

The Wymore Construct for Systems Engineering

18 Nov 2010

Michael Dee

INCOSE FLC, MBSE Initiative, Systems Science Working Group

Special thanks

Jack Ring, INCOSE Fellow

A REQUEST FOR PARTICIPANTS...

- Monitor your impressions of this material. Then, answer the following questions
 - To what level will you say that the Wymore construct is merely common sense / intuitive?
 - To what level will you say that your organization utilizes this (or equivalent) construct for performing systems engineering in a disciplined sense?

THE DEBATE

- What is the essence of Systems Engineering?
 - INCOSE Discussion Forum “What is the Essence of SE”
 - Many viewpoints: DOD / Regulated / Free Market / etc.

*Contract/Regulatory
Compliance*

(1) SE as a Fiduciary Exercise:

- Work Products and Artifacts
- Reqs / Contracts / SOW
- Regulatory Compliance

*Effectively Develop
a System
(quality proxy)*

(2) SE as a Process:

- Life Cycle Process
- Technical Project Management
- INCOSE / IEEE / ISO-15288, etc

*Develop an Effective
System
(direct quality)*

(3) SE as Decision Making

- Value Proposition / Algorithm
- Predictive Models
- Information Generation

MOTIVATION: IS2010 in CHICAGO

- A tutorial was offered by a well established consulting firm:
 - *“Why Johnny Still Can’t Write Requirements”*
- Same consulting firm teaches the course:
 - *“How to Write Good Requirements”*
- Our response...
 - *Johnny still can’t write requirements **precisely** because he attended your first course!”*
- *The kernel of SE is in **engineering the requirements**, not writing “shall statements” or managing them in a RM tool*
 - *Requirements **Engineering** vs. Requirements **Management***
 - *“Writing Requirements Obfuscates Systems Engineering “*
 - *J. Ring, position paper*
 - *...their course material is not wrong - it is deceptively insufficient*

PHILOSOPHICAL STUFF...

- SE involves the making of decisions under conditions of risk and uncertainty
 - *Objective: Identify and eliminate uncertainty as soon as possible*
- The system will have an effect on its context
 - *Beware emergent properties – unintended consequences of the system*
 - *POSIWID: Purpose of the system is what it does (not what it was intended to do)*
- Stakeholders may be the problem
 - *Stakeholder interests are not automatically aligned with each other*
 - *The SE must resolve internal contradictions and conflicts*
- Customers don't want your product
 - *They want a solution to their problem!*
- **Job #1 of Systems Engineering is to define the problem**
 - *System failures are often due to solving the wrong problem*
 - *Ergo, Requirements Engineering is the kernel of SE*
 - *Solving problem is SE Job #2, and Job #1 for design engineering*

PROCESS COMPLIANCE vs. SYSTEM EFFECTIVENESS

An excerpt from “Operation Dark Heart”

- *Referring to special operations in Afghanistan...*

The trend in the Army had been to establish standards, train to those standards, and conduct operations to meet those standards.

The problem had become that achieving victory had been lost in the process – measures of performance became the measure to which one’s military success was held. [How well you followed process.]

What got dropped was the focus on the measures of effectiveness – or achieving victory - mission effectiveness.

Achieving and maintaining standards – even if those standards do not achieve victory – is the safest course of action [for your career]. Follow process no matter what.

Does this phenomenon occur in industry?

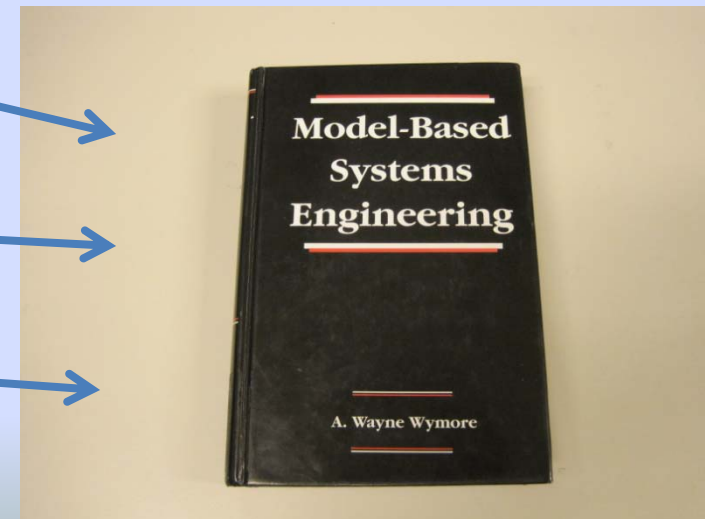
WHO IS WAYNE WYMORE?

- PhD Mathematician / SE
- Founded first academic SE department in US
 - University of Arizona
- INCOSE Pioneer
- INCOSE Fellow
- Author



WHY IS WYMORE IMPORTANT?

- Established the theoretical underpinning for SE
 - No prior theory of SE existed
- “Set-theoretic” approach
 - Distills essential information and activities for systems engineering
 - Coherent, logical, partitioned sets
 - Necessary and sufficient information set for SE in a canonical form
- Coined the term MBSE
- Defines 3rd viewpoint of SE
 - Engineering effective systems



DIFFERING NOTIONS OF MBSE...

- Wymore's Notion of MBSE
 - All engineering is the development of models
 - Requirements are an algorithm (a model) that characterizes the problem to be solved
 - The algorithm can compare any number of solution alternatives; find the best
 - Hence, Model Based SE

- INCOSE MBSE Initiative
 - Methods, languages and tools to support SE
 - Enable integrated information models
 - Requirements, Architecture, Behavior, Parametrics
 - *Eliminate document centric practices*
 - *An integrated model repository*

Model Based SE
(Methodology)

Supports

INCOSE MBSE Initiative
Languages & Tools

OVERVIEW OF THE WYMORE APPROACH

- Utilize a conventional life cycle model
- Emphasize problem definition and system effects – not requirements
 - *Focus on effects of the system on the problematic situation*
 - *Definition of system effectiveness must pre-exist engineering requirements*
- Create an effectiveness value algorithm to guide decisions
 - *A value model defines the design problem*
 - *Requirements (properly structured) are the value algorithm*
- Classify requirements against a general requirements taxonomy
 - *Establish formal mapping between requirements classes and design spaces*
 - *Requirements taxonomy strictly reflected in requirements relationships*
 - *Canonical – the mathematically simplest possible form*
- Clearly differentiate between functional and physical viewpoints
 - *Define “design spaces” for function, buildability, and implementation*
 - *“Form follows function.”*

WYMORE'S SYSTEM LIFE CYCLE PHASES

1. *Requirements Development*

2. *Concept Development*

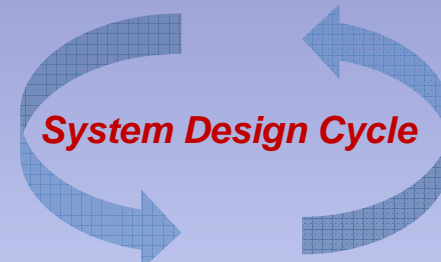
3. *Engineering Development*

4. *Support System Development / Production, Manufacturing, Deployment*

5. *System Test and Integration*

6. *Operations, Support, Modification*

7. *Retirement and Replacement*



Phases 2,3,4 are generally concurrent.

Phases 1 and 2 cannot be concurrent at the system level, but are concurrent at subsequent layers.

Otherwise: You will end up writing requirements to fit the design!

Trap: Single point design and the sunk cost fallacy!

SYSTEM REQUIREMENTS

- Purpose of system requirements...
 - *To state a design problem (not a solution) in terms of an algorithm for judging the fitness of a design alternative*
- Requirements Validation
 - *At least one **feasible** solution exists within the risk tolerance of the sponsor*
 - *Requirements reflect system effectiveness*
- System requirements must answer 6 questions...
 - *What is the system supposed to do?*
 - *How well must it do what it does?*
 - *What is available and allowable to build the system?*
 - *What are the criteria for judging how well resources have been utilized?*
 - *What are the trade-offs between performance and cost?*
 - *How can it be proven that the as-built system meets expectations?*

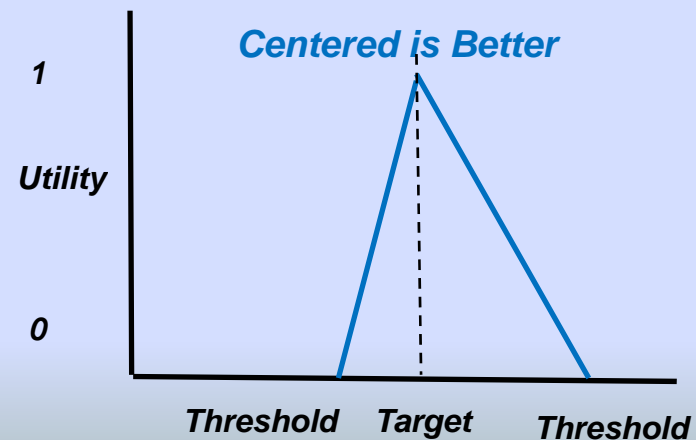
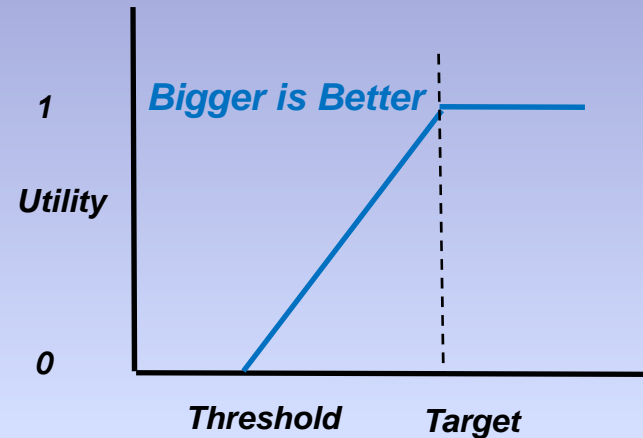
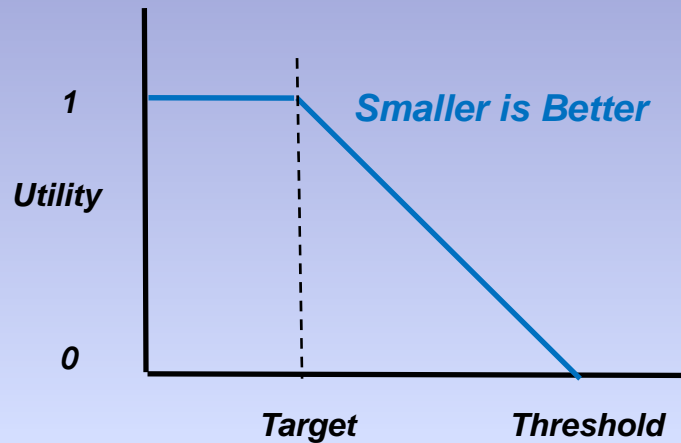
WYMORE'S REQUIREMENTS TAXONOMY

- Input / Output Requirements
 - *What does the system do? What doesn't it do?*
- Performance Requirements
 - *How well the does the system do what it is supposed to?*
- Cost Requirements
 - *How well have resources been utilized?*
- Trade-off Requirements
 - *What are the value trade-offs between performance and cost?*
- Technology Requirements
 - *What is available and allowable to build the system*
- System Test Requirements
 - *How can it be proven that the built system meets expectations?*

Answer the 6 questions...

UTILITY FUNCTIONS - BASIS OF THE REQTS ALGORITHM

- Can be linear or higher order...



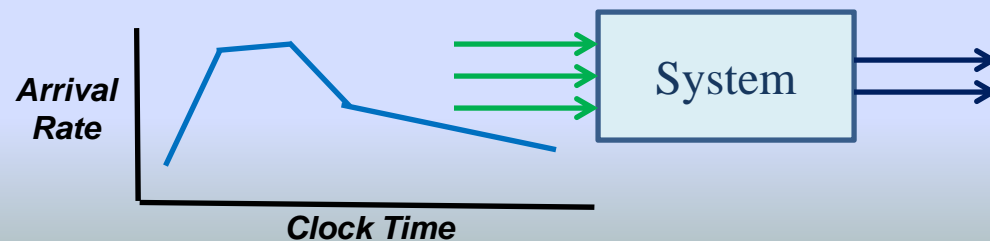
Utility Models for Measures

Target = Utility is saturated

Threshold = System is useless

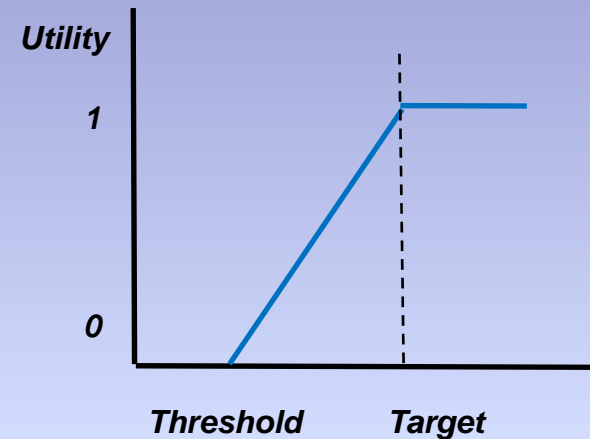
INPUT / OUTPUT REQUIREMENTS

- Treat the system as black box of functions defined by
 - *What goes in? What is imposed upon the system?*
 - *What comes out?*
 - *Without regard to performance*
- Specifies input and output in terms of “trajectories”
 - *Trajectories are a function of time*
- Includes things like...
 - *Transformation of specified inputs into specified outputs*
 - *Arrival and departing patterns of materials, energy, and data*
 - *Duty cycle*
 - *External environments*
 - *Etc.*



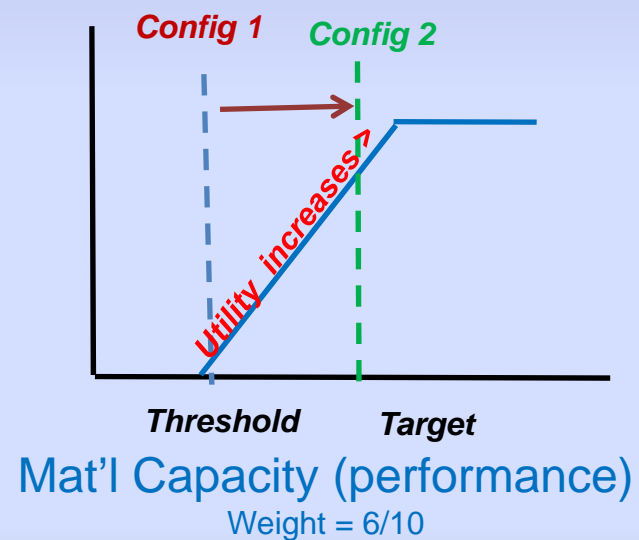
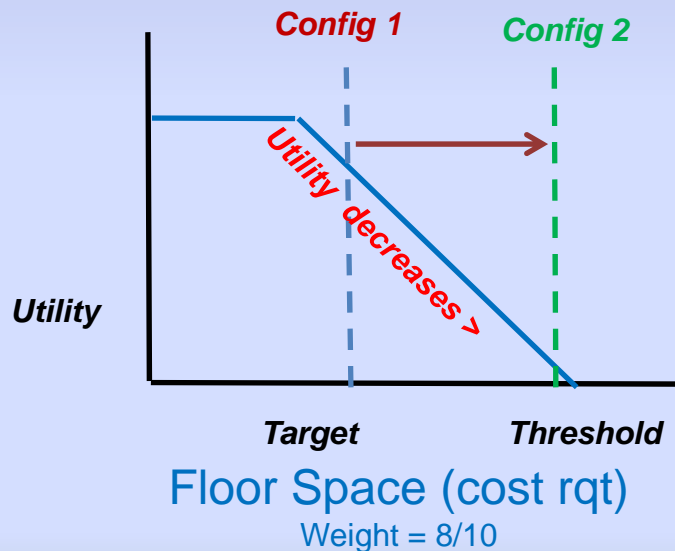
PERFORMANCE AND COST REQUIREMENTS

- Performance and Cost requirements express targets and thresholds as utility functions
 - *Related to maximum and minimum utility*
 - *Enable “concept selection” and trades*
- Performance requirements
 - *Define how well the I/O Requirements are met*
 - *Example: How fast an I-to-O transformation process must occur*
 - *Example: How robust a process must be against stated environments*
- Cost requirements
 - *Should consider entire life cycle*
 - *Includes: One-time engineering, production, delivery, support, retirement, operation, maintenance, and non-monetary resources consumed, etc.*



TRADE-OFF REQUIREMENTS

- Trade-off requirements are algorithms that define preferences between cost and performance requirements
- Trade-off algorithms can be expressed as weights on utility functions for performance and cost (or other ways)



TECHNOLOGY REQUIREMENTS

- Technology requirements constrain what is available from which to build the real system
 - *Specific technologies / prescribed interface standards*
 - *Components*
 - *Enforced leverage*
 - *Component or materials avoidance*
 - *Etc.*
- Some Design or Regulatory Standards
 - *Materials*
 - *Process*
- Technology requirements apply to the physical space
 - Not part of a functional allocation

SYSTEM TEST REQUIREMENTS

- Specifies for each “Implementable Test Item” (a.k.a. configuration item)...
 - **[Test] Observance:** Show how the real system will be observed, sampled, measured, stimulated, simulated etc. for each Performance, Cost, and Trade-off requirement
 - **[Design] Conformance:** Show in terms of test data that the real system is in conformance with the “implementable system design” (the design specification)
 - **[Requirements] Compliance:** Show that the real system is in compliance with the stakeholder requirements
 - **[Customer] Acceptance:** Show that the real system is acceptable to the customer

REQUIREMENTS DEVELOPMENT PROGRESSION

- Problem Situation (Customer Speak)
 - *Identify Problem / Context / Stakeholders / Approach*
- Operational Needs
 - *Identify how the system will interact with stakeholders and peers*
 - *Scenarios for each possible path*
- System Measures of Effectiveness (MOEs)
 - *Identify how major system capabilities are quantified*
 - *MOEs are stakeholder measures, not engineering measures*
- System Value Model; $V=f(\text{MOE Utility, Weight})$
 - *Deploy weights for Trade-offs*
- Measures of Performance (MOPs)
 - *Decompose MOEs into quantifiable MOPs*
 - *MOPs are the engineering basis for formal requirements*



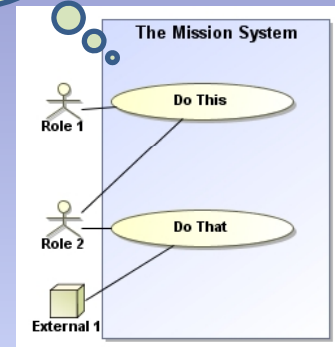
*Iterate
If
needed*

Note that we have not talked about “writing requirements” Focus on information.

PROBLEM SITUATION

*Can be documented
using
Use Case Analysis*

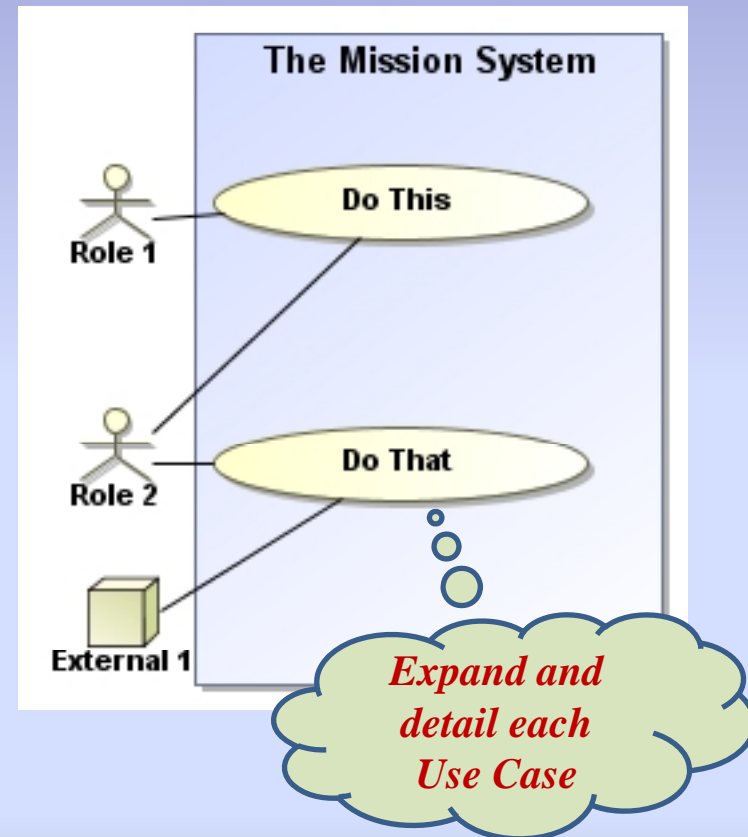
- Identifies the problem to be solved, in context
 - Identifies sources of requirements (stakeholders)
- Outlines the current situation
 - Including existing systems and context, if available
 - Can talk about competitive systems
- Proposes the means for extracting formal requirements
 - Defines the process for extracting requirements
- Identifies the Systems Engineering System
 - Project team / enterprise to define and integrate a solution
 - Incorporates the Systems Engineering Management Plan (SEMP)
- **Defines the approach, not the solution**
- **Is kept current throughout the life cycle of the system**
 - **Record of revisions over time (good for post-mortem analysis)**



OPERATIONAL NEEDS and CONTEXT

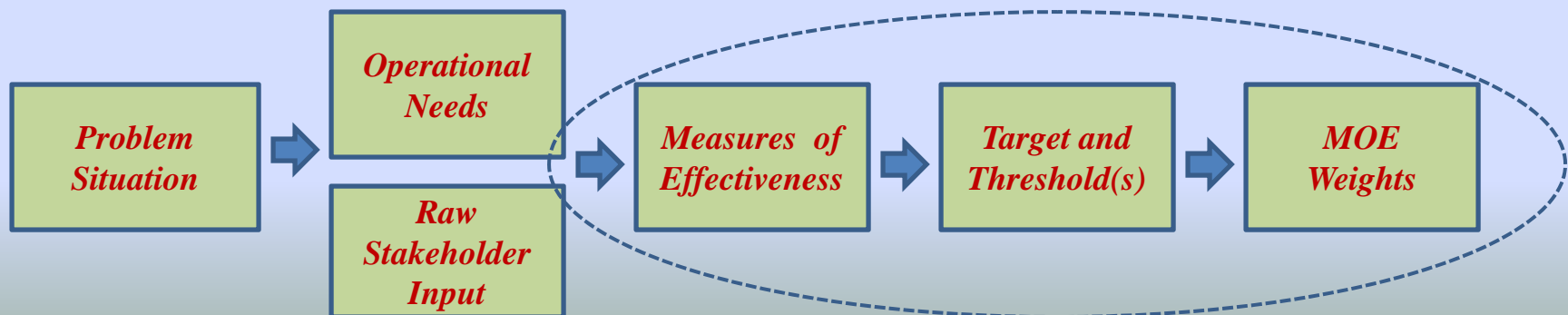
- Context describes all things with which the mission system will interact...
 - External Role Players (Actors)
 - External Peer Systems
 - System of Systems
- Collection of operational scenarios; how the system will be used by the actors based on I/O.
- Process threads documented by...
 - Flow Charts / sequence diagrams / etc.
 - Natural Language
- Scenario composition includes...
 - Goals, triggers and terminations
 - Input and output
 - *Capabilities sought*

The Use Case Method is one approach for organizing Operational Needs



MEASURES OF EFFECTIVENESS (MOEs)

- MOEs are formal criteria that contribute to a value proposition, extracted from operational scenarios
 - Derived from specific capabilities identified in scenarios
 - Identify effects of the system on the problem situation, + or -
- Target / Threshold(s) / Weight
 - Targets define the point of maximum utility from the MOE
 - Thresholds define zero utility point(s) from the MOE
 - Weights define importance of the measure to total system value



CONSTRUCTING A VALUE PROPOSITION

- A value proposition enables...
 - Trade-off studies – cost vs. performance
 - Feasibility studies – size of available design space
 - Comparison of alternatives
 - Avoidance of biases and politics (as much as possible)
- Ex: Relative value proposition for any alternative
 - Assumption: Value = Utility – Cost, over some time span
 - Cost can be treated as a negative utility

$$Va = \sum_{i=1}^{Ne} (We_i \cdot Ue_i)$$

Va = relative value of a single alternative

Ne = Number of MOEs

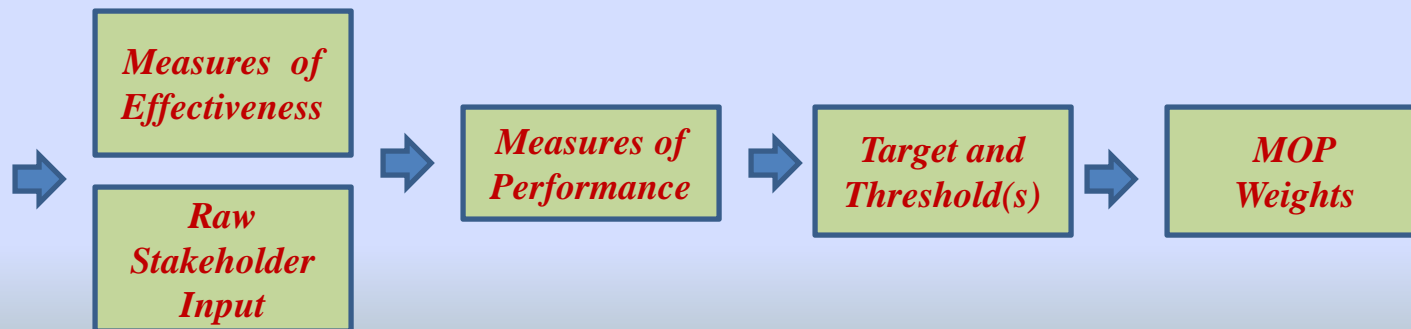
We_i = Weight of *i*th MOE

Ue_i = Achieved Utility of *i*th MOE

Note: An absolute value proposition can be constructed, say NPV, etc.

MEASURES OF PERFORMANCE (MOPs)

- MOPs define formal engineering measures
 - Basis for engineering requirements
 - Performance / Cost
- MOPs are derived from two sources
 - Decomposed from current set of MOEs
 - Extracted from stakeholder input not in the current MOE set
- MOPs are subject to standard criteria for requirements
 - Atomic / non-conflicting / non-contradictory / decoupled / etc...



DECOMPOSING THE VALUE FUNCTION

- Map each MOE utility into derived MOP utilities

$$Va = \sum_{i=1}^{Ne} (We_i \cdot Ue_i)$$

Va = relative value of a single alternative

Ne = Number of MOEs in the model

We_i = Weight of *i*th MOE

Ue_i = Achieved utility of *i*th MOE

$$Ue_i = \frac{\sum_{j=1}^{Np} (Wp_j \cdot Up_j)}{\sum_{j=1}^{Np} Wp_j}$$

Ue_i = Normalized achieved utility

Np = Number of MOPs in the model

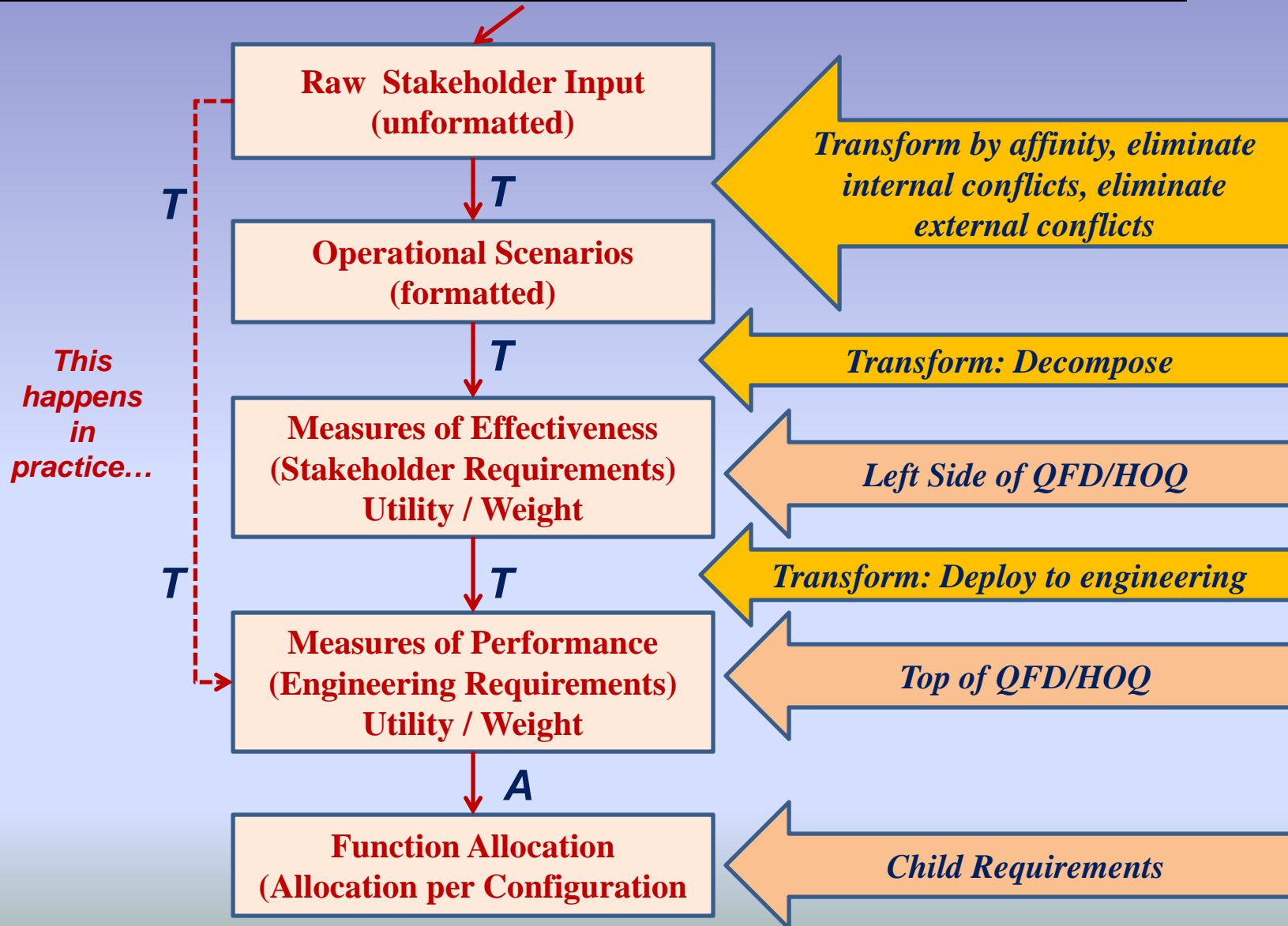
Wp_j = Weight of *j*th MOP w.r.t. *Ue_i*

Up_j = Achieved Utility of *j*th MOP

Note:

-An absolute value measure can be constructed using NPV or other measure

SUMMARY of WYMORE's REQUIREMENTS TRAIL



TYPICAL ARCHITECTURAL VIEWPOINTS

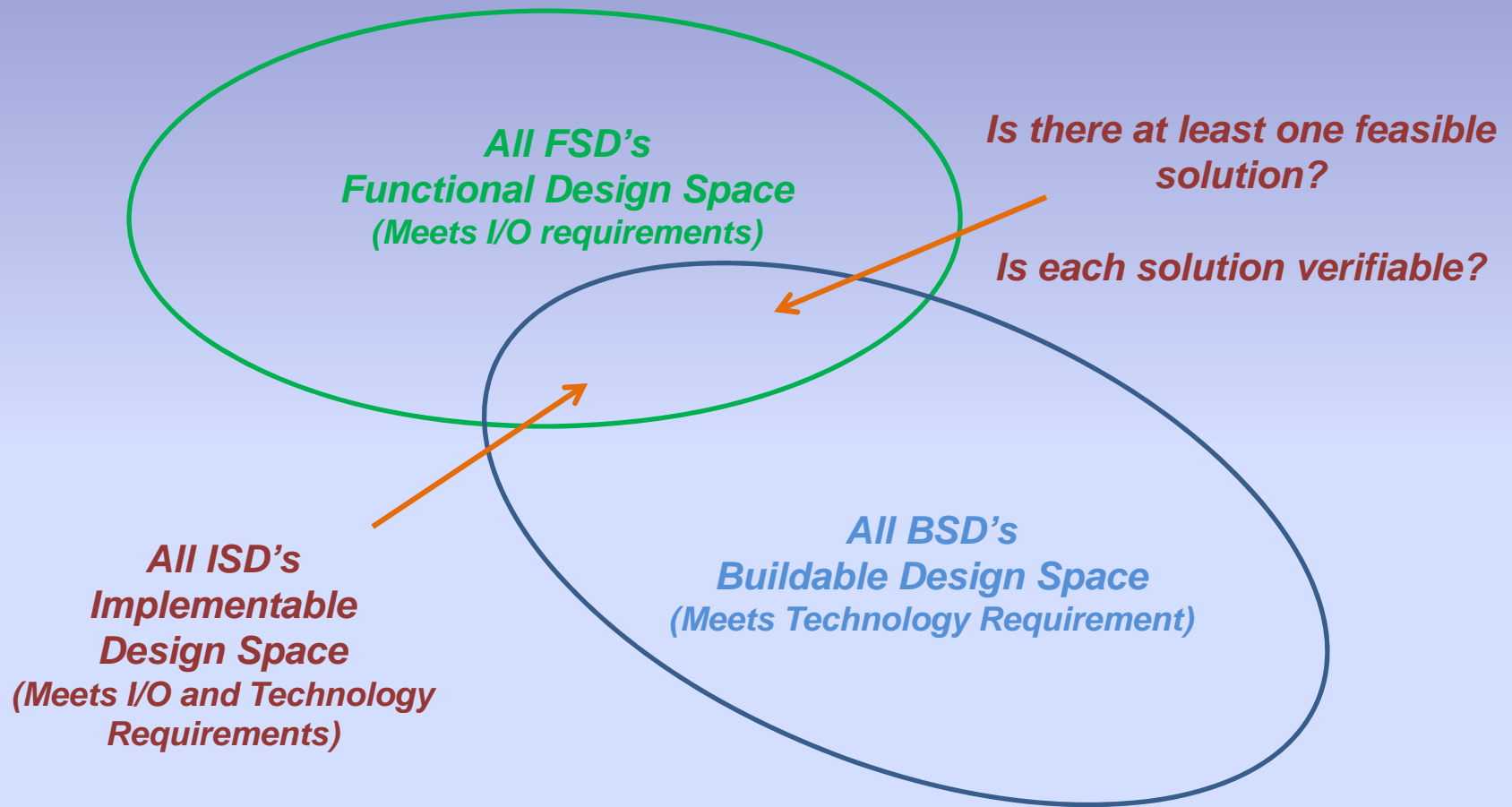
General (not directly from Wymore)...

- Functional / Logical Viewpoint (SE's live here)
 - *Abstract; organized by commonality of purpose or function*
 - *Consists of process algorithms and logical organizing principles*
- Physical Viewpoint (Design / Mfg live here)
 - *Deals with actual physical components (includes software)*
 - *Organized by physical connectivity and manufacturing process (BOM)*
- Technology Viewpoint (Disciplines live here)
 - *Cross-cutting implementation Themes*
 - *Electrical Architecture, SW Architecture, etc*

WYMORE CLASSIFICATION OF DESIGN SPACE

- **Functional System Design (FSD)**
 - *Any functional design that meets the Input / Output requirements*
 - *Becomes intertwined with the BSD*
 - *The Functional Design Space is the set of all possible FSDs [∞]*
- **Buildable System Design (BSD)**
 - *A system design that can be built I.A.W. the Technology Requirements*
 - *Does not imply that the buildable system design meets all requirements*
 - *The Buildability Design Space is the set of all possible BSDs*
- **Implementable System Design (ISD)**
 - *A system design that implements a Buildable System Design (BSD) that fully implements a Functional System Design (FSD)*
 - *The Implementable Design Space is the set of all feasible alternatives*

CLASSIFICATION OF DESIGN SPACE (cont)



WYMORE'S DESIGN SELECTION NOTIONS

- Best Choice vs. Optimization
 - *Pure global optimization is not practical; resources are limited.*
 - *Design selection must be made from a discrete set of alternatives.*
 - *Hence the danger of single point design!*
 - *The status quo is the zeroeth alternative; always score it!*
 - *Score competitive systems – where do your alternatives stand?*

ARCHITECTURAL FRAMEWORKS

- Wymore *does not* prescribe a specific hierarchical framework
 - *(ex: ISO15288 – System, element, subsystems, devices, components)*
 - *Wymore does address the nature of parent-child allocation*
- Wymore *does not* discuss optimal bounding of subsystems, etc.
 - *Affects complexity of the solution*
 - *Affects complexity of the organizational interfaces*
 - *Refer to methods such as Design Structure Matrix, etc. – (OOD!)*
- Wymore *does* discuss design concurrency between the functional system design and its physical realization.
 - *FSDs and BSDs are joined at each child layer*
 - *Each FSD can spawn more than one BSDs*

FUNCTION ANALYSIS AND ALLOCATION...

Define a logical (functional) hierarchy of logical layers, $k=0..n$

- Layer $k=0$ is the system layer
- Functional allocation begins with analysis of system I/O requirements

At each derived layer k , $k=1:n$... (ex: Element, Subsystem / Device...)

1. Decompose the parent I/O requirements into a set of inter-related I/O requirements based on a specific logical configuration
 - *Configuration and function are coupled – how many ways to skin the cat?*
2. Decompose the other requirements categories and allocate them to the I/O requirements at that layer
 - *Performance / Cost / Trade-offs*
3. Demonstrate (validate) that the solution defined by the ISD will resolve the design problem posed by the system requirements
 - *Create a complete performance estimate against the value proposition*
 - *Add this configuration to the list of ISDs*

Functional allocation and buildable design concepts will occur in couples, with at least one BSD for each unique FSD (Functional Allocation).

SUMMARY: PHASE 1 INFORMATION CONSTRUCTS

Information Class	Information Content for Requirements Phase
Problem Situation	Problematic situation; history of design project; method for soliciting information; definition of the Systems Engineering System (SEMP); define system boundaries and context; peers and actors.
Operational Needs	Stakeholder requirements (MOEs) traced to raw stakeholder input; properly classified using 6 categories; value classification via utility functions.
System Requirements	Engineering requirements (MOPs); value statement via utility functions; traceability to MOE from Operational Needs.
Requirements Validation	Demonstration that the design space is not empty for the System Requirements: at least one solution is feasible without internal contradictions.

SUMMARY: PHASE 2 INFORMATION CONSTRUCTS

Information Class	Information Content for Concept Phase
Concept Exploration (how decisions were made...)	<ul style="list-style-type: none">•Define each design alternative; provide value estimations of each•Provide supporting analysis; trade-off analysis; sensitivity; recommendations.
System Functional Analysis	<ul style="list-style-type: none">•Starting from system requirements, describe each FSD at each successive layer, K•Any FSD at layer K can spawn any number of FSDs at layer K+1•Allocate functions to support each I/O requirement•Validate FSD against system requirements.
Physical Synthesis	<ul style="list-style-type: none">•Defines the discrete set of ISDs, relative to the FSDs from which they emanate•Defines the physical realization alternatives

CONCLUSIONS

- Wymore defines the necessary and sufficient set of information needed to support decision making
- Wymore defines a canonical form for Systems Engineering practice.
- The difference between Wymore and other SE “Processes” is:
 - Recognition of Trade-offs as requirements in themselves
 - A value algorithm is needed for “requirements” to be complete.

CANONICAL FORM OF A REQUIREMENT

Attribute	Definition
Type:	MOE(Stakeholder), MOP (Engineering), or both.
Category:	One of these: I/O, Performance, Cost, Technology, Validation.
Measure:	Noun or phrase that defines a performance characteristic / units.
Weight:	Trade-off value for the measure; typically a relative scale (Trade-off Requirement)
Target:	The value of the measure at which maximum <u>desired</u> utility is achieved.
Threshold:	The value(s) if the measure at which utility becomes negative. At this point, the solution is not viable.
Condition:	(Applies to cost and performance only) The conditions under which the performance will be measured. References one or more I/O requirements.
Discussion:	Free field to allow clarification.
Rationale:	A statement or locator to the analysis that justifies the existence of the measure, and the values assigned to the weight, target and threshold(s).
Parent(s):	Pointer to the parent(s) of the requirement.

***Note the absence of any “shall statement”
Engineers do not make good novelists.***

ASSESSMENT...

- *To what level will you say that the Wymore construct is merely common sense / intuitive?*
- *To what level will you say that your organization actually utilizes this (or equivalent) disciplined construct for performing systems engineering?*

WYMORE'S RELATED "SYSTEMS"

Boundary systems that support development of the Mission System...

Systems Engineering System
Enterprise that develops the Mission System

System Test System
Process and infrastructure to support testing of the Mission System

Systems Analysis and Trade Study System
System to model, evaluate, and compare alternatives of the Mission System

Core of Wymore Construct

System Development System
Acquire and deploy the Mission System

Detailed Design System
Create engineering specifications & models that define the Mission System implementation

Create Design Output

USING TARGETS & THRESHOLDS

- A design alternative which scores below the threshold for an MOP or MOE is eliminated from consideration
 - Or, you must lower your standards...(Engineering rework!)
 - When setting the thresholds, ask at what point you no longer have an acceptable solution to the problem
 - Monitor predicted performance at each design review
- The target defines the point at which additional utility cannot be derived from increased performance
 - What happens if you have a segmented family of products?
 - At what point do you limit the performance of a segment?
- *Key: Get the Targets, Thresholds, and Weights right!*
 - *Subject to management bias!*
 - *Errors warp the design space and the decision space; cause overdesign*