
From the Complex Systems Engineering Series:

Integration of COTS:

Curse or Blessing?

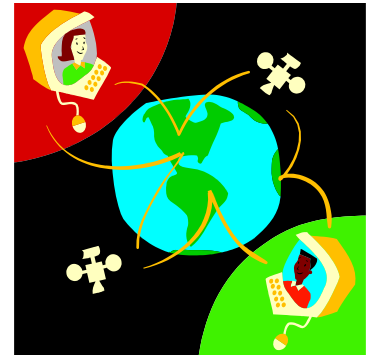
(Processes to make it the latter)

INCOSE San Diego Presentation, 26 April 2010

Mark Halverson

Presentation Contents

- COTS Integration into Complex Systems: **Background**
- COTS Integration: **Definitions**
- It's not that Traditional SE is Flawed, but.....
- Advantages of Using COTS
- Hazards of Using COTS
- COTS Integration Process Outline: Risk-Based Approach
- Conclusions
- Questions?



COTS Integration: Background

- Few complex system developments are completely from a blank piece of paper. There will always be some COTS or reuse [4].
- COTS are becoming more and more common. Even if a custom development would be desirable:
 - *Development costs are too high [4].*
 - *Schedule would be much too long for a custom design from “scratch”. Either the need is great (e.g., the CREW* anti-IED system), or the market moves so quickly (e.g., entertainment systems, cell phones, computer HW/SW).*
 - *For complex systems, evolutionary or spiral development has proven to be very effective: try it, and throw away what doesn’t work so well. (You don’t like to throw away an expensive custom development.)*

* CREW = Counter Radio-controlled IED Electronic Warfare

COTS Integration: Background (cont.)

- Many systems today are exclusively COTS, with possibly a little “glue” software.
- Today’s strong reliance on COTS is not the engineer’s choice (engineers love new developments, whereas COTS and reuse have been a headache), but rather, is driven by other stakeholders [4]:
 - *The marketplace, competition*
 - *Business interests (profit, market, quick innovation)*
 - *The customer (need, schedule, cost)*
 - *Teaming and partnering arrangements*
- Traditional systems engineering processes do not handle COTS well [2,4,6]. Very few companies have effective COTS and reuse processes.
- The emphasis has shifted from development to procurement [2,4]

COTS Integration: Background (cont.)

- Most “systems-of-systems”, and most “complex systems” involve COTS (the COTS items are some of the “systems” in the “system-of-systems”).
- With the increasing functionality and numbers of interfaces of the individual COTS items, the complexity of the COTS systems engineering problem grows geometrically.
- The most common experience with COTS is the ***under-estimation*** of the costs of dealing with, and mitigating the risks of, the integration of COTS into a **system** [4,5,8]. *This biases the results of the make-buy trade study towards using COTS, perhaps leading to an inappropriate system design.*

COTS Integration: Background (cont.)

- Most literature on COTS and “reuse” are software based, and don’t maintain a balanced systems engineering perspective.
- The purpose of this study is to look at the issues from a SE perspective.

COTS Integration: Definitions

- **COTS:** This will be used as a generic term for all pre-existing equipment of a defined form (commercial, MOTS, GOTS, NDI, etc.). COTS = “for sale” or “someone gives it to you” [3]
- **COTS-Based System:** a system that will contain predominantly COTS items (possibly with new designs as well as “glue” components used to adapt the COTS items), and that will be specifically designed to adapt to COTS [4].
- **COTS-Involved System:** either an existing system to which one or more COTS items are to be added, or a new design system that will contain one or more COTS items (but the system design will not be driven/influenced by the fact that COTS items will be used).

COTS Integration: Definitions (cont.)

- **Reuse:** the application of a previously developed item to a new system with little or no modification. Depending upon your knowledge of the item, and how well documented it is, “reuse” may be very similar to the application of COTS items.
- **Black Box Item:** an item for which some external specifications are available, but where you know nothing of its internal design or the processes used during its design.
- **Glass Box Item:** an item where you have complete access to its internal design, the processes used during its development, and the associated documentation.

COTS Integration: Definitions (cont.)

- **Traditional Systems Engineering (TSE):** the systems engineering processes defined by industry best practices, and documented in the INCOSE Systems Engineering Handbook [1] (among other sources).

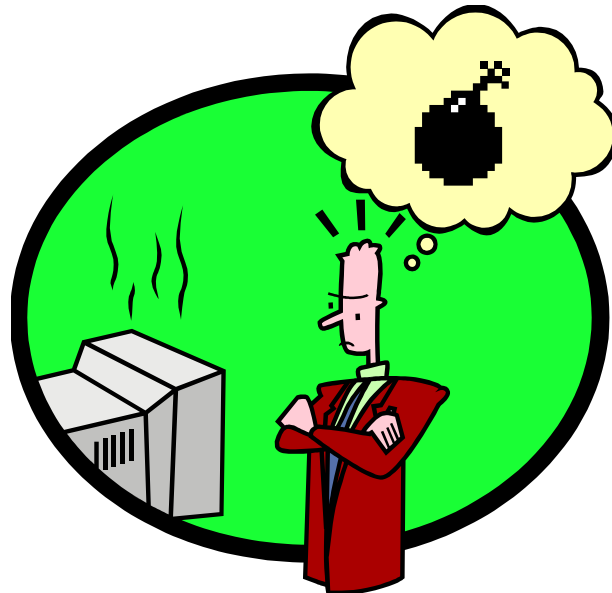
It's Not that TSE is Flawed.....

- Traditional systems engineering (TSE) is still the basis for our understanding of the art and science of systems engineering.
- In many situations, traditional systems engineering will still be the best approach.
- It is just that circumstances sometimes conspire against the use of TSE alone.....



A Process for COTS Integration into Systems

- OK, so you can't avoid using COTS, then you had better have a way of handling them.....



Advantages of Using COTS

- **If** appropriate systems engineering processes are used:
 - *Reduces Development Costs, recurring costs, and support [5]*
 - *Shortens Schedule [5]*
 - *Reduces risk by using proven designs and technology [5]*
 - *Due to many users, latent defects tend to become known*
 - *Due to many users, updates tend to be made automatically (COTS serve the marketplace and strive to keep up with it [4])*
 - *Due to many users, interfacing items as well as support items tend to come on the market, and bugs are found/fixed [5]*
 - *Tends to promote open architectures and modularity, and therefore multiple sources [5]*
- **However, there are many potential pitfalls.....**

Hazards of Using COTS

- **Trouble with configuration management** [3,5,8]
 - **Config Identification:** different suppliers with incompatible number systems; same supplier can change part number without notice
 - **Change Management:** Supplier can change item characteristics, as long as it still meets the spec; you may be depending upon unpublished behaviors (COTS specs are often ambiguous or incomplete) [4]
 - **Compatibility and Interoperability:** because of the above, there is always uncertainty over how well the COTS will be compatible with the system (now and in the future).
 - **Modified COTS:** Avoid if you can! This opens the door to complete chaos (use of designed-in options, tailoring, is OK) [5]

Hazards of Using COTS (cont.)

- ***Supplier/Subcontract Management Challenges***
 - ***Quality Flow-Down:*** *we must maintain a uniform level of quality throughout the system, but there is no way to match the quality of the COTS item(s) to the remainder of the system.*
 - ***Diminishing Manufacturing Sources:*** *The COTS item is at the core of your system, but the market place moves on. Your small program will have no influence on the suppliers. You will often end up with single-sources, but no option to pay for “build-to-print”.*
 - ***Obsolescence:*** *the life of commercial items is typically much shorter than for military items. You must assume that your COTS items will become obsolete every few years (and sometimes without replacement) [7].*

Hazards of Using COTS (cont.)

- ***Compatibility and Interoperability in Doubt***
 - ***Functionality and Performance:*** *the spec sheet for a COTS item rarely has specific and complete info. The interfaces are very complex and difficult to document.*
 - ***Operating Environment:*** *each design of a COTS item assumes explicit and implicit environmental conditions. It is very difficult to find out what they are, and match this to the overall system [4].*
 - ***CONOPS and Behavior:*** *every design is based upon some vision of the concepts of operation and the necessary behaviors for the item. Usually not documented*
 - ***Design Assumptions:*** *each design is based upon assumptions about where and how the item will be used. These are typically not documented [4].*

Hazards of Using COTS (cont.)

- **Design Assurance Level Can NOT be Achieved** [5]
 - **Safety and Reliability:** *as the COTS items become larger and more complex, there is a need to analyze the functions internal to the item. But because the COTS items are often proprietary, this information is not available to the integrator. Also, no hazard analysis or FMECA are available (since these tend to be application-specific)*
 - **Airworthiness:** *airworthiness requires insight into the internal design of the COTS item. Test reports are also needed. As above, all of this is typically not possible for COTS (or comes at a very large extra cost).*
 - **Information Assurance:** *as with the above, detailed access to inside the COTS item is needed. As above, this is typically not possible for COTS.*

Hazards of Using COTS (cont.)

- **Design Assurance Level Can NOT be Shown [3]**
 - *For safety, airworthiness, information assurance, as well as quality assurance, the engineering development processes used are indicative of the ultimate item integrity. “Black box” testing alone can **NOT** prove the item’s safety, airworthiness, and quality.*
 - *The COTS items rarely come with detailed process information. Even if they did, it might not match the specific requirements (DAL) on your program.*
 - *Industry practice and the regulatory authorities require “glass box” access to all complex items. Treating COTS as “black box” items will not be able to prove compliance with safety, airworthiness, information assurance, or quality requirements [5].*

Hazards of Using COTS (cont.)

- ***Added Risks in Logistic Support of COTS***
 - *Dependence on the supplier, loss of control [5].*
 - *Test, support, and repair concepts may not be compatible with the rest of the system.*
 - *Maintenance and spare parts for the COTS item are not assured [5].*
 - *System technicians are not able (not allowed) to maintain the COTS items [5].*
 - *Due to obsolescence, the COTS item itself may become suddenly unavailable [5].*
 - *Updates and extensions may not be possible if the supplier disappears, or does not want to [5].*

COTS Integration Processes

Wow, that's scary!

Let's look at some SE processes that can keep us safe.



COTS Integration Processes: Risk-Based Approach

- **COTS Hazards:** The bottom line is that one is confronted with high levels of uncertainty whenever COTS items are integrated into the system
- **Consequences:** can be very severe
- In this situation, the application of risk management techniques is the only rational approach [5,6,8]. Therefore, I am proposing a risk-based approach to using COTS.
- The following processes will depict the full set of activities to eliminate or mitigate the risks. For any specific program, a subset of these will be used, depending upon the risks and program requirements.

COTS Integration Processes

- There are three phases of COTS integration:
 1. *Phase 1: Designing the system for **future** COTS integration*
 - This is rare, but nice if you can do it
 - Allows for efficient product lines, and easy future extensions/improvements
 - Necessary to remain competitive in tomorrow's marketplace
 2. *Phase 2: Actually integrating the COTS into the **current** system*
 - If Phase 1 (see above) were not done, then this task can be difficult (this is a mine field of potential problems)
 - This involves not only technical/functional issues, but also requires special support functions
 - *Configuration management*
 - *Quality Assurance*
 - *Supply chain, subcontractor management*
 - *Airworthiness, Safety, Reliability, Information Assurance*
 - Experience has shown that traditional systems engineering processes **will not produce reliable results** when integrating COTS into systems

COTS Integration Processes (cont.)

- *Phase 3.* Designing a hardware or software item expressly for supporting its future “reuse” is the topic of another presentation; will not be discussed further here.
- Although very relevant to today’s topic, it will not be discussed further.
 - *Let’s assume that where applicable, that has already been done.*

COTS Integration Processes (cont.)

- An outline of the recommended systems engineering processes for COTS integration (Phase 1 and Phase 2) is given in the following charts.
- For a particular program, a risk assessment should be done in order to anticipate the areas where problems due to the COTS items could be expected [5,6,8]. Apply the following processes as appropriate for your program.
 - *Depending upon the system, some of the following activities will need more, or less, emphasis*

*Phase 1: Designing the System for **Future COTS** Integration*

- *The architecture should be specifically designed for COTS integration (strive for “**solution stability**”)*
 - *Use the system-of-systems approach; highly integrated, tightly coupled complex systems do not adapt well to COTS*
 - *Study the marketplace to find the best and most likely to be long-lived standards. Also look for the highest quality suppliers and multiple sources. Perform trade studies to select the best. Design the preliminary system architecture around these type of COTS items [2].*
 - *Use the modular open-system approach (MOSA)** as the basis for the preliminary architectural design*
 1. Establish an enabling environment for open architectures
 2. Employ modular design
 3. Designate key interfaces
 4. Use open standards (not just for external interfaces, but also for internal interfaces)
 5. Certify conformance

** Refer to separate MOSA presentation in the Complex Systems Engineering series, July 2007

Phase 1: Designing the System for *Future* COTS Integration (cont.)

- *Define plans and specifications for the following, then flow down to the COTS selection/procurement process*
 - *Define the **environmental and EMC conditions**, transform down to the box level, then flow down to the COTS item selection/procurement*
 - *Define the required **configuration management** characteristics, then flow down to the COTS item selection/procurement*
 - *Define the **quality assurance** and quality system management requirements, then flow down to the COTS item selection/procurement*
 - *Define the plan for **logistic support** (repairs, spare parts, training, etc.), then flow down to the COTS item selection/procurement*
 - *Define the needs for the **design assurance level**, then flow down to the COTS item selection/procurement*
 - For high-reliability and safety-critical applications, special emphasis needed for “complex” functions
 - *Hardware (RTCA/DO-254)*
 - *Software (RTCA/DO-178B)*

Phase 1: Designing the System for *Future* COTS Integration (cont.)

- *With a COTS-based system design, the systems engineer must have realistic expectations.*
- *A system designed to integrate COTS into the final solution is going to be sub-optimal.*
- *It can be faster, better, and cheaper. But it can't also optimize SWAP (size, weight, and power) at the same time.*
- *Get over it.....*

Phase 2: Integrating COTS into the Current System

- ***Assumed assignment:***
 - *A COTS-based system is to be developed*
 - *The top-level system requirements and CONOPS are given*
 - *Some COTS items are pre-determined (e.g., GFE in the contract), other COTS items may be selected by the designer, and the remainder of the system is a new design*
 - *The systems engineer is free to design the remainder (non-COTS parts) of the system*
 - *Phase 1 may, or may not, have previously been completed*

Phase 2: Recommended Process

- **Develop and analyze the System Requirements; Be Clear on the Constraints** (*i.e., the non-functional requirements*)
 - *This is the same as with traditional systems engineering*
- **If Phase 1 were not done (for designer-selected COTS):**
 - *Research the industry standards,*
 - *Research the COTS marketplace,*
 - *Flow down the top-level requirements to the COTS item level*
 - *Perform a preliminary architectural design*
 - *Select the COTS to be used*
- **For each COTS item, develop abstract models for the functionality, behavior, and interfaces [7]** (this is the reverse engineering step, which will uncover unknowns, and reduce risk)

Phase 2: Recommended Process (cont.)

- *Develop a top-level, problem-space model for the entire system, but use the individual COTS (solution space) models instead of abstract functional models*
 - *This allows for a complete analysis of the problem space, thereby defining those functions that are to be newly developed (i.e., that are not COTS).*
- **Analyze the problem space**
 - *Identify missing functions or interfaces*
 - *Identify any missing, incomplete, or conflicting requirements*
 - *Identify any issues with the COTS item selections*
 - *Engage in a lively query system with the customer and with suppliers*
 - *Obtain agreement upon any changes that become necessary*
- **This is fundamentally an iterative process [2,4].**

Phase 2: Recommended Process (cont.)

- *Design the final architecture to transform the problem space into a specific solution, which incorporates the selected COTS items* (note: the COTS items bound the solution space)
 - *Perform trade studies to assess alternatives and optimize the design*
 - *Architecture should be flexible and easily adapt to future changes*
 - *Analyze the final solution to make sure that the top system requirements will be fulfilled*
- **Generate the COTS item purchase agreements**** [2]
 - *Identify the functions, their performance, and interfaces needed*
 - *Identify the design, manufacturing, and test processes to be used*
 - *Identify the documentation and evidence of compliance to be provided*
 - *Identify any change/configuration management requirements*
 - *Identify the required quality assurance and quality management system requirements*
 - *Identify any logistic support requirements*

Phase 2: Recommended Process (cont.)

- *Define and implement a proactive obsolescence management plan [4]*
- *Integrate and Test the System*
 - *This is unchanged from traditional systems engineering processes*
- *Refer to the Systems Engineering process models depicted graphically in the back-up slides.*

Examples of Specific Issues to Be Considered (lessons learned)

- One may be able to choose from a variety of COTS item models and suppliers. The choice must consider the flow-down of system needs:
 - *Is a FMECA needed? Does this supplier have one? How much will it cost?*
 - *For airworthiness, untold problems can be saved by selecting a COTS item that was **already certified***
 - To what Design Assurance Level was it certified (A thru E)?
 - But the documentation must be made available
 - A TSO is even better
 - *Is there any HW or SW complexity in the COTS item? How was it handled?*
 - *How is the configuration identified? How will we be notified of changes?*
 - *What is the supplier's quality management system? Is it compatible with ours? Will it meet the contract requirements?*
 - *What previous applications used this COTS item, and how similar are they to our application?*
 - *Were environmental and EMC qualification tests performed? How complete are they? Did anything fail? Are the test procedures and test reports available?*

Examples (cont.)

- The selection of the COTS items is critical.
 - *The decisions made at this point will be critical, and will have a considerable impact on project success or failure.*
 - *Most projects are assigned under schedule pressure and necessitate quick decision making, usually in the face of unavoidable uncertainty [2].*
 - *Good luck.....*

- The selection of suitable COTS products is often a non-trivial task, and requires careful consideration of multiple criteria [2].

Conclusions

- The real novelty here, and the point of this whole presentation, can be summarized as follows:

When using a COTS item, you must not only use its literal definition in the solution space, but first you should use its abstracted definition in the analysis of the problem space. That will solve a lot of problems, greatly reduce risk, and give you traceability to the top requirements.



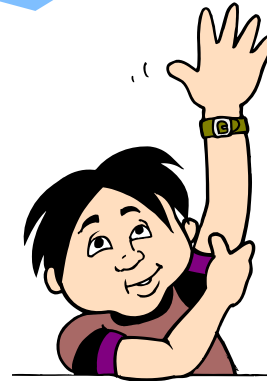
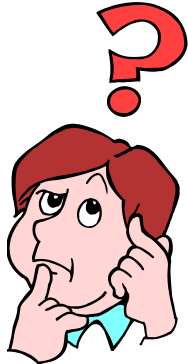
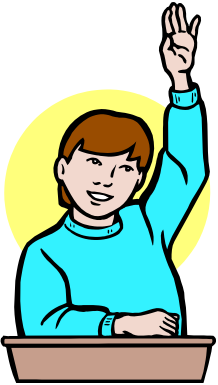
Conclusions (cont.)

- There are many potential advantages to a COTS-based system, but the traditional systems engineering process must be modified to assure success
- A COTS-based system is always a compromise between competing stakeholder needs
- The process for COTS-based systems must balance [4,5]:
 - *System requirements*
 - *Marketplace trends and available COTS items*
 - *System architectural design*
 - *Program risks and stakeholder demands*
- A risk-based SE approach can allow you to reap the benefits, but mitigate the risks, of COTS-based systems [8]

Conclusions (cont.)

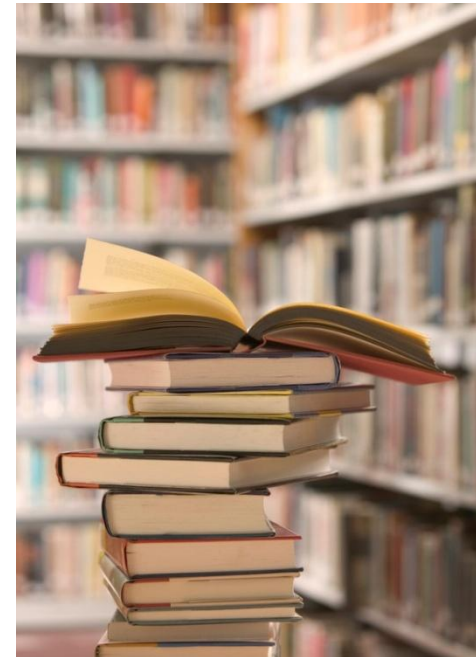
- With a COTS-based system, the systems engineer will have to spend greater efforts in managing the procurement process, suppliers, and the business environment.
- With a COTS-based system, you are no longer in full control. The marketplace and suppliers retain ultimate control over the critical components of your system [4].
- Get over it..... and deal with it.

Questions?



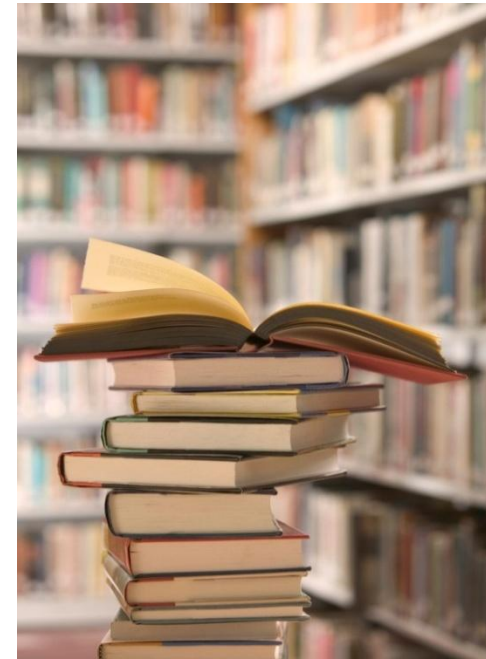
Bibliography

1. INCOSE Systems Engineering Handbook, Version 3
2. “Requirements Engineering for COTS Selection”, C. Alves, F. de Alencar, J. Castro, published in Workshop de Engenharia de Requisitos WER2000, 2000
3. “COTS Integration Issues”, Aerospace Corp. PowerPoint Presentation, S. Eslinger and K. Owens, 2001
4. “Evolutionary Process for Integrating COTS-Based Systems (EPIC)”, C. Albert and L. Brownsword, CMU/SEI-2002-TR-005, SW Engineering Institute, November 2002



Bibliography (cont.)

5. “SysML Modeling of Off-the-Shelf-Option Acquisition Risk Mitigation in Military Programs”, J. Constantine and S. Solak, Wiley Interscience, March 2009
6. *“Risk Based COTS Systems Engineering Assessment Model: A Systems Engineering Management Tool and Assessment Methodology to Cope with the Risk of COTS Technology Insertion”*, R. Lebron, R. Rossi, and W. Foor, RTO SCI Symposium Paper, October 2000
7. “Systems Engineering in a COTS World”, R. Sorensen, Vitech Corp, 2004
8. “FAA COTS Risk Mitigation Guide”, Rev 3.2, January 2010



Back-Up Slides

Let's Back Up and Review the Top-Level SE Taxonomy

- INCOSE and most textbooks address only the TSE macro process.
- Experienced systems engineers know that life is not that simple. Research is being done on other types, to complete the taxonomy of systems engineering.
 - *Traditional Systems Engineering (TSE)*
 - *Incremental (Iterated) TSE*
 - *Extreme Systems Engineering*
 - ***Complex Systems Engineering***
- Each SE macro process type has its place, and appropriate fields of application.
 - *Note that there are circumstances for which no rational SE process exists.*

Top-Level SE Taxonomy (cont.)

Rational Systems Engineering Macro Processes

Mission and System Rqmnts Known and Fixed?	All Techn. and Standards Available Now?	Is Program Schedule-Driven?	Is Program Budget-Driven?	Is There Central Control of all System Elements?	Appropriate Type of Systems Engineering to Apply	Type of Development Life Cycle(s) Used	Comments
Yes	Yes	Yes (within limits)	Yes	Yes	TSE	Waterfall (Big Bang)	This will produce the least waste and the least development costs, if the entry conditions are satisfied.
Only Key Requirements and Missions	No	No	Yes/No	Yes	Incremental SE	Iterated Waterfall	This is probably the most common real-world macro process in use.
Only Key Requirements and Missions	Yes	Yes	No	Yes	Extreme SE	Parallel, Waterfall, Evolutionary	Can reduce schedule with careful planning and management, but tends to be chaotic.
No	No	No	No	No	CSE	Incremental, Evolutionary, Spiral	This is a new field, where little has been published about the successful strategies to support it.

Note: other combinations of circumstances are theoretically possible, but there may not be a rational SE macro process that corresponds to their needs.

How Do You Know When CSE is Needed?

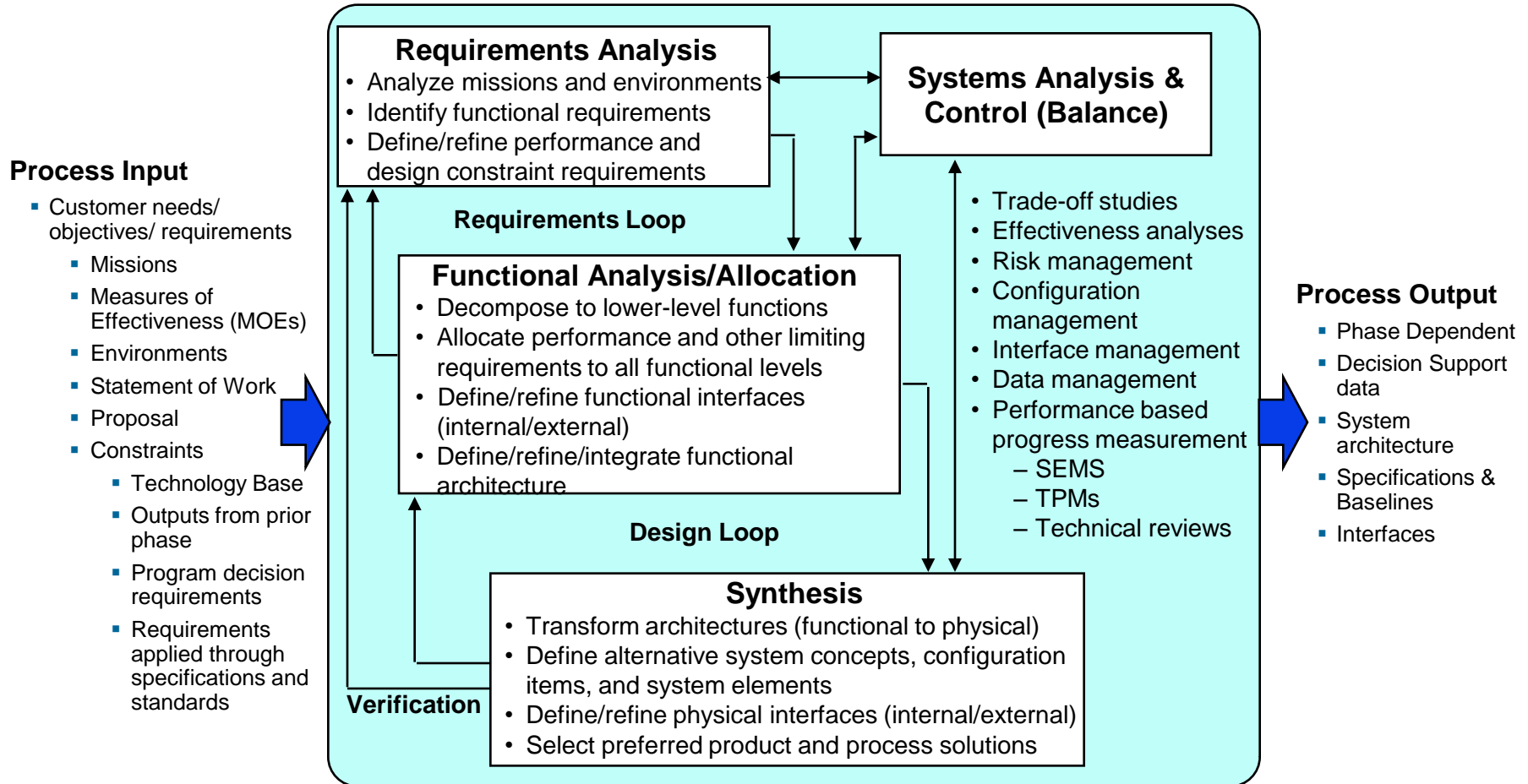
- The universe of systems engineering is divided into TSE, CSE, and Extreme SE. (See later chart.)
 - *TSE can be considered to be a special case (subset) of CSE. TSE may be useful within the application of CSE.*
 - *Do not apply TSE where CSE is needed. Must be able to recognize when the system is “complex”. See next chart.*
 - *Extreme SE is a topic for a subsequent presentation.....not examined further here.*
- Complex Systems Engineering (CSE) is needed when two or more of the key criteria are present.
 - *The columns of the next chart are the key complex system criteria.*
 - *CSE employs different approaches and techniques in order to deal with the complexity.*

System Complexity Chart - Examples

Project	Key Capabilities Unknown at Start?	Key Requirements Unknown at Start?	Users and Mission Unknown at Start?	Dispersed Control of Elements?	System Elements Variable, Adaptive?	Independent Change Agents at Work?	Complete System Test Not Feasible?	Dominant Emergent Behaviors ?	System of Systems?
Manhattan Project	No	Yes	No	No	Yes	No?	No	No?	No?
NASA Man to the Moon	No	Yes	No	No	Yes	No	Yes	No?	Yes
FAA AAS	No	Yes	No	No?	Yes	Yes	Yes	Yes	Yes
Internet	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
JSF CNI	No	Yes?	No	No	No?	Yes?	No	Yes?	Yes
NCOW RM/GIG	Yes?	Yes	No?	Yes	Yes	Yes	Yes	Yes	Yes
The PC	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No?

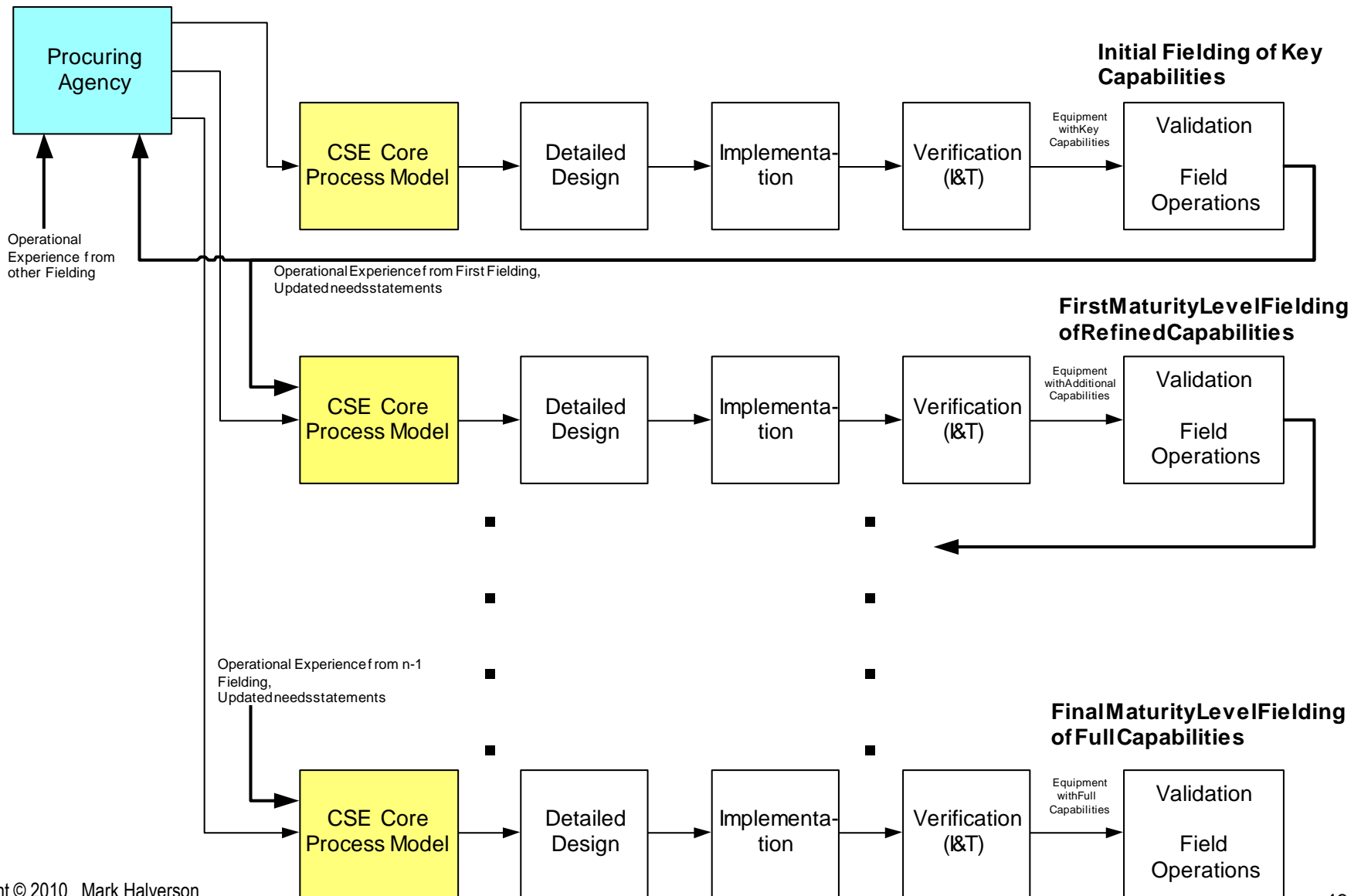
A “Yes” in any two columns is sufficient to make the system “complex”. The more columns with “Yes”, then the more complex the system is.

First, We Need to Define the Terms (cont.)

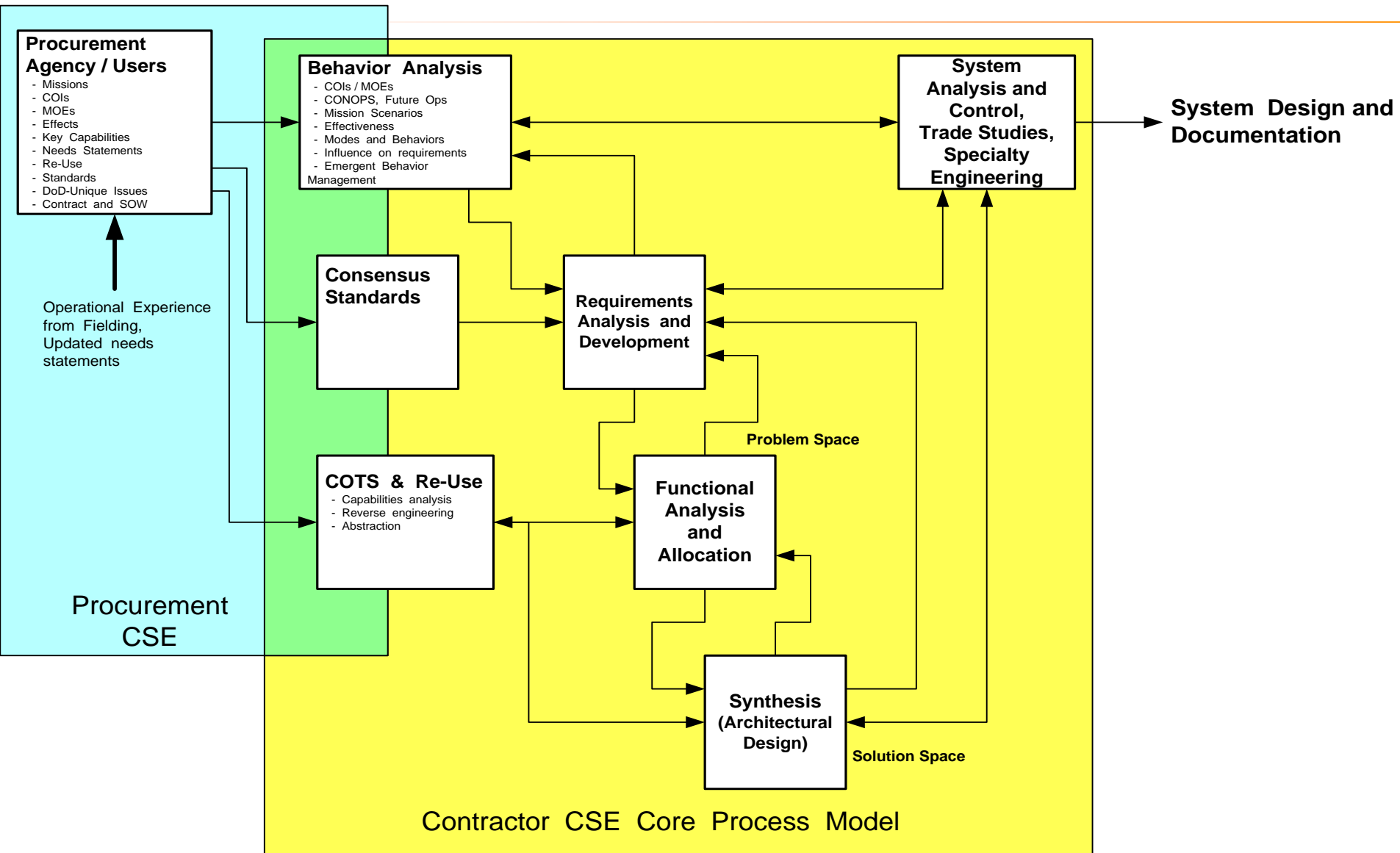


Traditional Systems Engineering Macro Process

The Evolutionary/Spiral Nature of CSE



The CSE Core Process Model



The CSE COTS Process Model

