk is *resolved by* a Program activity
  - Verification method is *fulfilled by* Verification Event
  - Verification method *verifies* a Function or Requirement

Example in CORE



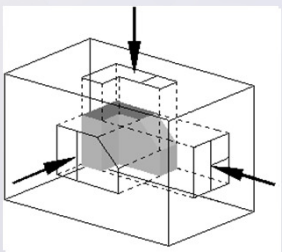
Diagrams, Views and Model

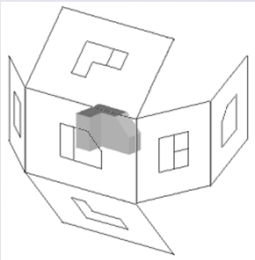
- What are the most common "MBSE" tools?

Yes, MS Office

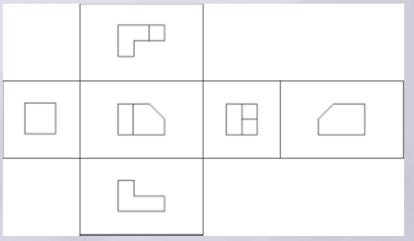
- Don't the Diagrams = the Model?
  - Referential Integrity
  - Virtual Prototype
  - Defect Identification

View and Viewpoints



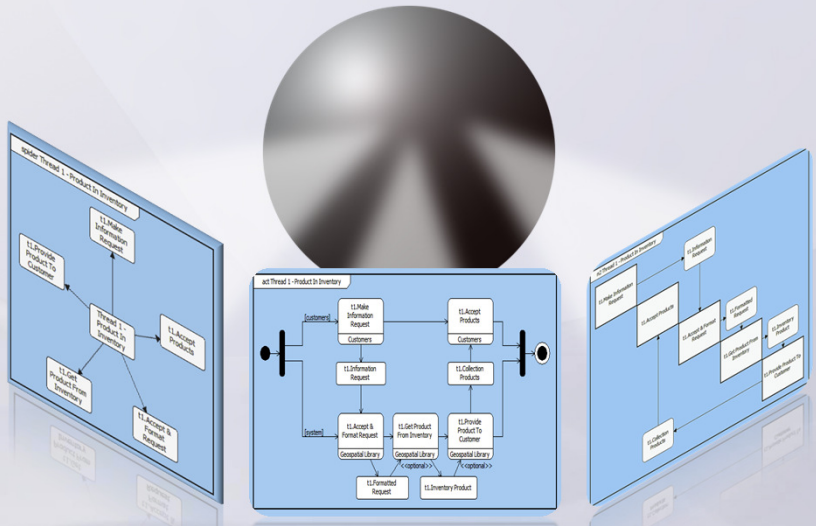


Diagrams are views of the model
Multiple views;
How many viewpoints?

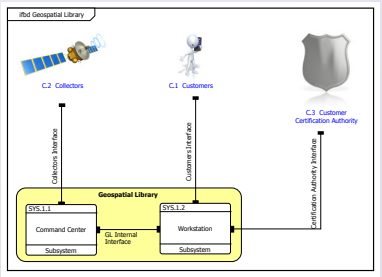


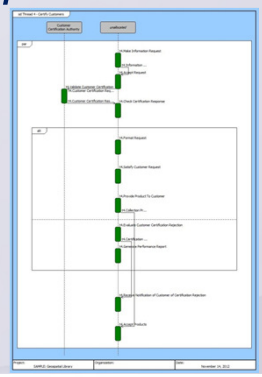
http://en.wikipedia.org/wiki/Multiview_orthographic_projection

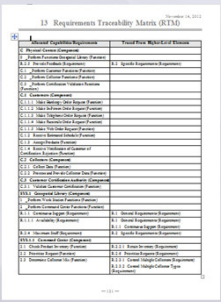
A Consistent View of Views

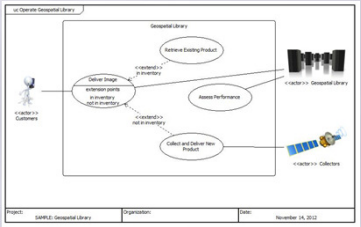


Audience View*points*





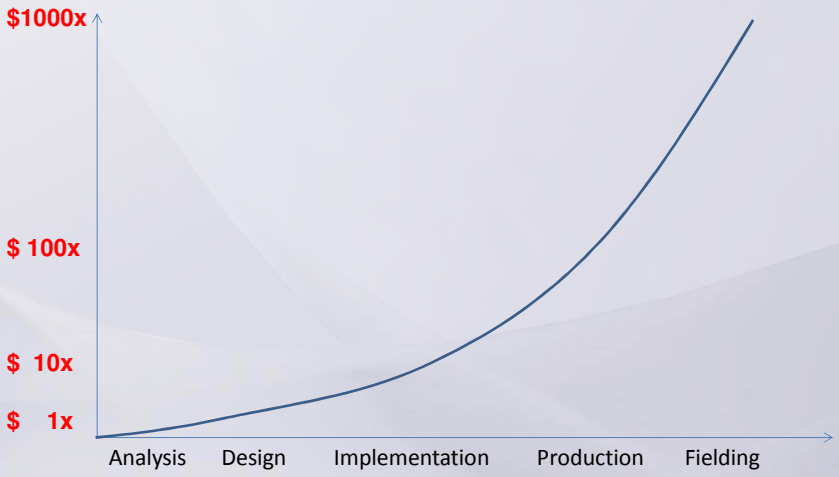




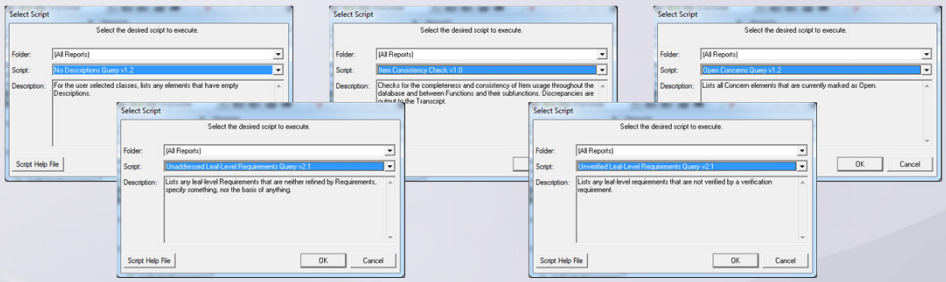
Complete, Query-able and Virtual System Prototype

- Model has
  - Referential Integrity - Diagrams
  - Virtual Prototype
  - Defect Identification

Cost of Correcting Defects

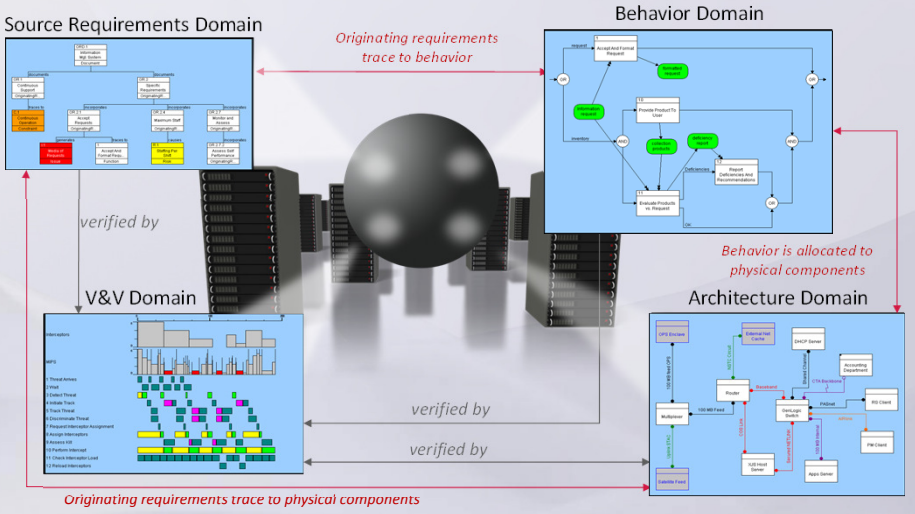


MBSE should provide *early* Defect Identification



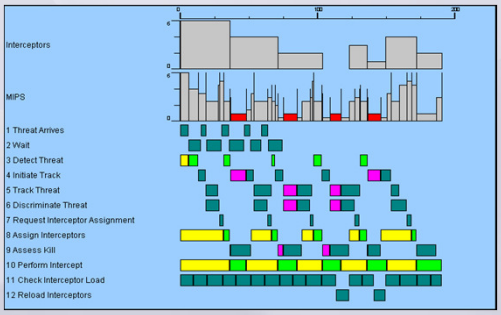
- Input-Output Consistency Checks
- Unaddressed Requirements Checks
- Unverified Requirements Checks
- Unaddressed Concerns Checks
- No Description (documentation) Checks

Complete: One Integrated Model should support *Virtual Prototyping*



Virtual Prototyping

- Simulate your system *from the model*
- Try out different Architectures
- Trade-off various performance parameters
- Monte-Carlo: systems are non-deterministic



Summary and Conclusion

- **Vitech's insight to MBSE:**
  - **Model-Centric, not Diagram-Centric**
  - **Unambiguous Notation**
  - **Views generated from the model**
  - **Complete, Query-able, Virtual System Prototype**
  - **The "STRATA" Layered approach to SE**

Vitech
Insight through integration